Content Manager OnDemand
Version 10 Release 5

Web Enablement Kit Implementation Guide

SC19-3353-03

Note

Before using this information and the product it supports, read the information in “Notices” on page 99.

This edition applies to the following products and to all subsequent releases and modifications until otherwise indicated in new editions:

- IBM® Content Manager OnDemand for z/OS®, Version 10 Release 5
- IBM Content Manager OnDemand for Multiplatforms, Version 10 Release 5
- IBM Content Manager OnDemand for i, Version 7 Release 3

Contents

ibm.com® and related resources

Product support and documentation are available from ibm.com®.

Support and assistance

From ibm.com, click **Support & downloads** and select the type of support that you need. From the Support Portal, you can search for product information, download fixes, open service requests, and access other tools and resources.

IBM Knowledge Center

See your online product information in IBM Knowledge Center at one of the following locations:

- For IBM Content Manager OnDemand for Multiplatforms, see https://www.ibm.com/support/knowledgecenter/SSEPCD
- For IBM Content Manager OnDemand for z/OS, see https://www.ibm.com/support/knowledgecenter/SSQHWE

PDF publications

See the following web sites for PDF publications for your product:

- For IBM Content Manager OnDemand for Multiplatforms, see https://www.ibm.com/support/pages/node/1079037.
- For IBM Content Manager OnDemand for z/OS, see https://www.ibm.com/support/pages/node/1079043.

Contacting IBM

For general inquiries, call 800-IBM-4YOU (800-426-4968). To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

For more information about how to contact IBM, including TTY service, see the Contact IBM website at http://www.ibm.com/contact/us/.

Chapter 1. IBM Content Manager OnDemand Web Enablement Kit overview

ODWEK provides a programming interface that can search for and retrieve documents from Content Manager OnDemand servers.

ODWEK allows users to access data that is stored in an IBM Content Manager OnDemand server with IBM Content Navigator or a user-written program. An application uses the Java APIs to verify permissions, manage hit lists, and return data. For example, ODWEK verifies that the user information is valid on the Content Manager OnDemand server, such as permission to access the server and data stored in an application group. After the user submits a search, the ODWEK Java API returns a list of the documents that match the query. The user selects a document to view and IBM Content Navigator or the user-written program sends the document to the browser.

The following figure illustrates how a workstation with a web browser accesses data stored in a Content Manager OnDemand server.

Figure 1. Accessing data stored in Content Manager OnDemand by using ODWEK

ODWEK can search for and retrieve documents from Content Manager OnDemand servers that are running any currently supported versions of Content Manager OnDemand.

IBM Content Manager OnDemand Web Enablement Kit components

ODWEK is made up of a programming interface and viewer software.

ODWEK contains the following components:

- The ODWEK Java™ Application Programming Interface (Java API). APIs provide a way to access Content Manager OnDemand data from user-written programs. The programming interface uses standard Content Manager OnDemand interfaces and protocols to access data stored in a Content Manager OnDemand server. No additional code is needed on the Content Manager OnDemand server to support ODWEK.
- The IBM Content Manager OnDemand Advanced Function Presentation (AFP) Web Viewer. Users can use the AFP Web Viewer to search, retrieve, view, navigate, and print AFP documents from a web browser.
- The IBM Content Manager OnDemand Image Web Viewer. Users can use the Image Web Viewer to search, retrieve, view, navigate, and print BMP, GIF, JPEG, PCX, PNG, and TIFF documents from a web browser.
- Java Line Data Viewer. Users can use the Java Line Data Viewer to view line data documents from a web browser.

Important: To view other types of documents stored in Content Manager OnDemand, you must obtain and install the appropriate viewer. For example, to view Adobe Portable Data Format (PDF) documents, you can obtain the Adobe Acrobat viewer and install it to the browsers used in your organization.

Managing access to Content Manager OnDemand through ODWEK

The most common method of managing access to Content Manager OnDemand is by defining one user ID.

Most customers define one Content Manager OnDemand user ID to access a server with ODWEK. This situation is common in environments with many casual users of Content Manager OnDemand who access

the same folder. You can also provide each user with their own Content Manager OnDemand user ID. Regardless of how you decide to access Content Manager OnDemand with ODWEK, you must manage the user IDs in Content Manager OnDemand: you must add them to the server and set application group and folder permissions for the users.

Use cases of ODWEK Java APIs from a business perspective

Most businesses need to provide access to their Content Manager OnDemand servers through the Internet and through their own intranet. The flexibility that is provided by the ODWEK Java APIs allows for the design of customized interfaces that can meet the specific requirements of an organization.

From a business perspective, there are generally two different approaches to accessing the data that is stored in Content Manager OnDemand: Internet access and intranet access.

In the Internet case, users (people that are external to the organization) are given access to a specific subset of information. For example, in a banking application, an internet user can log on and view a current statement or statements that cover the last 12 months. In this situation, the user has access only to the Content Manager OnDemand archive, and the user is presented with a limited set of menus to obtain this data. Typical internet usage involves tens of thousands of users that access the Content Manager OnDemand archive concurrently. The Content Manager OnDemand architecture helps you scale your system to the limits of the available hardware and network resources, and enables your businesses to grow your system as your usage increases.

In the intranet case, users (people that work for an organization) are given access to a wide variety of data based on their user ID access privileges in the Content Manager OnDemand archive. The user is presented with a selection criteria to access any of the stored data based on the security profile. The flexibility that is provided by the ODWEK Java APIs allows for the design of customized interfaces that can meet the specific requirements of an organization.

A common customer use case

In a common customer use case, the user interacts with IBM Content Navigator or a user-written application, which interacts with Content Manager OnDemand to authenticate the user, search for documents, and retrieve documents.

A common use case across many different business segments involves allowing registered customers to select a range of documents to view. For the banking industry, these documents can be bank statements that can be selected from a predefined number of months. For health insurance companies, the documents can be explanation of benefit statements. For utility companies, the documents can be bills or invoices.

The customer is usually a user who is registered to the company website. The registration process provides credentials, for example, a user ID and password. To gain access to any of the company applications, the user submits the credentials from a web page, such as a portal, to be authenticated by the company web application. After the user is authenticated, the application can restrict the user's access when the user navigates the website.

Users that need access to Content Manager OnDemand must have a user ID. Administrators cannot create a user ID for each registered Internet user. The user population might reach thousands or millions. Therefore, most applications search on behalf of a user ID that has the permissions to search a specific folder and retrieve a document. The assumption is that if the user was not authenticated, and the constraints are placed on the search values to guarantee unique results, then one or more user IDs can be defined to have full permissions to the folder that is searched.

Regardless of the industry that has Content Manager OnDemand statements to present to customers, a customer typically runs two kinds of transactions:

- One transaction involves displaying a search results list of documents, usually within some predefined date range. The application searches on behalf of the user. After the user is authenticated, the application chooses a unique key or key combination to ensure that the search results are for that user, and searches without additional input from the user.

- Another transaction involves the user selecting a document from the search results list. Content Manager OnDemand considers the data type of the document before displaying it to the user. For example, if the document is stored in Content Manager OnDemand as an AFP data stream, one of the following situations must occur:
  - The user needs an AFP viewer program that is locally installed on the user's workstation
  - Data conversion needs to take place to provide a data stream that can be displayed to the user

  In many cases, the application transforms the data (for example, AFP to PDF) to make it easier for the user to view. The PDF viewer is easily obtainable if it is not already installed on the user's workstation. A plug-in (for example, the AFP viewer) might be more difficult for a user to obtain and for a company that provides web service to manage.

In this common customer use case, the following list describes the interaction between the user, the web application, and Content Manager OnDemand:

1. The user signs on to the website with their credentials. The web application authenticates the user, and retrieves a Content Manager OnDemand connection from a pool of connections.
2. The web application selects a predefined folder to search.
3. The web application assigns one or more Content Manager OnDemand folder field values to run a search that is unique to the user.
4. The web application searches, for example, by account number and date range, and sends a search result list to the user.
5. The web application closes the folder and releases the connection back to the pool of connections.
6. The user selects a document to view from the search results list. Optional: The user is authenticated again before the user is allowed to retrieve the document.
7. The web application retrieves a Content Manager OnDemand connection from a pool of connections.
8. The web application retrieves the document and optionally transforms data before it sends the data to the user.
9. The web application releases the Content Manager OnDemand connection back to the pool of connections.

IBM Content Manager OnDemand Web Enablement Kit functions

ODWEK provides an interface to implement functions to do tasks like change a user's password, retrieve a document, or delete an annotation. You can implement these functions and many others using the ODWEK Java API.

The following functions are a sample of the Content Manager OnDemand functions supported by ODWEK.

Add Annotation
The Add Annotation function enables users to add an annotation to the specified document. To add an annotation, the user must be given the Add permission under Annotation in the Content Manager OnDemand application group. (A user is automatically given the Add permission when they are given permission to access an application group.)

Change Password
The Change Password function allows users to change their Content Manager OnDemand passwords.

Delete Annotations
The Delete Annotations function enables users to delete the annotations attached to a specified document.

Logoff
The Logoff function allows users to log off of a Content Manager OnDemand server.

Logon
The Logon function allows the users to log on to a Content Manager OnDemand server.

Retrieve
The Retrieve function retrieves a document from Content Manager OnDemand. The data returned from the server includes the document, and depending on the data type, the resources required to view the document. The returned data must not be modified in any way. The browser, along with the viewer, interprets and decodes the data stream and displays the document. If the document is stored in Content Manager OnDemand as a large object, then only the first segment of the document is returned. Subsequent segments of the document are retrieved and displayed as needed.

Search
The Search function returns a collection of items that match the search criteria. Each item that matches the search is returned as an ODHit object that can be used to call the Retrieve function.

Search Criteria
The user can accept the default search criteria or enter search criteria to search for specific documents.

Server Print Document
The Server Print Document function sends copies of documents to a Content Manager OnDemand server printer. To use server print, the user must be given the Print permission under Document in the Content Manager OnDemand application group. (A user is automatically given the Print permission when they are given permission to access an application group.) At least one server printer must be defined on the Content Manager OnDemand server.

Update Document
The Update Document function allows users to update the database. The Update Document function updates one or more database fields for a specific document. To update a document, the user must be given the Update permission under Document in the Content Manager OnDemand application group.

View Annotations
The View Annotations function enables users to view the annotations attached to the specified document. To view annotations, the user must be given the View permission under Annotation in the Content Manager OnDemand each application group. (A user is automatically given the View permission when they are given permission to access an application group.)

Additional resources

ibm.com offers a variety of articles to help you use ODWEK.

Related information

Best practices for building Web Applications using IBM Content Manager OnDemand Web Enablement Kit (URL: http://www.ibm.com/support/docview.wss?uid=tss1wp101203)

Best practices for AFP Resource caching in a ODWEK Java API application (URL: http://www.ibm.com/support/docview.wss?uid=tss1wp101247)

Debugging a Custom ODWEK Java API Web Application (URL: http://www.ibm.com/support/docview.wss?uid=tss1wp101248)

Chapter 2. Planning your ODWEK solution

Before you install ODWEK, verify the prerequisites, review the descriptions of the programming interfaces and viewers, and learn about the security features.

Prerequisites

Before you install ODWEK, make sure that your system meets the hardware and software requirements. Review the prerequisite information described in the documents available on ibm.com.

Related information

Hardware and software requirements for IBM Content Manager OnDemand (URL: http://www.ibm.com/support/docview.wss?uid=swg27049168)

Hardware and software requirements for IBM Content Manager OnDemand (URL: http://www.ibm.com/support/docview.wss?uid=swg27049168)

Hardware and software requirements for IBM Content Manager OnDemand (URL: http://www.ibm.com/support/docview.wss?uid=swg27049168)

About the programming interfaces

You can have one or multiple instances of ODWEK that connect to one or more Content Manager OnDemand servers.

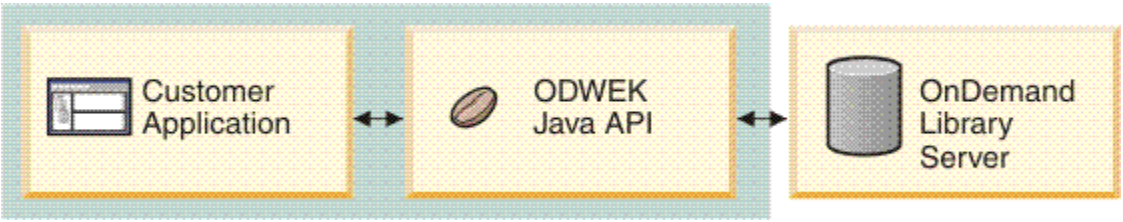
An *instance* of ODWEK is defined by the following features:

- Accesses the Content Manager OnDemand server through the ODWEK Java API, either through IBM Content Navigator or a user-written application.
- Communicates with the Content Manager OnDemand server with its own TCP/IP port.

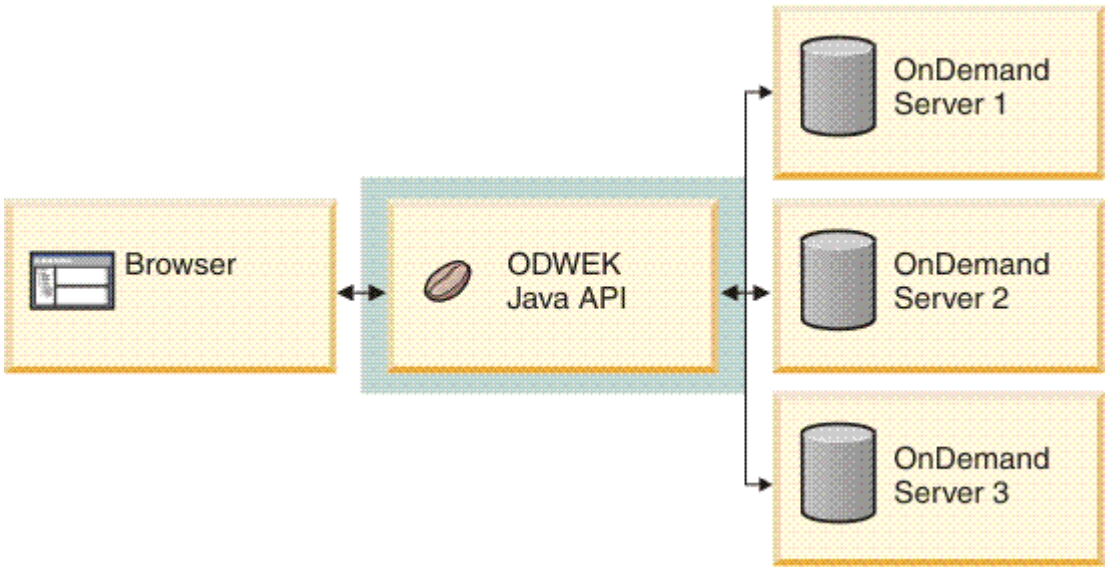
An instance controls what can be done to the data, and manages the system resources that are assigned to it. You can run multiple instances of ODWEK on a single system; however, the instances cannot communicate with each other.

The most common implementation of ODWEK is a single instance on a system. This implementation is typically for developers or stand-alone production computing, which involve a single application server instance that operates independently of any other applications.

The following figure illustrates an example of a customer application connecting to a Content Manager OnDemand server through the Java API interface.



The following figure illustrates an example of how a single instance (implemented with the Java API interface) connects to three different Content Manager OnDemand servers. Each Content Manager OnDemand server communicates through its own port. Your ODWEK application must manage the communication with each Content Manager OnDemand servers: When you write your ODWEK application, you must ensure that you send the correct information to the correct Content Manager OnDemand server.



You can implement multiple instances of ODWEK on the same system or on different systems. Each instance requires its own storage and security resources. The instances can communicate with a single Content Manager OnDemand server or multiple Content Manager OnDemand servers. The instances operate independently of each other. You might want to implement multiple instances for any of the following reasons:

- You might want to develop and run your applications in different environments (for example, development, testing and production) on the same system.
- You might want different groups of client applications to connect through different ODWEK instances for security, workload distribution, or other business reasons.

About the viewers

The viewers help you display documents that are saved in a specific format or need special processing before they can be displayed on a user's screen.

ODWEK provides the following viewers:

- AFP Web Viewer
- HTML5 Line Data Viewer
- Java Line Data Viewer

The AFP Web Viewer is a software program that extends the capabilities of a Web browser in a specific way. The AFP Web Viewer lets users view AFP documents in the browser window. The AFP Web Viewer adds a toolbar to the top of the display window. The viewer toolbar is in addition to the browser's toolbar. The viewer toolbar provides controls that can help users work with documents. Each user who plans to use the AFP Web Viewer to view AFP documents must install it on their PC.

The HTML5 Line Data Viewer is an HTML5-based line data viewer with similar functionality to the Java Line Data Viewer, but it does not require the Java Plugin in order to extend the browser functionality. Most modern browsers have already phased out support for the Java Plugin. The HTML5-based line viewer is intended as a replacement for the Java Line Data Viewer.

The Java Line Data Viewer is an applet that requires the Java Plugin. The Java Line Data Viewer allows users to view line data documents that are stored in Content Manager OnDemand. You can continue to use the Java Line Data Viewer as long as your browser meets the requirements to run applets.

The Line Data Viewers display line data documents in the browser window and add a toolbar to the top of the display window. The toolbars of the Line Data Viewers provide controls that can help users work with documents. An administrator enables the use of the Line Data Viewers programmatically by writing their

application to use one of the Line Data Viewers when retrieving line data documents. See Chapter 6, “Installing and configuring viewing and transform software,” on page 85 for more details.

One advantage of both of the Line Data Viewers is that users never have to install or upgrade software on the PC to use them, unlike the AFP Web Viewer, which must be installed on the PC. Also, if IBM provides a new version of the AFP Web Viewer, you must distribute the updated software to all users.

When using the viewers that are provided by IBM, the documents that are retrieved from a Content Manager OnDemand server remain compressed until reaching the viewer. The viewer uncompresses the documents and displays the pages in a Web browser window. If a document was stored in Content Manager OnDemand as a large object, then the viewer retrieves and uncompresses segments of the document, as needed, when the user moves through pages of the document.

Server and data security

You can protect your company’s data by controlling access to data and ODWEK programs and the web pages.

There are two levels of security that you need to consider before you use ODWEK:

- Who can access the Content Manager OnDemand Java API
- Who can access data on the Content Manager OnDemand server

Any user that can access your web server and the programs and web pages that interface with ODWEK can potentially access data stored in Content Manager OnDemand. Limit access to the programs and web pages to help protect the data stored in Content Manager OnDemand. There are many ways that you can limit access to programs and web pages on your web server. For example, many web servers provide a system of security to sensitive web pages to help you restrict access to directories. You can also use a password file on the web server, which requires users to enter a user ID and password before accessing the web pages. However, even though web server user IDs and passwords are similar to operating system user IDs and passwords, there is no correspondence between them. There is also no correspondence between web server user IDs and passwords and Content Manager OnDemand user IDs and passwords.

ODWEK provides access to Content Manager OnDemand servers and data. The APIs verify that the Content Manager OnDemand user ID can access the server and the requested data. Someone in your organization must administer user and data security on the Content Manager OnDemand server.

Chapter 3. Configuring IBM Content Manager OnDemand Web Enablement Kit

Configuring IBM Content Manager OnDemand Web Enablement Kit is the first piece you install of all the pieces required to run ODWEK applications. When you install Content Manager OnDemand for Multiplatforms or Content Manager OnDemand for z/OS, you also install ODWEK; you do not need to run a separate installation procedure to install ODWEK. The instructions for installing ODWEK for Content Manager OnDemand for i are described in "IBM Content Manager OnDemand for i: Common Server Planning and Installation Guide".

About this task

You must install ODWEK for it to work properly. If you copy the ODWEK files from physical or electronic media, then configure ODWEK, it will not work.

Installing documentation for Java APIs

The documentation for the Java APIs is stored in a compressed file that you can copy to any directory or file system. Then, you extract the files, which are HTML files, and view them with any web browser.

Procedure

To install the documentation for the Java APIs, do the following steps:

1. Copy (by using a copy command or FTP) the `ODApiDoc.zip` file from the following locations.

Option	Description
Multiplatforms	`x:\install_dir\www\api` Where *x* is the drive on which you installed Content Manager OnDemand and *install_dir* is the installation directory for the Content Manager OnDemand software.
z/OS	`install_dir/www/api` Where *install_dir* is the installation directory for the Content Manager OnDemand software.
i	`/QIBM/ProdData/OnDemand/www/api` This path is located in the Integrated File System (IFS).

2. Extract the documentation from the `ODApiDoc.zip` file.

Results

To view the documentation, open the `index.html` file with a web browser.

Installing the Java API components

To develop Java API programs that work with Content Manager OnDemand, you must set up your development environment by providing ODWEK with access to the files or objects.

Related concepts

Client/server architecture

The APIs provide a convenient programming interface for application users. APIs can be stored on both the Content Manager OnDemand server and the client (both provide the same interface), and the applications can be located locally or remotely. The client API communicates with the server to access

data through the network. Communication between the client and the server is performed by classes; it is not necessary to add any additional programs.

Related tasks

Setting up the development environment
To set up the ODWEK application development environment on your system, you must make sure that ODWEK has access to library files, shared objects, and DLLs.

Installing documentation for Java APIs
The documentation for the Java APIs is stored in a compressed file that you can copy to any directory or file system. Then, you extract the files, which are HTML files, and view them with any web browser.

Related reference

Packaging for the Java environment
The API classes help facilitate communication with Content Manager OnDemand and instantiate programming models of Content Manager OnDemand objects (for example, application groups and folders).

Chapter 4. Developing Java applications

You can develop your own applications to access Content Manager OnDemand with the ODWEK Java APIs.

IBM i z/OS Multiplatforms Client/server architecture

The APIs provide a convenient programming interface for application users. APIs can be stored on both the Content Manager OnDemand server and the client (both provide the same interface), and the applications can be located locally or remotely. The client API communicates with the server to access data through the network. Communication between the client and the server is performed by classes; it is not necessary to add any additional programs.

Important: The ODServer object represents a single connection to a Content Manager OnDemand server. This object, and all objects created from it (ODFolder, ODHit, and so on), cannot be accessed by multiple threads at the same time. To handle large numbers of users, it is recommended to either create a single ODServer object per user or create an ODServer connection pool where these connections can be distributed to a single transaction at a time.

Related tasks

Cancelling a search
The cancel method in the ODServer class cancels a search in progress.

IBM i z/OS Multiplatforms Packaging for the Java environment

The API classes help facilitate communication with Content Manager OnDemand and instantiate programming models of Content Manager OnDemand objects (for example, application groups and folders).

The API classes are contained in one package: `com.ibm.edms.od`. The following list describes the classes:

com.ibm.edms.od.ODApplication
This class represents a Content Manager OnDemand application. An instance of the ODApplication object provides an application developer access to information that is specified for a Content Manager OnDemand application.

com.ibm.edms.od. ODApplicationGroup
This class represents a Content Manager OnDemand application group. An instance of the ODApplicationGroup object provides an application developer access to information that is specified for a Content Manager OnDemand application group. An instance of the ODApplicationGroup object is generated through an instance of the ODServer object by calling the `getApplicationGroup()` method.

com.ibm.edms.od. ODApplicationGroupField
This class represents a Content Manager OnDemand application group field. It contains application group field information.

com.ibm.edms.od.ODCabinet
This class represents a collection of Content Manager OnDemand folders. The administrator defines and manages cabinets on the Content Manager OnDemand server.

com.ibm.edms.od.ODCallback
This class is used with all methods in which the server operation returns data while processing.

com.ibm.edms.od.ODConfig
The ODConfig Java object is the preferred method to configure the system parameters.

com.ibm.edms.od.ODConstant
ODConstant is the public interface.

com.ibm.edms.od.ODCriteria
A class that represents the search criteria from a Content Manager OnDemand folder. The criteria class contains methods to set a search operator and search values.

com.ibm.edms.od.ODException
This class represents the exceptions which might occur when using the APIs.

com.ibm.edms.od.ODFolder
A class that represents a Content Manager OnDemand folder. This object is returned from a successful call to `ODServer.openFolder()`. This class contains folder criteria information. These criteria objects are what need to be modified in order to narrow the query on the server.

com.ibm.edms.od.ODHit
This class represents a Content Manager OnDemand document.

com.ibm.edms.od. ODHitProperties
Use this class to obtain the Content Manager OnDemand internal property values for a hit.

com.ibm.edms.od.ODHold
This class represents a Content Manager OnDemand hold definition. This object is returned from a successful call to (`ODServer.getHolds()`).

com.ibm.edms.od.ODLogicalView
This class represents a Content Manager OnDemand logical view.

com.ibm.edms.od. ODNamedQuery
This class represents a Content Manager OnDemand named query. This class contains the details of a named query, and enables the system to retrieve existing named queries and save new named queries to the Content Manager OnDemand server.

com.ibm.edms.od. ODNamedQueryCriteria
This class represents the criteria of a Content Manager OnDemand named query. This class contains search criteria details that are stored in a named query.

com.ibm.edms.od.ODNote
This class represents a Content Manager OnDemand annotation.

com.ibm.edms.od.ODServer
This class represents a connection to a Content Manager OnDemand server. From this class you can log on, log off and change the password. After a successful logon, this object will contain a list of all folders that the session has access to.

com.ibm.edms.od.ODTransform
This class represents the configuration settings for a single Generic transform to be defined to the ODWEK Java APIs.

com.ibm.edms.od.ODUser
This class represents a Content Manager OnDemand user. From this class, you can gather user information such as address and phone number that is stored in the Content Manager OnDemand server.

Related tasks
Configuring Content Manager OnDemand instance parameters
Configure the Content Manager OnDemand instance parameters by using the ODConfig Java object.

IBM i z/OS Multiplatforms Programming tips

You must import the `com.ibm.edms.od` package into your ODWEK application.

You do not need an HTTP server or a web application server to run ODWEK applications that use the Java API. You can run the Java interpreter on ODWEK applications.

To run the Java interpreter on an ODWEK application:

1. z/OS Multiplatforms Copy the shared libraries to the runtime directory. If you do not want to copy the shared library to the runtime directory, you can copy the files to the `Java.Library.Path`.

Table 1. Shared library file names	
Operating System	**Shared Libraries**
AIX®	`libars3wapi32.a` `libars3wapi64.a`
Linux	`libars3wapi32.so` `libars3wapi64.so`

Windows	`ars3wapi32.dll`
	`ars3wapi64.dll`
	`icudt65.dll`
	`icuin65.dll`
	`icuio65.dll`
	`icule65.dll`
	`iculx65.dll`
	`icuuc65.dll`
	`arsgsk64.dll` or `arsgsk32.dll`

z/OS	`libars3wapi32.so` `libars3wapi64.so`

IBM i The Java API shared library (service program) is ARS3WAPI64 and is found in the QRDARS library.

2. Run the Java interpreter on your application by entering a command similar to the following example: `java Logon server userid passwd`.

   IBM i You must issue the command in Qshell or by entering the Run Java Program (JAVA) command, with the appropriate parameters, from the command line.

Setting up the development environment

To set up the ODWEK application development environment on your system, you must make sure that ODWEK has access to library files, shared objects, and DLLs.

About this task

ODWEK must have access to the following files or objects:

- Library files
- For AIX, Linux, and z/OS: shared objects
- For Windows: DLLs
- For IBM i: Service programs

Procedure

To provide ODWEK with access to these files or objects, do one of the following tasks:

- Manually set the **LIBPATH**, **LD_LIBRARY_PATH**, or **PATH** environment variables.
- Use the Java Library Path parameter in the JVM to specify the library path.

The following table describes which environment variables to set for your operating system. For AIX and Linux, the values listed in the table assume that you installed ODWEK in the default installation directory. If you specified a different installation directory, substitute `/opt/IBM/ondemand/V10.5` with the directory that you specified. For IBM i, see additional information following the table.

Table 2. Environment variables to set for each operating system supported by ODWEK.

Operating system	Environment variable	Value to specify or task to complete
AIX	**LIBPATH**	`/opt/IBM/ondemand/V10.5/www`
AIX	**CLASSPATH**	`/opt/IBM/ondemand/V10.5/www/api/ODApi.jar:/opt/IBM/ondemand/V10.5/jars/gson-2.8.6.jar:/opt/IBM/ondemand/V10.5/jars/log4j-api-2.13.0.jar:opt/IBM/ondemand/V10.5/jars/log4j-core-2.13.0.jar`
IBM i	**LIBPATH**	`/QSYS.LIB/QRDARS.LIB`
IBM i	**CLASSPATH**	`/QIBM/ProdData/OnDemand/www/api/ODApi.jar`
Linux	**LD_LIBRARY_PATH**	`/opt/ibm/ondemand/V10.5/www`
Linux	**CLASSPATH**	`/opt/ibm/ondemand/V10.5/www/api/ODApi.jar:/opt/ibm/ondemand/V10.5/jars/gson-2.8.6.jar:/opt/ibm/ondemand/V10.5/jars/log4j-api-2.13.0.jar:opt/ibm/ondemand/V10.5/jars/log4j-core-2.13.0.jar`
Windows	**PATH**	`x:\install_dir\www;x:\install_dir\bin` (or `x:\install_dir\bin32`) where *x* is the drive on which you installed Content Manager OnDemand and *install_dir* is the installation directory for the Content Manager OnDemand software.
Windows	**CLASSPATH**	`x:\install_dir\www\api\ODApi.jar;x:\install_dir\jars\gson-2.8.6.jar;x:\install_dir\jars\log4j-api-2.13.0.jar;x:\install_dir\jars\log4j-core-2.13.0.jar` where *x* is the drive on which you installed Content Manager OnDemand and *install_dir* is the installation directory for the Content Manager OnDemand software.
z/OS	**Libpath**, **Classpath**	To configure the `libarswapi64.so` shared library and the CLASSPATH property, you must deploy an EAR file.

For IBM i LIBPATH:

When running Java in QSHELL, this is specified via the **LIBPATH** environment variable with the following export command:

```
export LIBPATH=/QSYS.LIB/QRDARS.LIB
```

When running Java by calling the JAVA CL command, this is specified on the PROP parameter. For example:

```
JAVA PROP((java.library.path '/QSYS.LIB/QRDARS.LIB')) …
```

For IBM i CLASSPATH:

When running Java in QSHELL, the **CLASSPATH** value can be specified with the following export command:

```
export CLASSPATH=/QIBM/ProdData/OnDemand/www/api/ODApi.jar:your__path
```

or on the java –classpath or -cp option. For example:

```
java –cp /QIBM/ProdData/OnDemand/www/api/ODApi.jar:your_path …
```

For the JAVA CL command, it is specified directly on the **CLASSPATH** parameter. For example:

```
JAVA CLASSPATH('/QIBM/ProdData/OnDemand/www/api/ODApi.jar:your_path') …
```

or by using an environment variable. For example:

```
ADDENVVAR ENVVAR(CLASSPATH) VALUE('/QIBM/ProdData/OnDemand/www/api/ODApi.jar:your_path')
JAVA CLASSPATH(*ENVVAR) …
```

If you store your Java code in the `/ODWEKJAVA` directory in the IFS (Integrated File System), substitute *your_path* with `/ODWEKJAVA`.

ODWEK is dependent on some external Java libraries. On IBM i, symbolic links to these libraries are added in a directory named `/QIBM/ProdData/OnDemand/www/api` so that they do not need to be explicitly specified in the **CLASSPATH**. If you move the `ODApi.jar` file out of its shipped location in the `/QIBM/ProdData/OnDemand/www/api` directory, you will need to add the following jars to your **CLASSPATH**:

- `/QIBM/ProdData/OnDemand/jars/gson-2.8.6.jar`
- `/QIBM/ProdData/OnDemand/jars/log4japi-2.13.0.jar`
- `/QIBM/ProdData/OnDemand/jars/log4jcore-2.13.0.jar`

Running an ODWEK application

To run an ODWEK application, you need to import the Java API package into your ODWEK application, compile your application, and then run it.

About this task

Before you begin developing and running an ODWEK application, ensure that you set up the development environment.

If you are not familiar with the classes and methods available to you, see the Javadoc.

Procedure

To develop and run an ODWEK application, the following list describes the general steps:

1. Create your ODWEK application by using the methods that are available to you in the Java API. Import the Java API package in your ODWEK application file.

   The following example shows sample code with the line that imports the Java API package highlighted.

```
//*********************************************************
import java.util.*;
import java.io.*;
 import com.ibm.edms.od.*;

public class Logon
{
  public static void main ( String argv[] )
  {
    .
    .
    .
  }
}
```

2. Compile your ODWEK application file (`.java`) with **javac** to produce the `.class` file. For instructions on compiling Java applications, see your Java reference publication.

3. Run the Java interpreter on your application (`.class` file).

   The following example shows how you might run the Java interpreter on an application called Logon:`java Logon server userid passwd 1450`. In this example, `Logon` is the name of the `.class` file; `server`, `userid`, `passwd`, and `1450` are parameters for the application.

Related tasks

Setting up the development environment

To set up the ODWEK application development environment on your system, you must make sure that ODWEK has access to library files, shared objects, and DLLs.

Code page conversion in ODWEK

ODWEK internally runs in UTF-8, which leads to a conversion of all index and annotation data from UTF-16 (the format in which the data are sent in TCP/IP) to UTF-8. Because ODWEK is a mid-tier system, you must consider the technology used to implement the presentation layer, whether that technology can correctly display the data, and, if necessary, convert the code page of the data.

In contrast to the standard Content Manager OnDemand client, ODWEK behaves differently when it converts code pages. Because ODWEK is a mid-tier system, there is always an additional presentation layer. In most cases, the layer is a browser; however, it can also be a stand-alone Java application that uses the ODWEK API.

When you implement a Java application that uses ODWEK, ensure that you correctly handle the information that you pass to and retrieve from ODWEK API. Because Java works in UTF-16 Unicode internally, you can ignore data that are returned from ODWEK API functions. Java handles the conversion from UTF-8. When you pass strings to ODWEK methods, you do not need to perform any additional tasks because Java supports only string variables that are in UTF-16. The conversion to UTF-8 is done by ODWEK subroutines.

Despite the internal conversion to UTF-8, ODWEK does not do any other conversion on indexes. Therefore, a client that displays index data that is received through the ODWEK API must correctly handle UTF-8 Unicode data. If you deliver index data to any external applications by using your ODWEK-based Java application, you must ensure that those applications can handle Unicode data or you must convert the data manually. The same implications apply if you want to save any indexes or annotation data to file. If you do not perform any explicit conversion, the data is written as a Unicode data stream. As web browsers usually can display and send UTF-8 data, it is not a problem when you implement web applications.

For the document data, you can handle the conversion in different ways. If you request raw native document data, ODWEK returns the data in its unaltered form: In the same code page in which it was archived. When you request the line data to be displayed by using an applet, ODWEK sends the UTF-8 ASCII data to the applet, but only standard HTML code containing applet invocation code is returned. When you request an ASCII conversion, ODWEK returns a UTF-8 ASCII converted representation of the original AFP or line data document. For most other document types, ODWEK works the same as the Content Manager OnDemand Windows client: it passes the data in the format that is native to the data format.

IBM i z/OS Multiplatforms Tracing and diagnostic information

ODWEK helps you handle exceptions you might encounter in your Java API applications by providing the ODException class. If you encounter problems that require assistance from IBM support, IBM support might direct you to enable ODWEK tracing.

Enabling ODWEK tracing

ODWEK tracing writes trace statements to the binary file `arswww.trace`. ODWEK tracing assists IBM support in problem determination; however, as with any form of tracing, you might notice a decline in

performance when you enable ODWEK tracing. Enable ODWEK tracing only when you need to re-create a problem and as directed by IBM support.

About this task

By default, ODWEK appends information to the trace file until you disable tracing; it does not cycle through the file. You must periodically delete old trace files. If you need to set a maximum size that your trace file can grow to, you must set a value using the ODConfig.MAX_TRACELOG_SIZE field when you create your ODConfig object. For more information, see the Javadoc.

Procedure

To enable the ODWEK tracing, do the following steps:

1. Add code to your application that indicates where to save the trace information.

   The following code is an example:

```
ODConfig cfg = new ODConfig(
/*AfpViewer*/           ODConstant.PLUGIN,
/*LineViewer*/          ODConstant.APPLET,
/*MetaViewer default*/  null,
/*MaxHits*/             500,
/*AppletDir*/           "/applets",
/*Language*/            "ENU",
/*TempDir*/             "c:\\temp",
/*TraceDir*/            "trace_directory",
/*TraceLevel*/          trace_level);
```

   The following list describes what values to provide for the *TraceDir* and *TraceLevel*:

   TraceDir
   Specify the full path of the directory where you want ODWEK to store the trace file. If you need to enable logging for extended periods of time, make sure that the directory is on a storage device with sufficient free space.

   TraceLevel
   Specify a numeric value (0, 1, 2, 3, or 4), as directed by IBM Software Support.

2. Re-compile and restart your ODWEK application.
3. Re-create the situation that created the problem.
4. Send the `arswww.trace` file to IBM Software Support.

Related tasks

Installing documentation for Java APIs
The documentation for the Java APIs is stored in a compressed file that you can copy to any directory or file system. Then, you extract the files, which are HTML files, and view them with any web browser.

IBM i z/OS Multiplatforms Exception handling

When the Java APIs encounter a problem, they throw an exception. Throwing an exception creates an exception object of ODException class or one of its subclasses.

When an exception object of ODException class is created, the API saves diagnostic information in to a log file, assuming that logging is enabled.

When a ODException is caught, it helps you to see any error messages, error codes, and error states that occurred while running. When an error is caught, an error message is issued along with the location of where the exception was thrown. The error ID and exception ID are also given. The following code shows an example of the throw and catch process:

```
try
 {
   odServer = new ODServer(new ODConfig());
   odServer.initialize( "TcUpdate.java" );
   System.out.println( "Logging on to " + argv[0] + "..." );
   odServer.logon( argv[0], argv[1], argv[2] );
   odServer.logoff( );
   odServer.terminate( );
```

```
    }

   catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }
```

Related tasks

Enabling ODWEK tracing

ODWEK tracing writes trace statements to the binary file `arswww.trace`. ODWEK tracing assists IBM support in problem determination; however, as with any form of tracing, you might notice a decline in performance when you enable ODWEK tracing. Enable ODWEK tracing only when you need to re-create a problem and as directed by IBM support.

IBM i z/OS Multiplatforms Vector of hits not maintained by the Content Manager OnDemand API

When you use some of the Content Manager OnDemand Java APIs, the vector of hits are not maintained between retrieve calls.

While using the Java APIs to do the following actions, the vector of hits is not maintained by the Content Manager OnDemand API from one retrieve call to another retrieve call:

- Searching for documents
- Listing search criteria
- Displaying documents
- Printing documents
- Updating documents

For example, when you use the ODFolder.printDocs API to do a print action, if you use the hits retrieved from a previous module, you might encounter this error:

```
ODException errorcode: 2170
ODException errormsg: No hits specified
```

Content Manager OnDemand server connections

An object of the class ODServer represents and manages a connection to a Content Manager OnDemand server, provides transaction support, and runs server commands. The online reference that describes the ODServer class and its methods is in Javadoc format.

When you connect to a Content Manager OnDemand server, remember to comply with any requirements or restrictions implemented by your site. For example, your site might configure the Content Manager OnDemand server to not accept passwords longer than eight characters in length.

Related tasks

Installing documentation for Java APIs

The documentation for the Java APIs is stored in a compressed file that you can copy to any directory or file system. Then, you extract the files, which are HTML files, and view them with any web browser.

Establishing a connection

The ODServer class provides methods for connecting to a Content Manager OnDemand server and disconnecting from the server.

Procedure

To establish a connection to a Content Manager OnDemand server, program the following tasks in your ODWEK application:

1. Instantiate an ODConfig object with correct property values.
2. Instantiate an ODServer object with the ODConfig object.
3. Initialize the ODServer object.
4. Perform a logon with a valid OnDemand ID.

Example

The following example uses a Content Manager OnDemand library server named LIBSRVR1, the user ID ADMIN, password PASSWD, and port number 1450. The example creates an ODServer object for the Content Manager OnDemand server, connects to it, works with it (not specified in the example), and then disconnects from it.

```
odServer = new ODServer( new ODConfig() );
int port = Integer.parseInt('1450');
System.out.println( "Logging on to " + "LIBSRVR1" + "..." );
odServer.setPort(port);
odServer.logon( "LIBSRVR1", "ADMIN", "PASSWD" );
  .
  .
  .
odServer.logoff( );
odServer.terminate( );
```

Related reference

Working with a Content Manager OnDemand server
An object of the ODServer class represents and manages a connection to a Content Manager OnDemand server, provides transaction support, and runs server commands.

Passwords

You can access or set a user's password on a Content Manager OnDemand server by using the methods in ODServer.

The following example shows how to set and get a user's password on a Content Manager OnDemand library server.

```
odServer = new ODServer( new ODConfig() );
odServer.initialize( "MyCustomApp" );
odServer.setServerName( "LIBSRVR1" );
odServer.setUserId( "ADMIN" );
odServer.setPassword( "PASSWD" );

System.out.println( "Logging on to " + "LIBSRVR1" + "..." );
int port = Integer.parseInt('1450');

odServer.setPort(port);
odServer.logon( odServer.getServerName( ),
                odServer.getUserId( ),
                odServer.getPassword( ) );
```

While using the odServer.logon() method, the application might fail with the following message:

```
Connection cannot be established for the server_name server [id = 2086].
```

That means that the Content Manager OnDemand server name provided in the odServer.logon() method is not a valid server name. To fix this problem, ensure that the ODServer.logon() call contains a valid host name or the IP address of the Content Manager OnDemand server.

Related reference

Working with a Content Manager OnDemand server

An object of the ODServer class represents and manages a connection to a Content Manager OnDemand server, provides transaction support, and runs server commands.

Connecting to a non-default port

If you need to connect to the non-default port of the Content Manager OnDemand server, you use the `ODServer.setPort()` API call immediately before the logon in your ODWEK application.

About this task

In some situations, you might need to access a non-default port. For example, you might have two instances of the Content Manager OnDemand server. One instance uses the default port, and another uses a different port. Unless you configure your system properly, when you run your Java program, you see the following error: "A connection cannot be established to the instance2 server".

To resolve this situation, use the `ODServer.setPort()` API call immediately before the logon within your ODWEK application.

Configuring Content Manager OnDemand instance parameters

Configure the Content Manager OnDemand instance parameters by using the ODConfig Java object.

About this task

You must set several Content Manager OnDemand instance parameters to create a working ODServer instance. To create an ODServer instance with the default values for those parameters, add code that calls the ODConfig constructor similar to the following example:

```
try{
     ODConfig cfg = new ODConfig();
     ODServer srvr = new ODServer(cfg);
     srvr.initialize("MyCustomApp");
     cfg.printConfig();
 }
 catch(ODException e){
     System.out.println("Exception " + e);
 }
```

This following list describes the default values for the Content Manager OnDemand instance parameters:

AfpViewOpt
PLUGIN

LineViewOpt
APPLET

MaxHits
200

MetaViewOpt
NATIVE

AppletDir
/applets

Language
ENU

TempDir
The temp path as defined by the Java System.getProperty("java.io.tmpdir") method.

TraceDir
The temp path as defined by the Java System.getProperty("java.io.tmpdir") method.

TraceLevel
0

To create an ODServer instance with values other than the default values, add code that calls the ODConfig constructor similar to the following example:

```
try{
     ODConfig cfg = new ODConfig(ODConstant.PLUGIN, //AfpViewer
                                 ODConstant.APPLET, //LineViewer
                                 null,              //MetaViewer
                                 500,               //MaxHits
                                 "c:\\applets",     //AppletDir
                                 "ENU",             //Language
                                 "c:\\temp",        //TempDir
                                 "c:\\temp\\trace", //TraceDir
                                 1);                //TraceLevel

    ODServer srvr = new ODServer(cfg);
    srvr.initialize("MyCustomApp");
    cfg.printConfig();
 }
 catch(ODException e){
     System.out.println("Exception " + e);
 }
```

Important: This object has no methods to set parameters except during construction. You cannot modify the object after constructing it.

Related tasks

Installing documentation for Java APIs
The documentation for the Java APIs is stored in a compressed file that you can copy to any directory or file system. Then, you extract the files, which are HTML files, and view them with any web browser.

Working with a Content Manager OnDemand server

An object of the ODServer class represents and manages a connection to a Content Manager OnDemand server, provides transaction support, and runs server commands.

The following example uses ODServer methods to do the following tasks:

- Prepare for logon
- Set the application name
- Display the server name, user ID and password
- Display and set the connection type
- Display and set the port
- Disconnect from the server

This example demonstrates the following ODServer methods:

- initialize
- logon
- logoff
- terminate
- getConnectType
- getPassword
- getPort
- getServerName
- getUserId
- setConnectType
- setPassword
- setPort
- setServerName

- setUserId

This example uses the following runtime parameters:

- Server name
- User Id
- Password
- Port

Example of working with a Content Manager OnDemand server:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;


public class TcServerMisc
{
    public static void main ( String argv[] )
    {
        ODServer odServer;

        String str;

        int j;
        //----------
        // If too few parameters, display syntax and get out
        //----------
        if ( argv.length < 4 )
        {
            System.out.println( "usage: java TcServerMisc <server> <port> <userid>
<password>" );
            return;
        }
        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "Testcase TcServerMisc started." );
            System.out.println( "This testcase should:" );
            System.out.println( "  Use ODServer methods setServer, setUserId, and
setPassword" );
            System.out.println( "    to prepare for logon" );
            System.out.println( "  Set the application name" );
            System.out.println( "  Display the" );
            System.out.println( "    Server name" );
            System.out.println( "    User Id" );
            System.out.println( "    Password" );
            System.out.println( "  Set and display the port" );
            System.out.println( "  Logoff" );
            System.out.println( "" );
            System.out.println( "Ensure that all information is correct." );
            System.out.println( "" );

System.out.println( "--------------------------------------------------" );
            System.out.println( "" );

            //----------
            // Logon to specified server
            //----------
            ODConfig odConfig = new ODConfig();
            if(odConfig == null)
                return;
            else
            {
                odServer = new ODServer(odConfig );
                odServer.initialize(  "TcServerMisc.java" );
                odServer.setServerName( argv[0] );
                odServer.setUserId( argv[2] );
```

```
                odServer.setPassword( argv[3] );
                odServer.setPort(Integer.parseInt(argv[1]));
                System.out.println( "Logging on to " + argv[0]  + " server with user
" + argv[2] + "..." );
                odServer.logon( );
                //----------
                // Test miscellaneous methods
                //----------
                System.out.println( "Server Name: " + odServer.getServerName( ) );
                System.out.println( "User Id: " + odServer.getUserId( ) );
                System.out.println( "Password: " + odServer.getPassword( ) );
                j = odServer.getPort( );
                System.out.println( "Setting port to " + j + "..." );
                odServer.setPort( j );
                System.out.println( "Port: " + j );
                //----------
                // Cleanup
                //----------
                System.out.println( "Logging off..." );
                odServer.logoff( );
                odServer.terminate( );
                System.out.println( "" );

System.out.println( "--------------------------------------------------" );
                System.out.println( "" );
                System.out.println( "Testcase TcServerMisc completed - analyze if
required" );
                System.out.println( "" );
            }
        }

        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "   id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}
```

Listing application groups in a folder

An object of the class ODFolder represents a Content Manager OnDemand folder.

About this task

The following example uses ODFolder methods to display the number of application groups that can be searched from the folder and display the name of each application group.

This example demonstrates the following ODFolder methods:

- getApplGroupNames
- getNumApplGroups
- close

This example also uses ODServer methods to prepare for logon, open the specified folder, and log off. This example demonstrates the following ODServer methods:

- getApplicationGroup
- initialize

- logoff
- logon
- openFolder
- terminate

This example uses the following runtime parameters:

- Folder name
- Password
- Port
- Server name
- User Id

Example of listing the application groups in a folder:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcApplGrp
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    String[] appls;
    int j;
    long agid=0;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcApplGrp <server> <port> <userid> <password> <folder>" );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "Test case TcApplGrp started." );
      System.out.println( "This test case should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Display the folder name" );
      System.out.println( "  Display the number of application groups" );
      System.out.println( "  Display the name of each application group" );
      System.out.println( "  Get application group by name and display" );
      System.out.println( "  Get application group by application group id and display" );
      System.out.println( "" );
      System.out.println( "-------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to the specified server
      //----------
      ODConfig odConfig = new ODConfig();

      if (odConfig != null)
      {
         odServer = new ODServer(odConfig );
         odServer.initialize(  "TcApplGrp.java" );
               odServer.setPort(Integer.parseInt(argv[1]));
         System.out.println( "Logging on to " + argv[0]  + " server with user " + argv[2] + "..." );
         odServer.logon( argv[0], argv[2], argv[3]);

         //----------
         // Open the specified folder
         //----------
         System.out.println( "Opening " + argv[4] + " folder..." );
         odFolder = odServer.openFolder( argv[4] );
```

```
        //----------
        // Display number and names of application groups
        //----------
        System.out.println( "There is(are) " + odFolder.getNumApplGroups( ) + " application group(s) in
the folder:" );
        Object[] appl_grps = odFolder.getApplGroupNames( );
        String agname =  appl_grps[0].toString();
        for ( j = 0; j < appl_grps.length; j++ )
            System.out.println( "   " + appl_grps[j] );

        System.out.println("\nGet the Application Group by name: " + agname);
        ODApplicationGroup odAG1 = odServer.getApplicationGroup(agname);
        System.out.println("Application Group Name = " + odAG1.getName());
        System.out.println("Application Group Description = " + odAG1.getDescription());
        System.out.println("Application Group ID = " + odAG1.getId());
        System.out.println("Application Group ID Permissions = " + odAG1.getIdPerms());
        System.out.println("isRMEnabled = " + odAG1.isRMEnabled());
        System.out.println("isHoldEnabled = " + odAG1.isHoldEnabled());
        System.out.println("isCFSODEnabled = " + odAG1.isCFSODEnabled());
        System.out.println("Mapped Applications:");
        appls = odAG1.getApplicationNames();
        for (int i = 0; i < appls.length; i++)
            System.out.println("    " +appls[i]);

               System.out.println("Display AGField details");
               Enumeration AGFields = odAG1.getFields();
               //System.out.println(" There are " + AGFields.size() + " fields in the AG");

               while(AGFields.hasMoreElements())
               {
                  ODApplicationGroupField field = (ODApplicationGroupField)AGFields.nextElement();
                  System.out.println(" Name= " + field.getName());
                  System.out.println("   Mask= " + field.getMask());
                  System.out.println("   Type= " + field.getType());
                  System.out.println("   Qual= " + field.getQual());
                  if (field.getType() == ODConstant.OD_FLD_STRING)
                     System.out.println("The field type is STRING");
               }
               agid =  odAG1.getId();
               System.out.println("Application Group ID = " + agid);

               //8410_20_03 enhancement to list all Application Groups for this user/server
               System.out.println("List Application Group Names Based on Criteria");
               String[] agNames = odServer.getApplicationGroupNames("%");
               for(int i = 0; i < agNames.length; i++)
               {
                   System.out.println("  " + agNames[i]);
               }

           //----------
           // Cleanup
           //----------
           odFolder.close( );
           odServer.logoff( );
           odServer.terminate( );
      }
      else
      {
           System.out.println( "" );
           System.out.println( "Test case TcApplGrp failed  -" );
           System.out.println( "  ODConfig could not be initialized.  " );
           System.out.println( "  Probable cause: " );
           System.out.println( "    File arswww.props is not found in directory " +  argv[5] );
           System.out.println( "" );
      }

      System.out.println("Re-Logon and attempt to get the Application Group using Application Group ID
saved above.");
      odConfig = new ODConfig();

      if (odConfig != null)
      {
           odServer = new ODServer(odConfig );
           odServer.initialize(  "TcApplGrp.java" );
           System.out.println( "Logging on to " + argv[0]  + " server with user " + argv[2] + "..." );
           odServer.logon( argv[0], argv[2], argv[3] );
           System.out.println( "Opening " + argv[4] + " folder..." );
           odFolder = odServer.openFolder( argv[4] );

           System.out.println("\nGet the Application Group for id " + agid);
           ODApplicationGroup odAG = odServer.getApplicationGroup(agid);
```

```
          System.out.println("Application Group Name = " + odAG.getName());
          System.out.println("Application Group Description = " + odAG.getDescription());
          System.out.println("Application Group ID = " + odAG.getId());
          System.out.println("isRMEnabled = " + odAG.isRMEnabled());
          System.out.println("isHoldEnabled = " + odAG.isHoldEnabled());
          System.out.println("isCFSODEnabled = " + odAG.isCFSODEnabled());
          System.out.println("Mapped Applications:");
          appls = odAG.getApplicationNames();
          for (int i = 0; i < appls.length; i++)
              System.out.println("    " +appls[i]);

          //----------
          // Cleanup
          //----------
          odFolder.close( );
          odServer.logoff( );
          odServer.terminate( );
          System.out.println( "" );
          System.out.println( "--------------------------------------------------" );
          System.out.println( "" );
          System.out.println( "Testcase TcApplGrp completed. - analyze results if required" );
             System.out.println( "" );
       }
       else
       {
          System.out.println( "" );
          System.out.println( "Testcase TcApplGrp failed  -" );
          System.out.println( "  ODConfig could not be initialized.  " );
          System.out.println( "  Probable cause: " );
          System.out.println( "    File arswww.props is not found in directory " +  argv[5] );
          System.out.println( "" );
       }

    }

    catch ( ODException e )
    {
        System.out.println( "ODException: " + e );
        System.out.println( "   id = " + e.getErrorId( ) );
        System.out.println( "  msg = " + e.getErrorMsg( ) );
        e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
        System.out.println( "exception: " + e2 );
        e2.printStackTrace( );
    }
  }
}
```

Searching a folder

An object of the ODFolder class represents a Content Manager OnDemand folder. An object of the ODCriteria class represents the search criteria for a Content Manager OnDemand folder. An object of the ODHit class represents a Content Manager OnDemand document. You use methods from these classes to search through a folder.

About this task

The following example uses ODFolder methods to open the specified folder, display the folder name, description, display order and search criteria, search the folder, and close the folder. This example uses ODCriteria methods to set the current search operand and search values. This example uses ODHit methods to do the following tasks:

- Get the display values for the document
- Get the document type
- Get a persistent identifier for the document
- Get the document location
- Get the MIME content type for the document

This example demonstrates the following ODFolder methods:

- close
- getName
- getDescription
- getDisplayOrder
- getCriteria
- getSearchMessage
- search
- setMaxHits

This example demonstrates the following ODCriteria methods:

- getName
- setOperator
- setSearchValue
- setSearchValues

This example demonstrates the following ODHit methods:

- getDisplayValue
- getDisplayValues
- getDocType
- getMimeType
- getDocLocation
- getDocId

Note: If you extract and reload the same document, the system generates a new DocID for the document, and the old DocID does not reference the new document.

This example also uses `ODServer` methods to prepare for logon, open the specified folder, and log off. This example demonstrates the following `ODServer` methods:

- initialize
- logoff
- logon
- openFolder
- terminate

This example uses the following runtime parameters:

- Criteria name
- Folder name
- Maximum hits
- Operator (must be one of eq, ne, lt, le, gt, ge, in, ni, li, nl, be, nb)
- Password
- Port
- Search value 1
- (optional) Search value 2
- Server name
- User Id

Important: The number of hits might be restricted by the MAXHITS specified in the ODConfig or passed to the search.

Example of searching a folder:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcSearch
{
    public static void main ( String argv[] )
    {
        ODServer odServer;
        ODFolder odFolder;
        ODCriteria odCrit;
        ODHit odHit;
        Enumeration values_enum;
        Vector hits;
        String[] display_crit;
        String server, userid, password, folder, directory;
        String crit = "", operator = "", value1 = "", value2 = "";
        String header, line1, line2, hit_value, useable_value;
        boolean mismatch_detected, use_default_values;
        int j, k, opr;
        int port, maxHits;

        //----------
        // If too few parameters, display syntax and get out
        //----------
        if ( argv.length <  10 && argv.length != 6 )
        {
            System.out.println( "usage: java TcSearch <server> <port> <userid> <password> <folder> " );
            System.out.println( "         <MaxHits (-1 to use default)> <criteria> <opr> <value1>
<value2>" );
            System.out.println( "       or, to use deafult search criteria" );
            System.out.println( "       java TcSearch <server> <port> <userid> <password> <folder>" );
            System.out.println( "         <MaxHits (-1 to use default)> " );
            return;
        }

        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "Testcase TcSearch started." );
            System.out.println( "This testcase should:" );
            System.out.println( "  Logon to the specified server" );
            System.out.println( "  Open the specified folder" );
            System.out.println( "  Display the folder name and description" );
            System.out.println( "  If specified:" );
            System.out.println( "    Get the specified criteria" );
            System.out.println( "    Set the operator" );
            System.out.println( "    Set the operand(s)" );
            System.out.println( "  Search the folder" );
            System.out.println( "  Display search message (if any)" );
            System.out.println( "  Display the number of hits" );
            System.out.println( "  Display the hitlist with each hit using 3 lines:" );
            System.out.println( "    1. The hit values returned by the ODHit.getDisplayValue method" );
            System.out.println( "    2. The hit values returned by the ODHit.getDisplayValues method" );
            System.out.println( "    3. The doc type, mime type, doc location, and doc id values" );
            System.out.println( "" );
            System.out.println( "Ensure that lines 1 and 2 of the hitlist are the same and that the" );
            System.out.println( "hitlist values are the same as those displayed using the Windows
Client." );
            System.out.println( "If arswww.props is restricting the number of hits, there may be
fewer" );
            System.out.println( "hits than displayed using the Windows Client." );
            System.out.println( "" );
            System.out.println( "-------------------------------------------------" );
            System.out.println( "" );

            //----------
            // Logon to specified server
            //----------
            use_default_values = argv.length == 6;
            server   = argv[0];
            port     = Integer.parseInt(argv[1]);
            userid   = argv[2];
            password = argv[3];
            folder   = argv[4];
            maxHits  = Integer.parseInt(argv[5]);
            crit      = argv[6];
```

```
            operator  = argv[7];
            value1    = argv[8];
            value2    = argv[9];

            ODConfig odConfig = new ODConfig();
            if(odConfig != null)
            {
                odServer = new ODServer(odConfig );
                odServer.initialize(  "TcSearch.java" );
                odServer.setPort(port);
                System.out.println( "Logging on to " + server  + " server with user " + userid + "..." );
                odServer.logon( server, userid, password);

                //----------
                // Open the specified folder
                //----------
                System.out.println( "Opening " + folder + " folder..." );
                odFolder = odServer.openFolder( folder );

                System.out.println( "Name='" + odFolder.getName( ) + "' Desc='" +
odFolder.getDescription( ) + "'" );

                //----------
                // If we are not using the default search values:
                //----------
                if ( !use_default_values )
                {
                    //----------
                    // Find the requested criteria
                    //----------
                    System.out.println( "Getting " + crit + " criteria..." );
                    odCrit = odFolder.getCriteria( crit );
                    if ( odCrit == null )
                        System.out.println( "  *** " + crit + " criteria does not exist - " );
                        System.out.println( "  NullPointerException will be reported" );

                    //----------
                    // Convert the operator parameter to the internal operator value and set
                    // the criteria operator
                    //----------
                    System.out.println( "Setting operator to " + operator + "..." );
                    if ( operator.equals( "eq" ) )
                        opr = ODConstant.OPEqual;
                    else if ( operator.equals( "ne" ) )
                        opr = ODConstant.OPNotEqual;
                    else if ( operator.equals( "lt" ) )
                        opr = ODConstant.OPLessThan;
                    else if ( operator.equals( "le" ) )
                        opr = ODConstant.OPLessThanEqual;
                    else if ( operator.equals( "gt" ) )
                        opr = ODConstant.OPGreaterThan;
                    else if ( operator.equals( "ge" ) )
                        opr = ODConstant.OPGreaterThanEqual;
                    else if ( operator.equals( "in" ) )
                        opr = ODConstant.OPIn;
                    else if ( operator.equals( "ni" ) )
                        opr = ODConstant.OPNotIn;
                    else if ( operator.equals( "li" ) )
                        opr = ODConstant.OPLike;
                    else if ( operator.equals( "nl" ) )
                        opr = ODConstant.OPNotLike;
                    else if ( operator.equals( "be" ) )
                        opr = ODConstant.OPBetween;
                    else if ( operator.equals( "nb" ) )
                        opr = ODConstant.OPNotBetween;
                    else
                        opr = -1;

                    System.out.println( "Setting operand(s)..." );
                    odCrit.setOperator( opr );

                    //----------
                    // Set the search values
                    //----------
                    if ( opr == ODConstant.OPBetween || opr == ODConstant.OPNotBetween )
                    {
                        odCrit.setSearchValues( value1, value2 );
                        System.out.println( "  " + odCrit.getName( ) + " " + getOperatorName( opr ) + "
" + value1);
                        System.out.println( " and " + value2 );
                    }
                    else
```

```
                    {
                        odCrit.setSearchValue( value1 );
                        System.out.println( "   " + odCrit.getName( ) + " " + getOperatorName( opr ) + "
" + value1 );
                    }
                }

                //----------
                // Set Max Hits limit if specified
                //----------
                if (maxHits != -1)
                {
                    System.out.println("Setting MaxHITS to " + maxHits + ".");
                    odFolder.setMaxHits(maxHits);
                }
                else
                    System.out.println("No MaxHits passed in. Will use the MaxHits in arswww.props/
ODConfig, ");
                    System.out.println("or the Folder defined max, which ever is less.");

                //----------
                // Search the folder
                //----------
                System.out.println( "  Searching " + folder + );
                System.out.println( "  ( use_default_values ? " using default values" : "" ) + "..." );

                long startTime  = System.currentTimeMillis();
                hits = odFolder.search( );
                long estimatedTime = System.currentTimeMillis() - startTime;
                System.out.println( "Elapsed Search Time: " +  estimatedTime + " ms");
                System.out.println( "    Search message: " + odFolder.getSearchMessage( ) );
                System.out.println( "    Number of hits: " + hits.size( ) );

                //----------
                // Display the hits
                //----------
                mismatch_detected = false;
                if ( hits != null && hits.size( ) > 0 )
                {
                    display_crit = odFolder.getDisplayOrder( );
                    header = "      ";
                    for( j = 0; j < display_crit.length; j++ )
                        header = header + display_crit[j] + "--";
                    System.out.println( "      ------------------------------------------------" );
                    System.out.println( header + " (from ODHit.getDisplayValue method)" );
                    System.out.println( header + " (from ODHit.getDisplayValues method)" );
                    System.out.println( "        DocType--MimeType--DocLocation--DocId" );
                    System.out.println( "      ------------------------------------------------" );
                    for ( j = 0; j < hits.size( ); j++ )
                    {
                        odHit = (ODHit)hits.elementAt( j );
                        line1 = "      ";
                        for ( k = 0; k < display_crit.length; k++ )
                        {
                            hit_value = odHit.getDisplayValue( display_crit[k] );
                            useable_value = ( hit_value.equals( "" ) ) ? " " : hit_value;
                            line1 = line1 + useable_value + "--";
                        }
                        System.out.println( line1 );
                        line2 = "      ";
                        for ( values_enum = odHit.getDisplayValues( ); values_enum.hasMoreElements( ); )
                        {
                            hit_value = (String)values_enum.nextElement( );
                            useable_value = ( hit_value.equals( "" ) ) ? " " : hit_value;
                            line2 = line2 + useable_value + "--";
                        }
                        System.out.println( line2 );
                        System.out.println( "        " + getDocTypeString( odHit.getDocType( ) ) +
                                "--" + odHit.getMimeType( ) +
                                "--" + getLocationString( odHit.getDocLocation( ) ) +
                                "--" + odHit.getDocId( ) );
                        if ( !line1.equals( line2 ) )
                            mismatch_detected = true;
                    }
                }


                //----------
                // Cleanup
                //----------
                odFolder.close( );
                odServer.logoff( );
```

```
                odServer.terminate( );
                System.out.println( "" );
                System.out.println( "-------------------------------------------------" );
                System.out.println( "" );
                System.out.println( "Testcase TcSearch completed - analyze if required" );
                System.out.println( "" );
                if ( mismatch_detected )
                {
                    System.out.println( "*** At least one mismatch was found between" );
                    System.out.println( "***   lines 1 and 2 of a hit" );
                    System.out.println( "" );
                }
            }
        }
        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "   id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }

    static String getOperatorName( int oper )
    {
        String str;

        switch( oper )
        {
        case ODConstant.OPEqual:
            str = "Equals";
            break;
        case ODConstant.OPNotEqual:
            str = "Not Equal";
            break;
        case ODConstant.OPLessThan:
            str = "Less Than";
            break;
        case ODConstant.OPLessThanEqual:
            str = "Less Than or Equal";
            break;
        case ODConstant.OPGreaterThan:
            str = "Greater Than";
            break;
        case ODConstant.OPGreaterThanEqual:
            str = "Greather Than or Equal";
            break;
        case ODConstant.OPIn:
            str = "In";
            break;
        case ODConstant.OPNotIn:
            str = "Not In";
            break;
        case ODConstant.OPLike:
            str = "Like";
            break;
        case ODConstant.OPNotLike:
            str = "Not Like";
            break;
        case ODConstant.OPBetween:
            str = "Between";
            break;
        case ODConstant.OPNotBetween:
            str = "Not Between";
            break;
        default:
            str = "Operator unknown";
            break;
        }

        return str;
    }

    static String getDocTypeString( char type )
    {
        String str;
```

```
        switch( type )
        {
        case ODConstant.FileTypeAFP:
            str = "AFP";
            break;
        case ODConstant.FileTypeBMP:
            str = "BMP";
            break;
        case ODConstant.FileTypeEMAIL:
            str = "EMAIL";
            break;
        case ODConstant.FileTypeGIF:
            str = "GIF";
            break;
        case ODConstant.FileTypeJFIF:
            str = "JFIF";
            break;
        case ODConstant.FileTypeLINE:
            str = "LINE";
            break;
        case ODConstant.FileTypeMETA:
            str = "META";
            break;
        case ODConstant.FileTypeNONE:
            str = "NONE";
            break;
        case ODConstant.FileTypePCX:
            str = "PCX";
            break;
        case ODConstant.FileTypePDF:
            str = "PDF";
            break;
        case ODConstant.FileTypePNG:
            str = "PNG";
            break;
        case  ODConstant.FileTypeSCS:
            str = "SCS";
            break;
        case ODConstant.FileTypeTIFF:
            str = "TIFF";
            break;
        case ODConstant.FileTypeUSRDEF:
            str = "USRDEF";
            break;
        default:
            str = "*** Invalid Doc Type ***";
            break;
        }

        return str;
    }

    static String getLocationString( int loc )
    {
        String str;

        switch( loc )
        {
        case ODConstant.DocLocationCache:
            str = "Cache";
            break;
        case ODConstant.DocLocationArchive:
            str = "Archive";
            break;
        case ODConstant.DocLocationExternal:
            str = "External";
            break;
        case ODConstant.DocLocationUnknown:
            str = "Unknown";
            break;
        default:
            str = "*** Invalid Doc Location ***";
            break;
        }

        return str;
    }
}
```

Searching a folder using an SQL string

An object of the ODFolder class represents a Content Manager OnDemand folder. An object of the ODHit class represents a Content Manager OnDemand document. You use methods from these classes to search through a folder using an SQL string.

About this task

The following example uses ODFolder methods to open the specified folder, search the folder with the specified SQL string, and close the folder. This example uses ODHit methods to display the number of items that match the query and to display the document list.

This example demonstrates the following ODFolder methods:

- getName
- getDescription
- getDisplayOrder
- setApplGroupForSearchWithSQL
- setMaxHits
- search
- close

This example demonstrates the following ODHit methods:

- getDisplayValue
- getDisplayValues
- getDocType
- getMimeType
- getDocLocation
- getDocId

This example also uses ODServer methods to prepare for logon, open the specified folder, and log off. This example demonstrates the following ODServer methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example uses the following runtime parameters:

- Server name
- Port
- User Id
- Password
- Folder name
- SQL string
- Max hits
- Date 1 (optional)
- Date 2 (optional)
- Date format (optional)
- Application group (optional)

Example of searching a folder using an SQL string:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcSQLSearch
{
    public static void main ( String argv[] )
    {
        ODServer odServer;
        ODFolder odFolder;
        ODHit odHit;
        Enumeration values_enum;
        Vector hits;
        String[] display_crit;
        String header, line1, line2, hit_value, useable_value;
        boolean mismatch_detected = false;
        int j, k, opr;

        //----------
        // If too few parameters, display syntax and get out
        //----------
        if ( argv.length < 6)
        {
            System.out.println( "usage: java TcSQLSearch <server> <port> <userid> <password>
<folder> <SQL>
                                    <MaxHits (-1 to use default)> <opt begin Date> <opt end
Date>
                                    <opt Date Format> <opt ApplGrpName>" );
            return;
        }

        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "Testcase TCSQLSearch started." );
            System.out.println( "This testcase should:" );
            System.out.println( "  Logon to the specified server" );
            System.out.println( "  Open the specified folder" );
            System.out.println( "  Display the folder name and description" );
            System.out.println( "  Set the application group (if specified)" );
            System.out.println( "  Search the folder using specified Sql" );
            System.out.println( "  Display search message (if any)" );
            System.out.println( "  Display the number of hits" );
            System.out.println( "  Display the hitlist with each hit using 3 lines:" );
            System.out.println( "    1. The hit values returned by the ODHit.getDisplayValue
method" );
            System.out.println( "    2. The hit values returned by the
ODHit.getDisplayValues method" );
            System.out.println( "    3. The doc type, mime type, doc location, and doc id
values" );
            System.out.println( "" );
            System.out.println( "Ensure that lines 1 and 2 of the hitlist are the same and
that the" );
            System.out.println( "hitlist values are the same as those displayed using the
Windows Client." );
            System.out.println( "If arswww.props is restricting the number of hits, there
may be fewer" );
            System.out.println( "hits than displayed using the Windows Client." );
            System.out.println( "" );
            System.out.println( "-----------------------------------------------------" );
            System.out.println( "" );

            //----------
            // Logon to specified server
            //----------
            ODConfig odConfig = new ODConfig();
            if(odConfig != null)
            {
```

```
                odServer = new ODServer(odConfig );
                odServer.initialize(  "TcSQLSearch.java" );
                odServer.setPort(Integer.parseInt(argv[1]));

                System.out.println( "Logging on to " + argv[0]  + " server with user " +
argv[2] + "..." );
                odServer.logon( argv[0], argv[2], argv[3]);

                //----------
                // Open the specified folder and find the requested criteria
                //----------
                System.out.println( "Opening " + argv[4] + " folder..." );
                odFolder = odServer.openFolder( argv[4] );
                System.out.println( "Name='" + odFolder.getName( ) + "' Desc='" +
odFolder.getDescription( ) + "'" );
                System.out.println( "Getting " + argv[5] + " criteria..." );

                //----------
                // Search with Application Group limiters if specified
                //----------
                if(argv.length == 10 )
                {
                    System.out.println("**Search is limited to Application Group " +
argv[8]);
                    odFolder.setApplGroupForSearchWithSQL(argv[9]);
                }
                //Set MaxHits
                int maxHits  = Integer.parseInt(argv[6]);
                if (maxHits != -1)
                {
                    System.out.println("Setting MaxHITS to " + maxHits + ".");
                    odFolder.setMaxHits(maxHits);
                }
                else
                    System.out.println("No MaxHits passed in. Will use the default, or the
Folder defined max,
                                       which ever is less.");


                //----------
                // Search with date limiters if specified
                //----------
                if(argv.length >= 9 ) //call search(sql,date1,date2,date format)
                {
                    String date1 = argv[7];
                    if(date1.compareTo(" ") != 0) //Only go in here if date was actually
spec'd
                    {
                        System.out.println("Searching " + argv[4] +" with Date Range limits
of " + argv[7] +
                                           " and " + argv[8]);
                        System.out.println("Date format is " + argv[9]);

                        hits = odFolder.search(argv[5],
                                argv[7],  //date1
                                (argv.length > 9) ?  argv[8] : " ",   //date2
                                        (argv.length >= 10) ? argv[9] : " ",    //date format
                                        null);      // null
                    }
                    else
                    {
                        //----------
                        // Search the folder
                        //----------
                        System.out.println( "  Searching " + argv[4] + "with SQL " + argv[5]
+ "..." );
                        hits = odFolder.search(argv[5] );
                    }
                }
                else if(argv.length > 7 && argv.length < 10 ) //call search(sql,date1,date2)
                {
                    String date1 = argv[7];
```

```
                    if(date1.compareTo(" ") != 0) //Only go in here if date was actually
spec'd
                    {
                        System.out.println("Searching " + argv[4] +" with Date Range limits
of " + argv[7] +
                                           " and " +argv[8]);

                        hits = odFolder.search(argv[5],
                                argv[7],  //date1
                                (argv.length > 9) ?  "" : argv[8]);   //date2

                    }
                    else //call search(sql)
                    {
                        //----------
                        // Search the folder
                        //----------
                        System.out.println( "  Searching " + argv[4] + "with SQL " + argv[5]
+ "..." );
                        hits = odFolder.search(argv[5] );
                    }
                }
                else //call search(sql)
                {
                    //----------
                    // Search the folder
                    //----------
                    System.out.println( "  Searching " + argv[4] + "with SQL " + argv[5] +
"..." );
                    hits = odFolder.search(argv[5] );
                }



                //----------
                // Display the hits
                //----------
                System.out.println(" Number of Hits Found = " + hits.size());
                mismatch_detected = false;
                if ( hits != null && hits.size( ) > 0 )
                {
                    display_crit = odFolder.getDisplayOrder( );
                    header = "      ";
                    for( j = 0; j < display_crit.length; j++ )
                        header = header + display_crit[j] + "--";
                    System.out.println( "
----------------------------------------------" );
                    System.out.println( header + " (from ODHit.getDisplayValue method)" );
                    System.out.println( header + " (from ODHit.getDisplayValues method)" );
                    System.out.println( "        DocType--MimeType--DocLocation--DocId" );
                    System.out.println( "
----------------------------------------------" );
                    for ( j = 0; j < hits.size( ); j++ )
                    {
                        odHit = (ODHit)hits.elementAt( j );
                        line1 = "      ";
                        for ( k = 0; k < display_crit.length; k++ )
                        {
                            hit_value = odHit.getDisplayValue( display_crit[k] );
                            useable_value = ( hit_value.equals( "" ) ) ? " " : hit_value;

                            line1 = line1 + useable_value + "--";
                        }
                        System.out.println( line1 );
                        line2 = "      ";
                        for ( values_enum = odHit.getDisplayValues( );
values_enum.hasMoreElements( ); )
                        {
                            hit_value = (String)values_enum.nextElement( );
                            useable_value = ( hit_value.equals( "" ) ) ? " " : hit_value;
                            line2 = line2 +useable_value + "--";
                        }
```

```
                        System.out.println( line2 );
                        System.out.println( "        " +
getDocTypeString( odHit.getDocType( ) ) +
                                "--" + odHit.getMimeType( ) +
                                "--" + getLocationString( odHit.getDocLocation( ) ) +
                                "--" + odHit.getDocId( ) );
                        if ( !line1.equals( line2 ) )
                            mismatch_detected = true;
                    }
                }


                //----------
                // Cleanup
                //----------
                odFolder.close( );
                odServer.logoff( );
                odServer.terminate( );
                System.out.println( "" );
                System.out.println( "-----------------------------------------------------" );
                System.out.println( "" );
                System.out.println( "Testcase TcSQLSearch completed - analyze if required" );
                System.out.println( "" );
                if ( mismatch_detected )
                {
                    System.out.println( "*** At least one mismatch was found between" );
                    System.out.println( "***   lines 1 and 2 of a hit" );
                    System.out.println( "" );
                }
            }
        }

        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "   id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }

    static String getOperatorName( int oper )
    {
        String str;

        switch( oper )
        {
        case ODConstant.OPEqual:
            str = "Equals";
        break;
        case ODConstant.OPNotEqual:
            str = "Not Equal";
        break;
        case ODConstant.OPLessThan:
            str = "Less Than";
        break;
        case ODConstant.OPLessThanEqual:
            str = "Less Than or Equal";
        break;
        case ODConstant.OPGreaterThan:
            str = "Greater Than";
        break;
        case ODConstant.OPGreaterThanEqual:
            str = "Greather Than or Equal";
        break;
        case ODConstant.OPIn:
```

```
            str = "In";
        break;
        case ODConstant.OPNotIn:
            str = "Not In";
        break;
        case ODConstant.OPLike:
            str = "Like";
        break;
        case ODConstant.OPNotLike:
            str = "Not Like";
        break;
        case ODConstant.OPBetween:
            str = "Between";
        break;
        case ODConstant.OPNotBetween:
            str = "Not Between";
        break;
        default:
            str = "Operator unknown";
        break;
        }

        return str;
    }

    static String getDocTypeString( char type )
    {
        String str;

        switch( type )
        {
        case ODConstant.FileTypeAFP:
            str = "AFP";
        break;
        case ODConstant.FileTypeBMP:
            str = "BMP";
        break;
        case ODConstant.FileTypeEMAIL:
            str = "EMAIL";
        break;
        case ODConstant.FileTypeGIF:
            str = "GIF";
        break;
        case ODConstant.FileTypeJFIF:
            str = "JFIF";
        break;
        case ODConstant.FileTypeLINE:
            str = "LINE";
        break;
        case ODConstant.FileTypeMETA:
            str = "META";
        break;
        case ODConstant.FileTypeNONE:
            str = "NONE";
        break;
        case ODConstant.FileTypePCX:
            str = "PCX";
        break;
        case ODConstant.FileTypePDF:
            str = "PDF";
        break;
        case ODConstant.FileTypePNG:
            str = "PNG";
        break;
        case ODConstant.FileTypeSCS:
            str = "SCS";
        break;
        case ODConstant.FileTypeTIFF:
            str = "TIFF";
        break;
        case ODConstant.FileTypeUSRDEF:
            str = "USRDEF";
```

```
        break;
        default:
            str = "*** Invalid Doc Type ***";
        break;
        }

        return str;
    }

    static String getLocationString( int loc )
    {
        String str;

        switch( loc )
        {
        case ODConstant.DocLocationCache:
            str = "Cache";
        break;
        case ODConstant.DocLocationArchive:
            str = "Archive";
        break;
        case ODConstant.DocLocationExternal:
            str = "External";
        break;
        case ODConstant.DocLocationUnknown:
            str = "Unknown";
        break;
        default:
            str = "*** Invalid Doc Location ***";
        break;
        }

        return str;
    }
}
```

Cancelling a search

The cancel method in the ODServer class cancels a search in progress.

About this task

The following example uses the ODServer.cancel method to cancel a search in progress.

This example uses ODServer, ODFolder, and ODCriteria methods to log on to a server, open a folder, and set the Date criteria to 1970-2001. A second thread is then initiated to do a search. When the second thread completes, the number of hits is displayed. A second thread is again initiated, to do a search. The process is put to sleep for 0.5 seconds and then the search is canceled. When the second thread completes, the number of hits is displayed.

This example demonstrates the following ODServer methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates the following ODFolder methods:

- getName
- getCriteria
- search
- close

This example demonstrates the following ODCriteria methods:

- setOperator
- setSearchValues

This example uses the following runtime parameters:

- Server name
- Port
- User Id
- Password
- Folder name

Example of canceling a search:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

class TestThread extends Thread
{
    ODFolder odFolder;

    TestThread( ODFolder fld )
    {
        odFolder = fld;
    }

    public void run( )
    {
        Vector hits;

        try
        {
            System.out.println( "  Second thread Searching..." );
            hits = odFolder.search( );
            System.out.println( "  Search completed - Number of hits: " + hits.size( ) );
        }

        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "    id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}

public class TcCancelSearch
{
    public static void main ( String argv[] )
    {
        ODServer odServer;
        ODFolder odFolder;
        ODCriteria odCrit;
        TestThread search_thread;
        int j;

        //----------
        // If too few parameters, display syntax and get out
        //----------
        if ( argv.length < 8 )
```

```
        {
            System.out.println( "usage: java TcCancelSearch <server> <port> <userid>
<password> <folder> " +
                                " <date field name> " +
                                " <from date mm/dd/yy or mm/dd/yyyy> " +
                                " <to date mm/dd/yy or mm/dd/yyyy> " );
            return;
        }

        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "Testcase TcCancelSearch started." );
            System.out.println( "This testcase should:" );
            System.out.println( "  Logon to the specified server" );
            System.out.println( "  Open the specified folder" );
            System.out.println( "  Set '" + argv[5] + "' criteria to " + argv[6] + " to " +
argv[7] );
            System.out.println( "  Initiate a second thread to perform the search" );
            System.out.println( "  When second thread completes, display the number of
hits" );
            System.out.println( "  Initiate a second thread to perform the search" );
            System.out.println( "  Sleep for .02 seconds" );
            System.out.println( "  Cancel the search" );
            System.out.println( "  When second thread completes, display the number of
hits" );
            System.out.println( "" );
            System.out.println( "Ensure that a folder is chosen that includes a criteria
named Date." );
            System.out.println( "Ensure that the folder contains many hits and that
arswww.props is" );
            System.out.println( "not overly restricting the number of hits which can be
returned." );
            System.out.println( "" );
            System.out.println( "-----------------------------------------------------" );
            System.out.println( "" );

            //----------
            // Logon to specified server
            //----------
            ODConfig odConfig = new ODConfig();
            if(odConfig != null)
            {
                odServer = new ODServer(odConfig );
                odServer.initialize( "TcCancelSearch.java" );
                System.out.println( "Logging on to " + argv[0]  + " server with user " +
argv[2] + "..." );
                odServer.setPort(Integer.parseInt(argv[1]));
                odServer.logon( argv[0], argv[2], argv[3]);

                //----------
                // Open the specified folder and display its name and description
                //----------
                System.out.println( "Opening " + argv[4] + "..." );
                odFolder = odServer.openFolder( argv[4] );
                odCrit = odFolder.getCriteria( argv[5] );
                odCrit.setOperator( ODConstant.OPBetween );
                odCrit.setSearchValues( argv[6], argv[7] );

                //----------
                // Start a search on a different thread, sleep briefly, awake and cancel
search
                //----------
                System.out.println( "Main thread initiating search (will not attempt to
cancel)..." );
                System.out.println( "  Searching " + odFolder.getName( ) + "..." );
                search_thread = new TestThread( odFolder );
                search_thread.start( );
                search_thread.join( );
```

```
                System.out.println( "Main thread initiating search (will attempt to
cancel)..." );
                search_thread = new TestThread( odFolder );
                search_thread.start( );
                System.out.println( "Main thread sleeping for .02 seconds..." );
                ( Thread.currentThread( ) ).sleep( 20 );
                System.out.println( "Main thread attempting to cancel search..." );
                odServer.cancel( );
                System.out.println( "Main thread returned from attempt to cancel" );
                search_thread.join( );

                //----------
                // Cleanup
                //----------
                odFolder.close( );
                odServer.logoff( );
                odServer.terminate( );
                System.out.println( "" );
                System.out.println( "--------------------------------------------------" );
                System.out.println( "" );
                System.out.println( "Testcase TcCancelSearch completed. - Ensure that the
second search," );
                System.out.println( "  which was cancelled, yielded fewer hits than the
first" );
                System.out.println( "" );
            }
        }


        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "   id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}
```

Listing search criteria

You use methods in the ODCriteria class to list the search criteria for a specified folder.

About this task

The following example demonstrates how to use ODCriteria methods to list the search criteria for a specified folder. For each search field, this example lists the name of the search field, the default operator, the operators that are valid for the field, the field type, and any default search values. The default values are listed by the ODCriteria.getSearchValues method. Fixed search values are listed for any search fields that are defined as FixedChoice or Segment.

This example demonstrates the following ODCriteria methods:

- getName
- getOperator
- getValidOperators
- getType
- getDefaultFmt
- getDisplayFmt

- getDisplayFmtQual
- getMaxEntryChars
- getMaxDisplayChars
- getMinSearchValue
- getMaxSearchValue
- getDBFieldNames
- getDBFieldMask
- getSearchValues
- getFixedValues
- isUpdateable
- isRequired
- isDefaultValueAvailable
- isDefaultValueFixed

This example demonstrates the following ODServer methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates the following ODFolder methods:

- getName
- getDescription
- getNumApplGroups
- getApplGroupNames
- getNumCriteria
- getCriteria
- close

This example uses the following runtime parameters:

- Server name
- Port
- User Id
- Password
- Folder name

Example of accessing search criteria:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;
public class TcListCriteria
{
    public static void main ( String argv[] )
    {
        ODServer odServer;
        ODFolder odFolder;
        ODCriteria odCrit;
        Enumeration crit_enum;
        String[] value_vec;
        String[] search_values, fixed_values, database_fields;
```

```
        Object[] appl_grps;
        int[] valid_oprs;
        int j, k, opr;
        char field_type;

        //----------
        // If too few parameters, display syntax and get out
        //----------
    if ( argv.length < 5 )
    {
          System.out.println( "usage: java TcListCriteria <server> <port> <userid>
<password> <folder>" );
           return;
        }

        try
        {
          //----------
          // Set the stage
          //----------
          System.out.println( "Test case TcListCriteria started." );
          System.out.println( "This test case should:" );
          System.out.println( "  Logon to the specified server" );
          System.out.println( "  Open the specified folder" );
          System.out.println( "  Display the folder name and description" );
          System.out.println( "  Display the number and names of folder application
groups" );
          System.out.println( "  Display the number of folder criteria" );
          System.out.println( "  For each criteria, display the" );
          System.out.println( "    Name" );
          System.out.println( "    Default operator" );
          System.out.println( "    Valid operators" );
          System.out.println( "    Field Type" );
          System.out.println( "    Required, DefaultValueAvailable, and DefaultValueFixed
(true/false) flags" );
          System.out.println( "    Maximum Entry Characters" );
          System.out.println( "    Maximum Display Characters" );
          System.out.println( "    Number and names of database fields" );
          System.out.println( "    Default values (by ODCrit.getSearchValues method)" );
          System.out.println( "    Default values (by ODCrit.getValues method)" );
          System.out.println( "    Fixed values (only for FixedChoice and Segment
criteria)" );
          System.out.println( "" );
          System.out.println( "Ensure that none of the operators indicates 'Unknown
operator'," );
          System.out.println( "that none of the field types indicates 'Unknown type',  that
the" );
          System.out.println( "default values are the same for each method, and that all" );
          System.out.println( "information is the same as that displayed using the Windows
Client." );
          System.out.println( "" );
          System.out.println( "-------------------------------------------------" );
          System.out.println( "" );

            ODConfig odConfig = new ODConfig();
            if(odConfig != null)
            {
                odServer = new ODServer(odConfig );
                odServer.initialize(  "TcListCriteria.java" );
                System.out.println( "Logging on to " + argv[0]  + " server with user " +
argv[2] + "..." );
                odServer.setPort(Integer.parseInt(argv[1]));
                odServer.logon( argv[0], argv[2], argv[3]);

                //----------
                // Open the specified folder and display its name and description
                //----------
                System.out.println( "Opening " + argv[4] + " folder..." );
                odFolder = odServer.openFolder( argv[4] );
                System.out.println( "Name='" + odFolder.getName( ) + "' Desc='" +
odFolder.getDescription( ) + "'" );
```

```
                //----------
                // Display number and names of application groups
                //----------
                System.out.println( "There is(are) " + odFolder.getNumApplGroups( ) +
                                    " application group(s) in the folder:" );
                appl_grps = odFolder.getApplGroupNames( );
                for ( j = 0; j < appl_grps.length; j++ )
                    System.out.println( "  " + appl_grps[j].toString( ) );

                //----------
                // For each folder criteria,
                //----------
                System.out.println( "There are " + odFolder.getNumCriteria( ) + "
criteria:" );
                for ( crit_enum = odFolder.getCriteria( ); crit_enum.hasMoreElements( ); )
                {
                    //----------
                    // Display criteria name
                    //----------
                    System.out.println( "" );
                    odCrit = (ODCriteria)crit_enum.nextElement( );
                    System.out.println( odCrit.getName( ) );

                    //----------
                    // Display default operator
                    //----------
                    opr = odCrit.getOperator( );
                    System.out.println( "  Default operator: " );
                    System.out.println( "    " + getOperatorName( opr ) );

                    //----------
                    // Display valid operators
                    //----------
                    valid_oprs = odCrit.getValidOperators() ;
                    System.out.println( "  Valid operators:" );
                    for ( j = 0; j < valid_oprs.length; j++ )
                        System.out.println( "    " + getOperatorName( valid_oprs[j] ) );

                    //----------
                    // Display field type
                    //----------
                    field_type = odCrit.getType( );
                    System.out.println( "  Type:" );
                    System.out.println( "    " + getTypeName( field_type ) );

                    //----------
                    // Display field format for Date
                    //----------
                    if ( odCrit.getDefaultFmt() != null)
                    {
                        System.out.println("  Date Format:" );
                        System.out.println("    DefaultDisplay Fmt " +
odCrit.getDefaultFmt());
                        System.out.println("    Display Fmt " + odCrit.getDisplayFmt());
                    }

                    System.out.println("    DisplayFmtQualifier " +
odCrit.getDisplayFmtQual());

                    //----------
                    // Display field id mask info
                    //----------
                    System.out.println( "  FieldIdMask:" );
                    System.out.println( "    Required=" + odCrit.isRequired( ) +
                            " Default=" + odCrit.isDefaultValueAvailable( ) +
                            " Fixed Default=" + odCrit.isDefaultValueFixed( ) );
                    //----------
                    // Display max entry chars
                    //----------
                    System.out.println( "  MaxEntryChars:" );
                    System.out.println( "    " + odCrit.getMaxEntryChars( ) );
```

```
                    //----------
                    // Display max display chars
                    //----------
                    System.out.println( "  MaxDisplayChars:" );
                    System.out.println( "    " + odCrit.getMaxDisplayChars( ) );

                    //----------
                    // Display min search values
                    //----------
                    System.out.println( "  MinSearchValue:" );
                    System.out.println( "    " + odCrit.getMinSearchValue( ) );
                    //----------
                    // Display max search values
                    //----------
                    System.out.println( "  MaxSearchValue:" );
                    System.out.println( "    " + odCrit.getMaxSearchValue( ) );

                    //----------
                    // Display the database field names
                    //----------
                    System.out.println("  Database fields:");
                    for (j = 0; j < appl_grps.length; j++)
                    {
                        database_fields = odCrit.getDBFieldNames((String)appl_grps[j]);
                       if(database_fields == null)
                            System.out.println("No DBFields Defined");
                       else
                       {
                              System.out.println("    " + database_fields.length + " for
ApplGroup '" +
                                                 (String)appl_grps[j] + "':");
                              for (k = 0; k < database_fields.length; k++)
                              {
                                   System.out.println("      DB Field Name = " +
                                           (database_fields[k].equals("") ? "[empty
string]" : database_fields[k]));
                                   long fieldMask =
odCrit.getDBFieldMask((String)appl_grps[j],database_fields[k]);
                                   System.out.println("        Field Mask - " + fieldMask);

                                   System.out.println( "      Application ID Field = " +
                                         ((fieldMask & ODConstant.OD_FLDMSK_APPL )== 0
                                          ? false : true ));
                                   boolean update_field =
odCrit.isUpdateable((String)appl_grps[j], database_fields[k]);
                                   if (update_field)
                                      System.out.println("        Field is Updateable" );
                              }
                       }
                     }
                    //----------
                    // Display default value(s) using ODCrit.getValues( )
                    //----------
                    value_vec = odCrit.getSearchValues( );
                    System.out.println("  Default Value(s) (ODCrit.getValues method):");
                    System.out.println( "    '" + value_vec[ 0 ] + "'" );
                    System.out.println( "    '" + value_vec[ 1 ] + "'" );

                    //----------
                    // Display default value(s) using ODCrit.getSearchValues( )
                    //----------
                    search_values = odCrit.getSearchValues( );
                    System.out.println("  Default Values (ODCrit.getSearchValues method):");
                    for ( j = 0; j < search_values.length; j++ )
                        System.out.println("    '" + search_values[j] + "'" );

                    //----------
                    // Display fixed choices
                    //----------
                    switch ( field_type )
                    {
                    case ODConstant.InputTypeChoice:
```

```
                    case ODConstant.InputTypeSegment:
                        fixed_values = odCrit.getFixedValues( );
                    System.out.println("  Fixed Values (only for field types FixedChoice and
Segment):");
                    for ( j = 0; j < fixed_values.length; j++ )
                        System.out.println("    '" + fixed_values[j] + "'" );
                    break;
                    }
                }

                //----------
                // Cleanup
                //----------
                odFolder.close( );
                odServer.logoff( );
                odServer.terminate( );
                System.out.println( "" );
                System.out.println( "-----------------------------------------------------" );
                System.out.println( "" );
                System.out.println( "Test case TcListCriteria completed -" );
                System.out.println( "  analyze and compare results to Windows Client if
required" );
                System.out.println( "" );
            }
        }

        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "   id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }

    static String getOperatorName( int oper )
    {
        String str;

        switch( oper )
        {
        case ODConstant.OPEqual:
            str = "Equal";
        break;
        case ODConstant.OPNotEqual:
            str = "Not Equal";
        break;
        case ODConstant.OPLessThan:
            str = "Less Than";
        break;
        case ODConstant.OPLessThanEqual:
            str = "Less Than or Equal";
        break;
        case ODConstant.OPGreaterThan:
            str = "Greater Than";
        break;
        case ODConstant.OPGreaterThanEqual:
            str = "Greater Than or Equal";
        break;
        case ODConstant.OPIn:
            str = "In";
        break;
        case ODConstant.OPNotIn:
            str = "Not In";
        break;
        case ODConstant.OPLike:
```

```
            str = "Like";
        break;
        case ODConstant.OPNotLike:
            str = "Not Like";
        break;
        case ODConstant.OPBetween:
            str = "Between";
        break;
        case ODConstant.OPNotBetween:
            str = "Not Between";
        break;
        default:
            str = "*** Unknown operator";
        break;
        }

        return str;
    }

    static String getTypeName( char type )
    {
        String str;

        switch( type )
        {
        case ODConstant.InputTypeNormal:
            str = "Normal";
        break;
        case ODConstant.InputTypeTextSearch:
            str = "TextSearch";
        break;
        case ODConstant.InputTypeNoteTextSearch:
            str = "NoteTextSearch";
        break;
        case ODConstant.InputTypeNoteColor:
            str = "NoteColor";
        break;
        case ODConstant.InputTypeChoice:
            str = "FixedChoice";
        break;
        case ODConstant.InputTypeSegment:
            str = "Segment";
        break;
        default:
            str = "*** Unknown type";
        break;
        }

        return str;
    }
}
```

Listing folders and folder information

You use methods from the ODServer class to create a list of folders. Then, you write code that prints the name and description of a folder on one line.

About this task

The following example uses ODServer methods to print a line that shows the number of folders on the specified server that might be searched by the specified user ID. The example prints one line for each folder, showing the folder name and description.

This example demonstrates the following ODServer methods:

- initialize
- logon

- getNumFolders
- getFolderNames
- getFolderDescription
- logoff
- terminate

This example uses the following runtime parameters:

- Server name
- Port
- User Id
- Password

Example of listing folders and folder information:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcListFolders
{
    public static void main ( String argv[] )
    {
        ODServer    odServer;
        Enumeration folders_enum;
        String folder_name;
        String folder_desc;
        int num_folders;

        //----------
        // If too few parameters, display syntax and get out
        //----------
        if ( argv.length < 4 )
        {
            System.out.println( "usage: java TcListFolders <server> <port> <userid>
<password>" );
            return;
        }

        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "Testcase TcListFolders started." );
            System.out.println( "This testcase should:" );
            System.out.println( "  Display a line showing number of folders on the server
available to the userid" );
            System.out.println( "  Display one line for each folder, showing name and
description" );
            System.out.println( "" );
            System.out.println( "The information should be the same as that displayed using
the Windows Client" );
            System.out.println( "(with the 'All' button checked if available), but the
sequence of the folders" );
            System.out.println( "may be different depending on the server specified" );
            System.out.println( "" );
            System.out.println( "-------------------------------------------------" );
            System.out.println( "" );

            //----------
            // Logon to specified server
            //----------
            ODConfig odConfig = new ODConfig();
            if(odConfig != null)
            {
                odServer = new ODServer(odConfig );
                odServer.initialize(  "TcListFolders.java" );
```

```
                System.out.println( "Logging on to " + argv[0]  + " server with user " +
argv[2] + "..." );
                odServer.setPort(Integer.parseInt(argv[1]));
                odServer.logon( argv[0], argv[2], argv[3]);

                //----------
                // Display the number of folders available.
                //----------
                num_folders = odServer.getNumFolders( );
                System.out.println( "" );
                System.out.println( "There are " + num_folders + " folders available to " +
argv[2] + " on "
                                    + argv[0] + ":" );

                //----------
                // Display the folder names and descriptions
                //----------
                for ( folders_enum = odServer.getFolderNames( );
folders_enum.hasMoreElements( ); )
                {
                    folder_name = (String)folders_enum.nextElement( );
                    folder_desc = odServer.getFolderDescription( folder_name );
                    System.out.println( "  " + folder_name + "  ---  " + folder_desc );
                }

                //----------
                // Cleanup
                //----------
                odServer.logoff( );
                odServer.terminate( );
                System.out.println( "" );
                System.out.println( "-----------------------------------------------------" );
                System.out.println( "" );
                System.out.println( "Testcase TcListFolders completed - compare results to
Windows Client if required" );
                System.out.println( "" );
            }
        }

        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "   id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }
        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}
```

Displaying folder criteria information

You use methods from the ODServer class to open a folder on a specified server, then display the folder name and description. Then, you write code that prints the folder criteria information.

About this task

The following example uses ODServer methods to open a folder on a specified server, displays the folder name and description, and prints the folder criteria information. The folder criteria includes:

- Name
- Default operator
- Valid operators

- Field Type
- Default values
- Fixed values

To display folder criteria information, ensure that:

- None of the operators indicate `Unknown operator`.
- None of the field types indicate `Unknown type`.
- For each method, the default values are the same.
- The folder criteria information that is displayed by ODWEK API matches what is displayed by the Content Manager OnDemand client.

This example demonstrates the following ODServer methods:

- initialize
- logon
- getNumFolders
- getFolderNames
- getFolderDescription
- logoff
- terminate

This example uses the following runtime parameters:

- Server name
- Port
- User ID
- Password
- Folder criteria

This example uses the following runtime parameters:

- Server name
- User ID
- Password
- Folder name

Example of listing folders and folder information:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;
public class TcListFoldersByCrit
{
  public static void main ( String argv[] )
  {
     ODServer odServer;
     Enumeration folders_enum;
     String folder_name;
     String folder_desc;
     int num_folders;
     //----------
     // If too few parameters, display syntax and get out
     //----------
     if ( argv.length < 5 )
     {
         System.out.println( "usage: java TcListFoldersByCrit <server> <port> <userid>
<password> <Folder Criteria>" );
         return;
     }
     try
```

```
     {
          //----------
          // Set the stage
          //----------
          System.out.println( "Testcase TcListFoldersByCrit started." );
          System.out.println( "This testcase should:" );
          System.out.println( "  Log onto the server.");
          System.out.println( "  Display the total number of folders available for the
user.");
          System.out.println( "  Display the number of folders available for the user with
criteria.");
          System.out.println( "  Display one line for each folder retrieved from search
criteria, ");
          System.out.println( "          showing name and description" );
          System.out.println( "" );
          System.out.println( "The information should be the same as that displayed using
the Windows Client" );
          System.out.println( "(with the 'All' button checked if available), but the
sequence of the folders" );
          System.out.println( "may be different depending on the server specified" );
          System.out.println( "" );
          System.out.println( "--------------------------------------------------" );
          System.out.println( "" );
          //----------
          // Logon to specified server
          //----------
          ODConfig odConfig = new ODConfig();
          if(odConfig != null)
          {
             odServer = new ODServer(odConfig );
             odServer.initialize(  "TcListFolders.java" );
             System.out.println( "Logging on to " + argv[0]  + " server with user " +
argv[2] + "..." );
             odServer.setPort(Integer.parseInt(argv[1]));
             odServer.logon( argv[0], argv[2], argv[3]);

             //----------
             // Display the total number of folders available.
             //----------
             num_folders = odServer.getNumFolders( );
             System.out.println( "" );
             System.out.println( "There are " + num_folders + " folders available to user "
+ argv[2]);
             System.out.println( "      on server " + argv[0] + "." );

             //----------
             // Display the number of folders available with search criteria.
             //----------
             num_folders = odServer.getNumFolders(argv[4]);
             System.out.println( "" );
             System.out.println( "There are " + num_folders + " folders with criteria " +
argv[4] );
             System.out.println( "      available to user " + argv[2] + " on server " +
argv[0] + "." );

             //----------
             // Display the folder names and descriptions
             //----------
             int i = 1;
             // get a limited folder list based on search criteria
             System.out.println( "" );
             System.out.println("List folders found using search criteria... " + argv[4] +
":");
             System.out.println( "" );
             i = 1;
             for (Enumeration folders_enum2 = odServer.getFolderNames(argv[4]);
folders_enum2.hasMoreElements( ); )
              {
                 folder_name = (String)folders_enum2.nextElement( );
                 folder_desc = odServer.getFolderDescription( folder_name );
                 System.out.println(i++ + "   " + folder_name + "  ---   " + folder_desc );
              }
```

```
            //----------
            // Cleanup
            //----------
            odServer.logoff( );
            odServer.terminate( );
            System.out.println( "" );
            System.out.println( "-------------------------------------------------" );
            System.out.println( "" );
            System.out.println( "Testcase TcListFoldersByCrit completed - ");
            System.out.println("          compare results to Windows Client if required" );
            System.out.println( "" );
         }
      }

      catch ( ODException e )
      {
        System.out.println( "ODException: " + e );
        System.out.println( "   id = " + e.getErrorId( ) );
        System.out.println( "  msg = " + e.getErrorMsg( ) );
        e.printStackTrace( );
      }
      catch ( Exception e2 )
      {
        System.out.println( "exception: " + e2 );
        e2.printStackTrace( );
      }
   }
}
```

Displaying a list of documents

You use methods from the ODCriteria, ODFolder, and ODHit classes to display a list of documents from a folder.

About this task

The following example uses ODCriteria, ODFolder, and ODHit methods to do the following tasks:

- Search a folder using the default search criteria
- Print the number of documents that matched the query
- Lists the documents that matched the query

This example demonstrates the following ODFolder methods:

- getDisplayOrder
- getFolderSortOrder
- getCriteria
- getSortLocation
- setSortLocation
- search
- close

This example demonstrates the following ODHit methods:

- getDisplayValue

This example also demonstrates these ODServer methods:

- initialize
- logon
- openFolder

- logoff
- terminate

This example uses the following runtime parameters:

- Server name
- User Id
- Password
- Folder name

Example of displaying a list of documents:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcSortedHitlist
{
  public static void main ( String argv[] )
  {
        ODServer odServer;
        ODFolder odFolder;
        ODCriteria odCrit;
        Enumeration crit_enum;
        String server, userid, password, folder, crit1, crit2, crit3;
        boolean ascend1, ascend2, ascend3;
        //----------
        // If too few parameters, display syntax and get out
        //----------
    if ( argv.length < 11 )
    {
      System.out.println( "usage: java TcSortedHitlist <server> <port> <userid> <password>
<folder> <criteria1>" +
                          " <ascending1> <criteria2> <ascending2> <criteria3>
<ascending3>" );
      return;
    }

    try
    {
            //----------
            // Set the stage
            //----------
            System.out.println("Testcase TcSortedHitlist started." );
            System.out.println("This testcase should:");
            System.out.println("  Logon to the specified server");
            System.out.println("  Open the specified folder");
            System.out.println("  1. Display sort order information");
            System.out.println("     Sort is disabled");
            System.out.println("     Search the folder using the default criteria");
            System.out.println("     Display the hitlist");
            System.out.println("  2. Display sort order information");
            System.out.println("     Sort is enabled");
            System.out.println("     Search the folder using default folder sort criteria");
            System.out.println("     Search folder using setSortLocation(MIDTIER)");
            System.out.println("     Search the folder using default criteria");
            System.out.println("     Display the hitlist");
            System.out.println("  3. Display sort order information");
            System.out.println("     Sort is disabled");
            System.out.println("     Search folder using setSortLocation(NONE)");
            System.out.println("     Search the folder using the default criteria");
            System.out.println("     Display the hitlist");
            System.out.println("  4. Display sort order information");
            System.out.println("     Sort is enabled");
            System.out.println("     Search folder using setSortLocation(MIDTIER)");
            System.out.println("     Search the folder using the default criteria");
            System.out.println("     Display the hitlist");
            System.out.println("  5. Display sort order information");
            System.out.println("     Sort is enabled");
            System.out.println("     Search folder using setSortLocation(SERVER)");
```

```
            System.out.println("     Search the folder using the default criteria");
            System.out.println("     Display the hitlist");
            System.out.println("");
            System.out.println("-----------------------------------------------------");
            System.out.println("");

            //----------
            // Logon to the server
            //----------
            server = argv[0];
            userid     = argv[2];
            password   = argv[3];
            folder     = argv[4];
            crit1      = argv[5];
            ascend1    = argv[6].equals( "0" ) ?false:true;
            crit2      = argv[7];
            ascend2    = argv[8].equals( "0" ) ?false:true;
            crit3      = argv[9];
            ascend3    = argv[10].equals( "0" ) ?false:true;

            ODConfig odConfig = new ODConfig();
            if(odConfig != null)
            {
                odServer = new ODServer(odConfig );
                odServer.initialize( "TcSortedHitlist.java" );
                odServer.setPort(Integer.parseInt(argv[1]));

                System.out.println( "Logging on to " + server  + " server with user " +
userid + "..." );
                odServer.logon( server, userid, password);

                //----------
                // Open the folder and display default folder sort order
                //----------
                System.out.println("Opening " + folder + "...");
                odFolder = odServer.openFolder(folder);
                System.out.println("Sort Location = " + odFolder.getSortLocation());
                displayFolderSortOrder(odFolder);
                //----------
                // Search the folder with sort disabled by default
                //----------
                System.out.println("With sort disabled by default:");
                System.out.println("  Searching with SORT=NONE...");
                searchAndListHits(odFolder);
                //----------
                // Search the folder after enabling sort
                //----------
                odFolder.setSortLocation(ODConstant.OD_SORT_LOCATION_MIDTIER);
                displayFolderSortOrder(odFolder);
                System.out.println("With Mid-tier sort enabled: USING Default Folder sort
settings");
                System.out.println("Sort Location = " + odFolder.getSortLocation());
                System.out.println("  Searching folder with SORT=M and default criteria...");
                searchAndListHits(odFolder);
                //----------
                // Create a new sort order and display it
                //----------
                System.out.println("Setting specified sort order...");

System.out.println("-----------------------------------------------------------------");
                for (crit_enum = odFolder.getCriteria(); crit_enum.hasMoreElements();)
                {
                    odCrit = (ODCriteria) crit_enum.nextElement();
                    odCrit.setSortOrder(0);
                }
                if (!(crit1.equals("-")))
                {
                    odCrit = odFolder.getCriteria(crit1);
                    odCrit.setSortOrder(1);
                    odCrit.setAscending(ascend1);
                }
                if (!(crit2.equals("-")))
```

```
                {
                    odCrit = odFolder.getCriteria(crit2);
                    odCrit.setSortOrder(2);
                    odCrit.setAscending(ascend2);
                }
                if (!(crit3.equals("-")))
                {
                    odCrit = odFolder.getCriteria(crit3);
                    odCrit.setSortOrder(3);
                    odCrit.setAscending(ascend3);
                }
                displayFolderSortOrder(odFolder);


                //----------
                // Search the folder after disabling sort
                //----------
                odFolder.setSortLocation(ODConstant.OD_SORT_LOCATION_NONE);
                System.out.println("With sort disabled:");
                System.out.println("  Searching folder with setSortLocation(NONE) ...");
                System.out.println("Sort Location = " + odFolder.getSortLocation());
                searchAndListHits(odFolder);
                //----------
                // Search the folder after enabling sort
                //----------
                odFolder.setSortLocation(ODConstant.OD_SORT_LOCATION_MIDTIER);
                displayFolderSortOrder(odFolder);
                System.out.println("  Searching folder with setSortLocation(MIDTIER) and
                                    CmdLine sort order (default criteria)...");
                System.out.println("Sort Location = " + odFolder.getSortLocation());
                searchAndListHits(odFolder);
                //----------
                // Search the folder after enabling Server sort
                //----------
                odFolder.setSortLocation(ODConstant.OD_SORT_LOCATION_SERVER);
                displayFolderSortOrder(odFolder);
                System.out.println("  Searching folder with setSortLocation(SERVER) and
                                    CmdLine sort order (default criteria)...");
                System.out.println("Sort Location = " + odFolder.getSortLocation());
                searchAndListHits(odFolder);
                //----------
                // Cleanup
                //----------
                odFolder.close();
                odServer.logoff();
                odServer.terminate();
                System.out.println("");
                System.out.println("-----------------------------------------------------");
                System.out.println("");
                System.out.println( "Testcase TcSortedHitlist completed - Ensure that:" );
                System.out.println("  1. The default sort order is the same as that shown by
the");
                System.out.println("     Windows Client");
                System.out.println("  2. The 1st hitlist is not sorted");
                System.out.println("  3. The 2nd hitlist is sorted according to the default
order");
                System.out.println("  4. The new sort order is correct for the command
line");
                System.out.println("  5. The 3rd hitlist is not sorted");
                System.out.println("  6. The 4th hitlist is sorted according to the
specified order");
                System.out.println("");
            }
    }
    catch ( ODException e )
    {
            System.out.println("ODException: " + e);
            System.out.println("   id = " + e.getErrorId());
            System.out.println("  msg = " + e.getErrorMsg());
            e.printStackTrace();
    }
```

```
  catch ( Exception e2 )
  {
          System.out.println("exception: " + e2);
          e2.printStackTrace();
  }
}

static void searchAndListHits( ODFolder odFolder )
{
      ODHit odHit;
      Vector hits;
      String[] display_crit;
      String value;
      int j, k;
  try
  {
    System.out.println( "  Searching folder with default criteria..." );
          hits = odFolder.search();
          System.out.println("    Number of hits: " + hits.size());
    if ( hits != null && hits.size( ) > 0 )
    {
              display_crit = odFolder.getDisplayOrder();
              value = "      ";
              for (j = 0; j < display_crit.length; j++)
                  value = value + display_crit[j] + "--";
              System.out.println(value);
      for ( j = 0; j < hits.size( ); j++ )
      {
                  odHit = (ODHit) hits.elementAt(j);
                  value = "      ";
                  for (k = 0; k < display_crit.length; k++)
                      value = value + odHit.getDisplayValue(display_crit[k]) + "--";
                  System.out.println(value);
      }
    }
  }

  catch ( ODException e )
  {
          System.out.println("ODException: " + e);
          System.out.println("   id = " + e.getErrorId());
          System.out.println("  msg = " + e.getErrorMsg());
          e.printStackTrace();
  }

  catch ( Exception e2 )
  {
          System.out.println("exception: " + e2);
          e2.printStackTrace();
  }
}
static void displayFolderSortOrder( ODFolder odFolder )
{
      ODCriteria odCrit;
      ArrayList sort_order;
      int j, order;
  try
  {
          sort_order = odFolder.getCriteriaSortOrder();
          System.out.println("Sort Order:");
    for ( order = -1; order < 100; order++ )
    {
      for( j = 0; j < sort_order.size( ); j++ )
      {
                  odCrit = (ODCriteria) sort_order.get(j);
                  if (odCrit.getSortOrder() == order)
                      System.out.println("  "
                              + order
                              + ". "
                              + odCrit.getName()
                              + (order <= 0 ? "" : " ("
                                  + (odCrit.getAscending() ? "ascending" : "descending")
```

```
                                        + ")"));
          }
        }
      }
      catch ( Exception e2 )
      {
              System.out.println("exception: " + e2);
              e2.printStackTrace();
      }
    }
}
```

Retrieving a document

You use methods from the ODServer, ODFolder, and ODHit classes to help you retrieve a specific document from a folder.

About this task

The example shows the following tasks:

- Logging on to the specified server
- Opening the specified folder
- Searching the folder by using the default criteria
- Displaying the number of hits
- Retrieving the data for the first hit by using `ODHit.getDocument (filename, true)`, which causes ODWEK to write all segments of the document to the specified file
- Retrieving the document again by using `ODHit.getDocument()`, which outputs the document data to a byte array
- Verifying whether the document is AFP, if so, calling `ODHit.getResource()` to retrieve the AFP resource file

The following example demonstrates three different methods of retrieving a document:

- ODServer
- ODFolder
- ODHit

This example demonstrates the following ODServer methods:

- initialize
- logon
- openFolder
- retrieve
- logoff
- terminate

This example demonstrates the following ODFolder methods:

- search
- retrieve
- close

This example demonstrates the following ODHit methods:

- getDocId
- retrieve

This example uses the following runtime parameters:

- Server name
- User Id
- Password
- Folder name

Important: If you extract and reload the same document, the system generates a new DocId for the document, and the old DocId does not reference the new document.

Example of retrieving a document:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcRetrieve
{
    public static void main ( String argv[] )
    {
        ODServer odServer;
        ODFolder odFolder;
        ODHit odHit;
        TcCallback callback;
        Vector hits;
        Vector hit_to_retrieve;
        byte[] data_from_hit;
        byte[] data_from_server;
        byte[] res_from_server;
        byte[] data_from_folder;
        int j;
        int getDocResSize =0;

        //----------
        // If too few parameters, display syntax and get out
        //----------
        if ( argv.length < 5 )
        {
            System.out.println( "usage: java TcRetrieve <server> <port> <userid> <password>
<folder>" );
            return;
        }

        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "Testcase TcRetrieve started." );
            System.out.println( "This testcase should:" );
            System.out.println( "  Logon to the specified server" );
            System.out.println( "  Open the specified folder" );
            System.out.println( "  Search the folder using the default criteria" );
            System.out.println( "  Display the number of hits" );
            System.out.println( "  Retrieve the data for the first hit using
ODHit.retrieve" );
            System.out.println( "  Retrieve the data for the first hit using
ODFolder.retrieve" );
            System.out.println( "  Display length of data retrieved from each method" );
            System.out.println( "  Compare the lengths and data retrieved from each
method" );
            System.out.println( "  Display the result of the comparisons" );
            System.out.println( "" );
            System.out.println( "-------------------------------------------------" );
            System.out.println( "" );

            //----------
            // Logon to specified server
            //----------
            ODConfig odConfig = new ODConfig();
```

```
            if(odConfig != null)
            {
                odServer = new ODServer(odConfig );
                odServer.initialize(  "TcRetrieve.java" );
                odServer.setPort(Integer.parseInt(argv[1]));

                System.out.println( "Logging on to " + server  + " server with user " +
userid + "..." );
                odServer.logon( argv[0], argv[2], argv[3]);

                //----------
                // Open the specified folder and search with the default criteria
                //----------
                System.out.println( "Opening " + argv[4] + " folder..." );
                odServer.cancel();
                odFolder = odServer.openFolder( argv[4] );
                System.out.println( "Searching with default criteria..." );
                hits = odFolder.search( );
                System.out.println( "Number of hits: " + hits.size( ) );

                //----------
                // Do some retrieves and comparisons
                //----------
                if ( hits.size( ) > 0 )
                {

                    odHit = (ODHit)hits.elementAt( 0 );
                    hit_to_retrieve = new Vector( );
                    hit_to_retrieve.addElement( odHit );

                    System.out.println( " View Mime Type = " + odHit.getViewMimeType());
                    System.out.println( " Doc Type = " + odHit.getDocType());


                    System.out.println( "Retrieving data from first hit using
ODHit.retrieve..." );
                    data_from_hit = odHit.retrieve(ODConstant.NATIVE);
                    FileOutputStream fos1 = new FileOutputStream("1hitR.out");
                    fos1.write(data_from_hit);
                    fos1.close();

                    System.out.println( "Retrieving data from first hit using
ODHit.getDocument/getResources..." );
                    data_from_server = odHit.getDocument();
                    FileOutputStream fos2 = new FileOutputStream("getDoc.out");
                    fos2.write(data_from_server);
                    fos2.close();
                    if(odHit.getDocType() == ODConstant.FileTypeAFP)
                    {
                       try{
                       res_from_server = odHit.getResources();
                       FileOutputStream fos8 = new FileOutputStream("getRes.afp");
                       fos8.write(res_from_server);
                       fos8.close();
                       getDocResSize = data_from_server.length + res_from_server.length;
                       }
                       catch(ODException E)
                       {
                         System.out.println("There are no resources for this document.");
                         getDocResSize = data_from_server.length;
                       }

                    }
                    else
                        getDocResSize = data_from_server.length ;


                    System.out.println( "Retrieving data from first hit using
ODFolder.retrieve (uses callback method)..." );
                    callback = new TcCallback( );
                    odFolder.retrieve( hit_to_retrieve, callback );
                    data_from_folder = callback.getData( );
```

```
                    if ( data_from_folder == null )
                    {
                        data_from_folder = new byte[0];
                        System.out.println( "Callback function not called during
ODFolder.retrieve" );
                    }
                    else
                    {
                        FileOutputStream fos = new FileOutputStream("3folderR.out");
                        fos.write(data_from_folder);
                        fos.close();
                    }
                    System.out.println( "Length of data from:" );
                    System.out.println( "  ODHit.retrieve=" + data_from_hit.length );
                    System.out.println( "  ODHit.getDocument (+getResource) =" +
getDocResSize );
                    System.out.println( "  ODFolder.retrieve=" + data_from_folder.length );
                    if ( data_from_hit.length == getDocResSize  )
                    {
                            System.out.println( "ODHit vs. ODHit.getDoc: Length and content
of data match" );
                            if ( data_from_hit.length == data_from_folder.length )
                            {
                                for ( j = 0; j < data_from_folder.length; j++ )
                                {
                                    if ( data_from_hit[j] != data_from_folder[j] )
                                        break;
                                }
                                if ( j == data_from_folder.length )
                                    System.out.println( "ODHit vs. ODFolder: Length and
content of data match" );
                                else
                                {
                                    System.out.println( "*** ODHit vs. ODFolder: Data
mismatch at offset " + j );
                                    System.out.println( "  ODHit    data is " +
data_from_hit[j] );
                                    System.out.println( "  ODFolder data is " +
data_from_folder[j] );
                                }
                            }
                            else
                                System.out.println( "*** ODHit vs. ODFolder: Length
mismatch" );
                    }
                    else
                        System.out.println( "*** ODHit vs. ODHit.getDoc: Length mismatch" );
                }
                else
                    System.out.println( "There is no document to retrieve" );

                //----------
                // Cleanup
                //----------
                odFolder.close( );
                odServer.logoff( );
                odServer.terminate( );
                System.out.println( "" );
                System.out.println( "--------------------------------------------------" );
                System.out.println( "" );
                System.out.println( "Testcase TcRetrieve completed - analyze the result of
the comparisons" );
                System.out.println( "" );
                System.out.println( "If the arswww.props file specifies 'native' for the
viewer type, all" );
                System.out.println( "lengths and data should match; otherwise, differences
are expected." );
                System.out.println( "" );
            }
        }
```

```
        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "   id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}
```

The following example uses ODCallback methods for bulk retrieval of document data.

```
//
*******************************************************************************************
*****
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcCallback extends ODCallback
{
  byte[ ] data_from_folder;
  boolean init = true;

  TcCallback( )
  {
  }

  public void HitHandleCallback( int hit, int off, int len )
  {
  }

  public boolean HitCallback( String docid, char type, String[ ] values )
                  throws Exception
  {
    return true;
  }

  public boolean DataCallback( byte[ ] data )
  {
    byte[ ] temp;
    int j, k;

    //----------
    // If first data block received, initialize container; otherwise,
    // append new data to that previously received.
    //----------
    if ( init )
    {
      data_from_folder = data;
      init = false;
    }
    else
    {
      temp = new byte[ data_from_folder.length + data.length ];
      for ( j = 0; j < data_from_folder.length; j++ )
        temp[j] = data_from_folder[j];
      k = data_from_folder.length;
      for ( j = 0; j < data.length; j++ )
        temp[k++] = data[j];
      data_from_folder = temp;
    }

    return true;
  }
```

```
  public byte[ ] getData( )
  {
    return data_from_folder;
  }
}
```

Printing a document

You use methods from the ODServer and ODFolder classes to help you find a printer and print a document to that server.

About this task

The following example uses ODServer and ODFolder methods to list the printers that are available on the server and to print a document to the specified server printer. This example also uses ODServer methods to prepare for logon, open the specified folder, and log off.

This example demonstrates the following ODServer methods:

- initialize
- logon
- openFolder
- getServerPrinters
- logoff
- terminate

This example demonstrates the following ODFolder methods:

- search
- printDocuments
- close

This example uses the following runtime parameters:

- Server name
- Port
- User Id
- Password
- Folder name
- Printer name

Example of printing a document:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcPrintHit
{
    public static void main ( String argv[] )
    {
        ODServer odServer;
        ODFolder odFolder;
        ODHit odHit;
        Vector hits;
        Vector hit_to_print;
        String [] printers;
        String printer_name;
        boolean match;
        int j;
```

```
        //----------
        // If too few parameters, display syntax and get out
        //----------
      if ( argv.length < 6 )
        {
            System.out.println( "usage: java TcPrintHit <server> <port> <userid> <password>
<folder> <printer>" );
            return;
        }

        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "Testcase TcPrintHit started." );
            System.out.println( "This testcase should:" );
            System.out.println( "  Logon to the specified server" );
            System.out.println( "  Display the list of printers available on the server" );
            System.out.println( "  Open the specified folder" );
            System.out.println( "  Search the folder using the default criteria" );
            System.out.println( "  Display the number of hits" );
            System.out.println( "  Print the first hit to the specified server printer" );
            System.out.println( "" );
            System.out.println( "------------------------------------------------------" );
            System.out.println( "" );

            //----------
            // Logon to specified server
            //----------
            ODConfig odConfig = new ODConfig();
            if(odConfig != null)
            {
                odServer = new ODServer(odConfig );
                odServer.initialize(  "TcPrintHit.java" );
                odServer.setPort(Integer.parseInt(argv[1]));

                System.out.println( "Logging on to " + argv[0]  + " server with user " +
argv[2] + "..." );
                odServer.logon( argv[0], argv[2], argv[3]);

                //----------
                // If any server printers are available on the server
                //----------
                System.out.println( "Retrieving list of server printers..." );
                printer_name = argv[5];
                printers = odServer.getServerPrinters( );
                if ( printers.length > 0 )
                {
                    //----------
                    // List the available server printers
                    //----------
                    System.out.println( "There are " + printers.length + " printers
available on the server:" );
                    match = false;
                    for( j = 0; j < printers.length; j++ )
                    {
                        System.out.println( "  " + printers[j] );
                        if ( printers[j].equals( printer_name ) )
                            match = true;
                    }

                    if ( match )
                    {
                        //----------
                        // Open the specified folder and search with the default criteria
                        //----------
                        System.out.println( "Opening " + argv[4] + " folder..." );
                        odFolder = odServer.openFolder( argv[4] );
                        System.out.println( "Searching with default criteria..." );
                        hits = odFolder.search( );
                        System.out.println( "  Number of hits: " + hits.size( ) );
```

```
                        //----------
                        // Print the first hit to the specified server printer
                        //----------
                        if ( hits.size( ) > 0 )
                        {
                            hit_to_print = new Vector( );
                            odHit = (ODHit)hits.elementAt( 0 );
                            hit_to_print.addElement( odHit );
                            System.out.println( "Printing first hit to " + printer_name +
"..." );
                            odFolder.printDocuments( hit_to_print, printer_name,2);
                        }
                        else
                            System.out.println( "There is no document to print" );

                        odFolder.close( );
                    }
                    else
                        System.out.println( "The specified printer (" + printer_name +
                                            ") is not available on this server" );
                }
                else
                    System.out.println( "No printers are avilable on this server" );

                //----------
                // Cleanup
                //----------
                odServer.logoff( );
                odServer.terminate( );
                System.out.println( "" );
                System.out.println( "-----------------------------------------------------" );
                System.out.println( "" );
                System.out.println( "Testcase TcPrintHit completed - Analyze the results" );
                System.out.println( "" );
            }
        }

        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "    id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}
```

When you click the Print icon in the web browser toolbar while viewing a line data report from a folder delivered by ODWEK, you are prompted to select a printer. Some users are unsure how to print a specific range of pages, versus printing all pages.

To be able to print a range of pages, click OK on the first print window, where the printer name and properties are specified. After you click OK, a second print window displays, where you can specify margins and the range of pages to print. This print page-range behavior is under control of a Java API that is started by ODWEK.

Listing information about annotations

You use methods from the ODNote class to display information about the annotations (also called notes) in a folder.

About this task

The following example uses ODNote methods to list detailed information about an annotation (also called notes). The following example does the following tasks:

- Logs on to the specified server
- Opens the specified folder
- Searches the folder using the default criteria
- Displays the number of hits
- Displays the number of annotations associated with the first document
- Displays detailed information for each annotation that is attached to the document

The methods list the following information about the annotation:

- The position of the annotation on the page of the document
- The background color
- Date and time that the annotation was attached to the document
- The user ID that created the annotation
- Other attributes

This example demonstrates the following ODNote methods:

- getColor
- getDateTime
- getGroupName
- getOffsetX
- getOffsetY
- getPageNum
- getText
- getUserid
- isOkToCopy
- isPublic

This example also demonstrates these ODServer methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example also demonstrates these ODFolder methods:

- search
- close

This example also demonstrates these ODHit methods:

- getNotes

- getNoteStatus

This example uses the following runtime parameters:

- Server name
- Port
- User Id
- Password
- Folder name

Example of listing information about annotations:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcListNotes
{
  public static void main ( String argv[] )
  {
        ODServer odServer;
        ODFolder odFolder;
        ODHit odHit;
        ODNote odNote;
        Vector hits, notes;
        String str;
        int j, exist;
        //----------
        // If too few parameters, display syntax and get out
        //----------
        if ( argv.length < 5 )
        {
              System.out.println( "usage: java TcListNotes <server> <port> <userid> <password>
<folder>" );
              return;
        }
        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println("Testcase TcListNotes started." );
            System.out.println("This testcase should:");
            System.out.println("  Logon to the specified server");
            System.out.println("  Open the specified folder");
            System.out.println("  Search the folder using the default criteria");
            System.out.println("  Display the number of hits");
            System.out.println("  Display the note status for the first hit");
            System.out.println("  Display the number of notes associated with the first hit");
            System.out.println("  Display info for each note");
            System.out.println("");
            System.out.println("-----------------------------------------------------");
            System.out.println("");
            //----------
            // Logon to specified server
            //----------
            ODConfig odConfig = new ODConfig();
            if (odConfig != null)
            {
                odServer = new ODServer(odConfig);
                odServer.initialize("TcListNotes.java");
                odServer.setPort(Integer.parseInt(argv[1]));

                System.out.println( "Logging on to " + argv[0]  + " server with user " + argv[2] +
"..." );
                odServer.logon( argv[0], argv[2], argv[3]);
                //----------
                // Open the specified folder and search with the default criteria
                //----------
                System.out.println( "Opening " + argv[4] + " folder..." );
                odFolder = odServer.openFolder( argv[4] );
                System.out.println("Searching with default criteria...");
                hits = odFolder.search();
                System.out.println("  Number of hits: " + hits.size());
                //----------
                // List info for each note for the first hit
                //----------
```

```
                if ( hits.size( ) > 0 )
                {
                    odHit = (ODHit) hits.elementAt(0);
                    System.out.println("  For the first hit:");
                    System.out.println("    Note status is: "
                            + odHit.getNoteStatus());
                    notes = odHit.getNotes();
                    if (notes.size() > 0)
                        System.out.println("    There is(are) " + notes.size() + " note(s)");
                    for ( j = 0; j < notes.size( ); j++ )
                    {
                        odNote = (ODNote) notes.elementAt(j);
                        System.out.println("      " + (j+1) + ". Text='" + odNote.getText( ) + "'" );
                        System.out.println("         UserId=" + odNote.getUserId());
                        System.out.println("         Page=" + odNote.getPageNum());
                        System.out.println("         Color=" + odNote.getColor());
                        System.out.println("         Date=" + odNote.getDateTime());
                        System.out.println("         Group=" + odNote.getGroupName());
                        System.out.println("         Offset=(" + odNote.getOffsetX( ) + "," +
                                                      odNote.getOffsetY( ) + ")" );
                        System.out.println("         OkToCopy=" + odNote.isOkToCopy());
                        System.out.println("         Public=" + odNote.isPublic());
                        System.out.println("         Text=" + odNote.getText());
                    }
                }
                else
                    System.out.println("There is no document - cannot list notes");
                //----------
                // Cleanup
                //----------
                odFolder.close();
                odServer.logoff();
                odServer.terminate();
                System.out.println("");
                System.out.println("--------------------------------------------------");
                System.out.println("");
                System.out.println( "Testcase TcListNotes completed - Ensure that the information" );
                System.out.println("  is the same as shown by the Windows Client");
                System.out.println("");
            }
        }
        catch ( ODException e )
        {
          System.out.println( "ODException: " + e );
          System.out.println( "   id = " + e.getErrorId( ) );
          System.out.println( "  msg = " + e.getErrorMsg( ) );
          e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
          System.out.println( "exception: " + e2 );
          e2.printStackTrace( );
        }
  }

  static String getExistenceName( int code )
  {
    String str;

    switch( code )
    {
        case ODConstant.NoteStatusUnknown:
            str = "UNKNOWN";
            break;
        case ODConstant.NoteStatusYes:
            str = "YES";
            break;
        case ODConstant.NoteStatusNo:
            str = "NO";
            break;
        case ODConstant.NoteStatusError:
            str = "ERROR";
            break;
        default:
            str = "UNDEFINED VALUE";
    }
        return str;
  }
}
```

Adding an annotation

You use methods from the ODHit class to add or retrieve an annotation (also called a note) from a folder.

About this task

An object of the class ODHit represents a Content Manager OnDemand document. The following example uses ODHit methods to display the number of annotations associated with the document and add an annotation with these attributes:

- The specified annotation text
- OkToCopy=false
- Public=false (that is, a private annotation)
- An empty group name

This example demonstrates the following ODHit methods:

- getNotes
- addNote

This example uses ODServer methods to do the following tasks:

- Prepare for logon
- Open the specified folder
- Log off

The example uses the ODFolder methods to search the folder, get the number of hits that matched the query, and close the folder.

This example demonstrates the following ODServer methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates the following ODFolder methods:

- search
- getDocId
- getHits
- close

This example demonstrates the following ODNote methods:

- getGroupName
- getText
- isOkToCopy
- isPublic
- setGroupName
- setOkToCopy
- setPublic
- setText

This example uses the following runtime parameters:

- Server name
- Port
- User Id
- Password
- Folder name
- Text of annotation

Example of adding an annotation:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcAddNote
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODHit odHit;
    ODNote odNote;
    Vector hits;
    Vector notes;
    int j;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 6 )
    {
      System.out.println( "usage: java TcAddNote <server> <port> <userid> <password> <folder> <note
text>" );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "Testcase TcAddNote started." );
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Search the folder using the default criteria" );
      System.out.println( "  Display the number of hits" );
      System.out.println( "  Display the number of notes associated with the first hit" );
      System.out.println( "  Add a new note with the these attributes" );
      System.out.println( "    The specified note text" );
      System.out.println( "    OkToCopy=false" );
      System.out.println( "    Public=false (i.e. a private note)" );
      System.out.println( "    An empty group name" );
      System.out.println( "" );
      System.out.println( "-----------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to specified server
      //----------

      ODConfig odConfig = new ODConfig();

      if (odConfig != null)
      {
        odServer = new ODServer(odConfig );
        odServer.initialize( "TcAddNote.java" );
        odServer.setPort(Integer.parseInt(argv[1]));
        System.out.println( "Logging on to " + argv[0] + " server with user " + argv[2] + "..." );
        odServer.logon( argv[0], argv[2], argv[3]);
        //----------
        // Open the specified folder and search with the default criteria
        //----------
        System.out.println( "Opening " + argv[4] + " folder..." );
        odFolder = odServer.openFolder( argv[4] );
        System.out.println( "Searching with default criteria..." );
        odFolder.search( );
```

```
        hits = odFolder.getHits( );
        System.out.println( "  Number of hits: " + hits.size( ) );

        //----------
        // Add a new note
        //----------
        if ( hits.size( ) > 0 )
        {
          odHit = (ODHit)hits.elementAt( 0 );
          System.out.println("Working with DocID " + odHit.getDocId());
          notes = odHit.getNotes( );
          if(notes.size() > 0)
          System.out.println("  There are " + notes.size( ) + " notes for the first hit" );

          odNote = new ODNote( );
          odNote.setText( argv[5] );
          odNote.setGroupName( "" );
          odNote.setOkToCopy( false );
          odNote.setPublic( false );

          System.out.println("  Adding a new note with:" );
          System.out.println("    Text='" + odNote.getText( ) + "'" );
          System.out.println("      UserId=" + argv[2] );
          System.out.println("      Page=1" );
          System.out.println("      Color=" + 'Y' );
          System.out.println("      Group=" + odNote.getGroupName( ) );
          System.out.println("      Offset=(0,0)" );
          System.out.println("    OkToCopy=" + odNote.isOkToCopy( ) );
          System.out.println("    Public=" + odNote.isPublic( ) );
          System.out.println("    Group=" + odNote.getGroupName( ) );

          odHit.addNote( odNote );
        }
        else
          System.out.println( "No document - cannot list notes" );

        //----------
        // Cleanup
        //----------
        odFolder.close( );
        odServer.logoff( );
        odServer.terminate( );
        System.out.println( "" );
        System.out.println( "-----------------------------------------------------" );
        System.out.println( "" );
        System.out.println( "Testcase TcAddNote completed. - Ensure that the new note was correctly" );
        System.out.println( "  added by displaying it with the Windows Client" );
        System.out.println( "" );
      }
      else
      {
         System.out.println( "" );
         System.out.println( "Testcase TcAddNote failed  -" );
         System.out.println( "  ODConfig could not be initialized.  " );
         System.out.println( "" );
      }
    }

    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }
}
```

Deleting an annotation

You use methods from the ODHit class to delete an annotation (also called a note) from a document.

About this task

This example completes the following functions:

- Logging on to the specified server
- Opening the specified folder
- Searching the folder by using the default criteria
- Displaying the number of hits
- Displaying the number of annotations that are associated with the first hit
- Deleting the first annotation

This example demonstrates the following ODServer methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates the following `ODFolder` methods:

- search
- getHits
- close

This example demonstrates the following `ODHit` methods:

- getDocId
- getNotes
- deleteNote

This example uses the following runtime parameters:

- Server name
- Port
- User Id
- Password
- Folder name
- Text of annotation

Example of deleting an annotation:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcDeleteNote
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODHit odHit;
    ODNote odNote;
    Vector hits;
```

```
    Vector notes;
    int j, numNotes1 = 0, numNotes2 = 0;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 5 )
    {
      System.out.println( "usage: java TcDeleteNote <server> <port> <userid> <password>
<folder>" );
      return;
    }

    try
    {
      //----------
      // Set the stage
      //----------
      System.out.println( "Testcase TcDeleteNote started." );
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Search the folder using the default criteria" );
      System.out.println( "  Display the number of hits" );
      System.out.println( "  Display the number of notes associated with the first hit" );
      System.out.println( "  Delete first note" );
      System.out.println( "" );
      System.out.println( "--------------------------------------------------" );
      System.out.println( "" );

      //----------
      // Logon to specified server
      //----------

      ODConfig odConfig = new ODConfig();

      if (odConfig != null)
      {
        odServer = new ODServer(odConfig );
        odServer.initialize( "TcDeleteNote.java" );
        odServer.setPort(Integer.parseInt(argv[1]));

        System.out.println( "Logging on to " + argv[0] + " server with user " + argv[2] +
"..." );
        odServer.logon( argv[0], argv[2], argv[3]);
        //----------
        // Open the specified folder and search with the default criteria
        //----------
        System.out.println( "Opening " + argv[4] + " folder..." );
        odFolder = odServer.openFolder( argv[4] );
        System.out.println( "Searching with default criteria..." );
        odFolder.search( );
        hits = odFolder.getHits( );
        System.out.println( "  Number of hits: " + hits.size( ) );

        //----------
        // Add a new note
        //----------
        if (hits.size() > 0)
        {
            odHit = (ODHit)hits.elementAt(0);
            System.out.println("Working with DocID " + odHit.getDocId());
            notes = odHit.getNotes();
            numNotes1 = notes.size();
            System.out.println("  There are " + numNotes1 + " notes for the first hit");

            if (numNotes1 > 0)
            {
                odNote = (ODNote)notes.elementAt(0);
                System.out.println("   Delete in the first note...");
                odHit.deleteNote(odNote);
```

```
                // destroy hits
                hits.removeAllElements();

                // search again
                System.out.println("Searching second time with default criteria...");
                odFolder.search();
                hits = odFolder.getHits();
                System.out.println("  Number of hits:  " + hits.size());

                // Retrieve notes again
                if (hits.size() > 0)
                {
                    odHit = (ODHit)hits.elementAt(0);
                    System.out.println("Working with DocID " + odHit.getDocId());
                    notes = odHit.getNotes();
                    numNotes2 = notes.size();
                    System.out.println("  There are " + numNotes2 + " notes for the first
hit");
                }
                else
                    System.out.println("No document - cannot list notes");
            }
            else
                System.out.println("No document - cannot list notes");

            if ((numNotes1 - numNotes2) == 1)
                System.out.println("\nSuccess!");
            else if (numNotes1 == 0)
                System.out.println("\nSuccess!");
            else
                System.out.println("\nFailed!");
        }
        else
            System.out.println("\nNo hits found");

        //----------
        // Cleanup
        //----------
        odFolder.close( );
        odServer.logoff( );
        odServer.terminate( );
        System.out.println( "" );
        System.out.println( "--------------------------------------------------" );
        System.out.println( "" );
        System.out.println( "Testcase TcDeleteNote completed. - Ensure that the note was
deleted" );
        System.out.println( "  by displaying document notes with the Windows Client" );
        System.out.println( "" );
      }
      else
      {
       System.out.println( "" );
       System.out.println( "Testcase TcDeleteNote failed  -" );
       System.out.println( "  ODConfig could not be initialized.  " );
       System.out.println( "  Probable cause: " );
       System.out.println( "    File arswww.props is not found in directory " +  argv[5] );
       System.out.println( "" );
      }
    }

    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
```

```
    }
  }
}
```

Updating a document

You use methods from the ODServer, ODFolder, ODHit, and ODCriteria classes to find specific documents in a folder, retrieve those documents, and update database values.

About this task

The following example demonstrates how to update a document.

This example uses ODServer, ODFolder, and ODCriteria methods to do the following tasks:

- connect to a server using the specified user ID and password
- Open the specified folder
- Set the search values for two search fields
- Set the Date search field to null
- Search the folder

For the document that matches the query, ODHit methods are then used to update one or more database values.

This example demonstrates the following ODServer methods:

- initialize
- logon
- openFolder
- logoff
- terminate

This example demonstrates the following ODFolder methods:

- getDisplayOrder
- getCriteria
- search
- close

This example demonstrates the following ODCriteria methods:

- setOperator

This example demonstrates the following ODHit methods:

- getDisplayValue
- getDocId
- updateValuesForHit

This example uses the following runtime parameters:

- Server name
- Port
- User Id
- Password
- Folder name
- Criteria name 1

- Search value 1
- Criteria name 2
- Search value 2
- New search value to replace search value 2

Example of updating a document:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcUpdate
{
  public static void main ( String argv[] )
  {
    ODServer odServer;
    ODFolder odFolder;
    ODCriteria odCrit, odCrit2;
    String criteria1;
    String criteria2;
    String search_value1;
    String search_value2;
    String replacement_value;

    //----------
    // If too few parameters, display syntax and get out
    //----------
    if ( argv.length < 9 )
    {
            System.out.println( "usage: java TcUpdate <server> <port> <userid> <password>
<folder>
                                    <criteria1> <value1> <criteria2> <value2> <new value2>" );
      return;
    }

    try
    {
      System.out.println( "Testcase TcUpdate started." );
      System.out.println( "This testcase should:" );
      System.out.println( "  Logon to the specified server" );
      System.out.println( "  Open the specified folder" );
      System.out.println( "  Set the search values" );
      System.out.println( "  Search the folder" );
      System.out.println( "  For the first hit, change the value of criteria 2 to the new
value" );
      System.out.println( "  Set the search values" );
      System.out.println( "  Change the value of criteria 2 to the original value" );
      System.out.println( "" );
      System.out.println( "------------------------------------------------------" );
      System.out.println( "" );


      //----------
      //Setup ODConfig object using defaults and initialize ODServer.
      //----------
      ODConfig odConfig = new ODConfig();
      if(odConfig != null)
      {
         odServer = new ODServer(odConfig);
         odServer.initialize( "TcUpdate.java");
         odServer.setPort(Integer.parseInt(argv[1]));

         System.out.println( "Logging on to " + argv[0]  + " server with user " + argv[2] +
"..." );
         odServer.logon( argv[0], argv[2], argv[3]);

         //----------
         // Open the specified folder and set the requested criteria
         //----------
         criteria1 = argv[5];
```

```
         criteria2 = argv[7];
         search_value1 = argv[6];
         search_value2 = argv[8];
         replacement_value = argv[9];
         System.out.println( "Opening " + argv[4] + " folder..." );
         odFolder = odServer.openFolder( argv[4] );

         ArrayList allCrit =odFolder.getCriteriaSortOrder();
         /*Clear all default search values*/
         for (int a =0; a < allCrit.size(); a++)
         {
             odCrit = (ODCriteria)allCrit.get(a);
             odCrit.setSearchValues("","");
         }
         odCrit = odFolder.getCriteria( criteria1 );
         if(odCrit == null)
            System.out.println("ERROR: Search Criteria Not Found [" + criteria1 + "]");
         else
         {
            odCrit.setOperator( ODConstant.OPEqual );
            odCrit.setSearchValue( search_value1 );
         }
         odCrit2 = odFolder.getCriteria( criteria2 );
         if(odCrit2 == null)
           System.out.println("ERROR: Search Criteria Not Found [" + criteria2 + "]");
         else
         {
            odCrit2.setOperator( ODConstant.OPEqual );
            odCrit2.setSearchValue( search_value2 );
         }
         //----------
         // Search the folder
         //----------
         if (odCrit != null && odCrit2 != null)
         {
                System.out.println( "Change " + criteria2 + " from original value " +
search_value2 +
                                             " to " + replacement_value  );

                UpdateHits(odFolder, criteria1, search_value1, criteria2, search_value2,
replacement_value);

                //
                //----------
                // Change values back to original
                //----------
                System.out.println( "Change " + criteria2 + " from new value " +
replacement_value +
                                             " to "  + search_value2  );
                odCrit2.setSearchValue(replacement_value);

                UpdateHits(odFolder, criteria1, search_value1, criteria2, replacement_value,
search_value2);

         }
         //----------
         // Cleanup
         //----------
         odFolder.close( );
         odServer.logoff( );
         odServer.terminate( );
         System.out.println( "" );
         System.out.println( "-------------------------------------------------" );
         System.out.println( "" );
         System.out.println( "Testcase TcUpdate completed," );
         System.out.println( "" );
      }
    }

    catch ( ODException e )
    {
      System.out.println( "ODException: " + e );
```

```
      System.out.println( "   id = " + e.getErrorId( ) );
      System.out.println( "  msg = " + e.getErrorMsg( ) );
      e.printStackTrace( );
    }

    catch ( Exception e2 )
    {
      System.out.println( "exception: " + e2 );
      e2.printStackTrace( );
    }
  }

  public static void UpdateHits(ODFolder odFolder1, String crit1, String value1, String
crit2,
                                String value2, String new_value)
  {
      ODHit odHit;
      Hashtable hash;
      Vector hits;
      String[] display_crit;
      String line;
      int j;

      try
      {
             System.out.println( "  Searching for " + crit1 + " = " + value1 + " and " +
crit2 + " = " + value2 + "..." );
             hits = odFolder1.search( );

            //----------
            // If there was at least one hit
            //----------
            if ( hits != null && hits.size( ) > 0 )
            {
              //----------
              // Display the values for the first hit
              //----------
              System.out.println( "    For first hit:" );
              odHit = (ODHit)hits.elementAt( 0 );
              System.out.println( "      DOCID = "+ odHit.getDocId() );
              System.out.println("      Application Group Name = " + odHit.getApplGrpName());
              line = "      ";
              display_crit = odFolder1.getDisplayOrder( );
              for( j = 0; j < display_crit.length; j++ )
                line = line + display_crit[j] + " ";
              System.out.println( line );
              line = "      ";
              for ( j = 0; j < display_crit.length; j++ )
                line = line + odHit.getDisplayValue( display_crit[j] ) + " ";
              System.out.println( line );

              //----------
              // Create a hash table of existing critera/value pairs, except for critera 2
               // which will be set to the new value. Update the hit values
              //----------
              System.out.println( "    Replacing " + crit2 + " = " + value2 + " with " +
crit2 + " = " + new_value );
              hash = new Hashtable( );
              for ( j = 0; j < display_crit.length; j++ )
              {
                if ( display_crit[j].equals( crit2 ) )
                     hash.put( display_crit[j], new_value );
                  else
                  hash.put( display_crit[j], odHit.getDisplayValue( display_crit[j] ) );
              }
              odHit.updateValuesForHit( hash );
              System.out.println("New display values as set in ODHit");
              odHit = (ODHit)hits.elementAt( 0 );
              line = "      ";
              System.out.println( "      DOCID = "+ odHit.getDocId() );
              display_crit = odFolder1.getDisplayOrder( );
              for( j = 0; j < display_crit.length; j++ )
```

```
                line = line + display_crit[j] + " ";
              System.out.println( line );
              line = "      ";
              for ( j = 0; j < display_crit.length; j++ )
                  line = line + odHit.getDisplayValue( display_crit[j] ) + " ";
              System.out.println( line );
            }
             else
              System.out.println( "There were no hits" );
      }

      catch ( ODException e )
      {
        System.out.println( "ODException: " + e );
        System.out.println( "    id = " + e.getErrorId( ) );
        System.out.println( "  msg = " + e.getErrorMsg( ) );
        e.printStackTrace( );
      }

      catch ( Exception e2 )
      {
        System.out.println( "exception: " + e2 );
        e2.printStackTrace( );
      }
  }
}
```

Changing a password

Use methods from the ODServer class to change the password of a Content Manager OnDemand user.

About this task

The following example uses the ODServer method changePassword to change the specified user's password to a new password. This example also uses ODServer methods to prepare for logon and log off.

This example demonstrates the following ODServer methods:

- initialize
- logoff
- logon
- changePassword
- getPassword
- setPassword
- terminate
- setServerName
- setUserId

This example uses the following runtime parameters:

- First logon
- New password
- Password
- Port
- Server name
- User Id

Example of changing a password:

```
import java.util.*;
import java.io.*;
import com.ibm.edms.od.*;

public class TcChangePassword
{
    public static void main ( String argv[] )
    {
        ODServer odServer;
        String server, userid, original_password, new_password;

        //----------
        // If too few parameters, display syntax and get out
        //----------
      if ( argv.length < 6 )
         {
            System.out.println( "usage: java TcChangePassword <server> <port> <userid>
<password>
                                          <new password> <initial logon>" );
            return;
        }

        try
        {
            //----------
            // Set the stage
            //----------
            System.out.println( "Testcase TcChangePassword started." );
            System.out.println( "This test case should:" );
            System.out.println( "  Logon to the server using the specified password" );
            System.out.println( "  Change the password to the new password" );

            System.out.println( "  Logoff" );
            System.out.println( "  Logon to the server using the new password" );
            System.out.println( "  Change the password back to the original password" );
            System.out.println( "  Logoff" );
            System.out.println( "" );
            System.out.println( "If the test case executes without exception, no further
analysis" );
            System.out.println( "is required." );
            System.out.println( "" );
            System.out.println( "-----------------------------------------------------" );
            System.out.println( "" );

            //----------
            // Create the specified server
            //----------
            server = argv[0];
            userid = argv[2];
            original_password = argv[3];
            new_password = argv[4];
            int logon_first= Integer.parseInt(argv[5]);

            ODConfig odConfig = new ODConfig();
            if(odConfig != null)
            {
                odServer = new ODServer(odConfig );
                odServer.initialize( "TcChangePassword.java" );
                odServer.setPort(Integer.parseInt(argv[1]));

                if(logon_first == 1)
                {
                    //----------
                    // Logon to the server using the original password
                    //----------
                    System.out.println( "Logging on to " + server + " with user " + userid +
" using original password..." );
                    odServer.logon( server, userid, original_password);
                }
                else
                {
                    System.out.println("Do NOT logon, but set the pre-req fields of
```

```
UID,PWD,and Server");
                    odServer.setUserId(userid);
                    odServer.setPassword(original_password);
                    odServer.setServerName(server);
                }
                //----------
                // Change to the new password and logoff
                //----------
                System.out.println( "Changing to new password..." );
                odServer.changePassword( new_password );
                System.out.println( "Current ODServer.password is " +
odServer.getPassword() );
                System.out.println( "Logging off..." );
                odServer.logoff( );
                //----------
                // Logon to the server using the new password
                //----------
                System.out.println( "Logging on to " + server + " with user " + userid + "
using new password..." );
                odServer.logon( server, userid, new_password );

                //----------
                // Change back to the original password and logoff
                //----------
                System.out.println( "Changing back to original password..." );
                odServer.changePassword( original_password );
                System.out.println( "Current ODServer.password is " +
odServer.getPassword() );
                System.out.println( "Logging off..." );
                odServer.logoff( );

                //----------
                // Cleanup
                //----------
                odServer.terminate( );
                System.out.println( "" );
                System.out.println( "-------------------------------------------------------" );
                System.out.println( "" );
                System.out.println( "Testcase TcChangePassword completed successfully." );
                System.out.println( "" );
            }
        }

        catch ( ODException e )
        {
            System.out.println( "ODException: " + e );
            System.out.println( "   id = " + e.getErrorId( ) );
            System.out.println( "  msg = " + e.getErrorMsg( ) );
            e.printStackTrace( );
        }

        catch ( Exception e2 )
        {
            System.out.println( "exception: " + e2 );
            e2.printStackTrace( );
        }
    }
}
```

Chapter 5. Implementing Content Manager OnDemand REST Services

You can develop your own applications for lightweight clients to access Content Manager OnDemand with the Content Manager OnDemand REST Services.

IBM i z/OS Multiplatforms REST Services overview

The Content Manager OnDemand REST (Representational State Transfer) Services enable lightweight clients to access Content Manager OnDemand data over HTTP.

Client applications can use the Content Manager OnDemand REST Services to perform fundamental OnDemand client operations, including:

- Listing Content Manager OnDemand folders defined to the server
- Listing complete Content Manager OnDemand folder definitions
- Listing generic transform viewers currently defined to the server
- Searching for archived documents
- Retrieving archived documents in either the document's original format or a converted format that has been transformed by using the Content Manager OnDemand Generic Transform interface

For more information, see Content Manager OnDemand REST Services.

Chapter 6. Installing and configuring viewing and transform software

Viewing and transform software can help you display your documents in formats that your users can more easily read on their screens.

Installing and configuring the AFP Web Viewer

The AFP Web Viewer helps users view AFP documents. The viewer must be distributed to all users and they must install it on their workstations.

Before you begin

Review the viewer requirements to ensure you have installed the prerequisites.

To install and configure the AFP Web Viewer on your users workstations, do the following tasks:

1. If you plan to distribute user-defined files with the AFP Web Viewer, configure the AFP Web Viewer installation file to hold the user-defined files.
2. Distribute one of the following files to your users and instruct them to close their browser, run or extract these files on their workstations, and restart their browser:

   `afpplgin.exe`
   The IBM Content Manager OnDemand AFP Web Viewer for all languages that include DBCS support. This file is a self-extracting file.

   `afpplgin.zip`
   The IBM Content Manager OnDemand AFP Web Viewer in a compressed file format for all languages include DBCS support. Instruct your users to uncompress the file and then run the `setup.exe` program to install the viewer.

   You can find these files in the following subdirectories:

   - Multiplatforms in the `plugins` subdirectory where ODWEK was installed
   - z/OS `/usr/lpp/ars/V10R5M0/www/plugins`
   - IBM i `/QIBM/ProdData/OnDemand/www/plugins`

The installation process copies the viewer and its associated files to directories of the user's choice. The installation program installs the viewers as ActiveX controls for Internet Explorer. The AFP Web Viewer requires approximately 36 MB of space on the workstation. The Image Web Viewer requires approximately 2 MB of space on the workstation. Remind your users to restart their browser if it was active during the installation process.

About this task

Most customers use one of the following ways to distribute the viewer files from a server, depending on whether they plan to distribute user-defined files with the AFP Web Viewer:

- Standard Installation. This method distributes the AFP Web Viewer files supplied by IBM and prepares for distributing user-defined files with the AFP Web Viewer. When an administrator installs the ODWEK software on the Web server, the installation files for the viewers are stored in a directory on the server. There is an installation file (with the file extension of .EXE) for each viewer and a ZIP archive file for the AFP Web Viewer. The administrator typically moves the installation files to the public directory on the server and creates a web page with the links to the files. A user installs a viewer by loading the web page into their browser and activating the link to the appropriate installation file.

- Custom Installation for the AFP Web Viewer. This method distributes user-defined files with the AFP Web Viewer.

Procedure

To create a custom installation, do the following steps:

1. Set up the server for a Standard Install.
2. Before any users actually install the viewer, obtain a copy of the AFP Web Viewer ZIP archive file.
3. Extract the files from the ZIP archive file to an empty work directory.
4. Add subdirectories to the work directory and store user-defined files in the directories.
5. After all of the directories and files are configured, create a self-extracting .EXE file for distribution.
6. Replace the .EXE file provided by IBM for a Standard Install with the self-extracting .EXE file that you created.
7. After an administrator completes steps "1" on page 86 through "6" on page 86, users can install the AFP Web Viewer and the user-defined files by loading the web page into their browsers and activating the link to the updated installation file.

Supported colors

The following settings might be applied from logical views on the server to the AFP Web Viewer.

- Background Color. The following colors are supported. No other colors are supported.

  Green Bar (displayed with a white background)
  Green
  Red
  Yellow
  Black
  White
  Grey

- Image Color. The following colors are supported. No other colors are supported.

  Yellow
  Blue
  Red
  Magenta
  Green
  Cyan
  Default (should display as black)

- Zoom.

Selected Area Color is not applied to the AFP Web Viewer. The selected area is always displayed with white text and a black background.

Distributing user-defined files

You can distribute user-defined files with the IBM OnDemand AFP Web Viewer software that is supplied by IBM so that when a user views an AFP document, the document is displayed with the correct fonts.

About this task

To distribute user-defined files with the AFP Web Viewer, you must package the files into an installation file and store the installation file in a shared location. When a user runs the installation file, the Setup program automatically installs the AFP Web Viewer and the user-defined files on the user's workstation.

You can distribute the following types of user-defined files with the AFP Web Viewer:

- AFP font files. These files are copied to the FONT subdirectory of the AFP Web Viewer destination directory on the workstation.
- TrueType font files. These files are copied to the Windows FONTS directory and installed in Windows by the Setup program.
- Miscellaneous user-defined files. These files are copied to the AFP Web Viewer destination directory on the user's workstation.

Tip: The Setup program copies user-defined files to the workstation after the AFP Web Viewer files that are supplied by IBM. If you name a user-defined file the same as one of the files supplied by IBM, then the user-defined file replaces the file supplied by IBM. You can take advantage of this feature, for example, to distribute an updated FTDPORT2.INI file or to distribute IBM AFP font files that your organization modified.

Adding subdirectories
The user-defined files that you plan to distribute must be stored in the CUSTOM subdirectory tree under the main viewer installation directory. For example, you can name the main viewer installation directory \ONDEMAND\AFP32.

Procedure

To configure the main viewer installation directory to hold user-defined files, do the following steps:

1. Create a CUSTOM directory under the main viewer installation directory.
   For example: \ondemand\afp32\custom

   Tip: The CUSTOM directory can hold user-defined files (other than AFP font files and Windows TrueType font files) that you want to distribute to your users. The Setup program copies files from this directory to the AFP Web Viewer destination directory on the workstation.
2. Add the subdirectories you need to store font files in the CUSTOM directory.

   The subdirectories you add depend on the type of user-defined files that you want to distribute to your users.

   - Create a FONT subdirectory under the CUSTOM directory to hold AFP font files (file types .FNT and .MAP). For example: \ondemand\afp32\custom\font. The Setup program copies these files to the AFP Web Viewer FONT directory on the workstation.
   - Create a TRUETYPE subdirectory under the CUSTOM directory to hold Windows TrueType font files (file type .TTF). For example: \ondemand\afp32\custom\truetype. The Setup program copies files from this directory to the Windows FONTS directory and installs the fonts in Windows.

Storing user-defined files
After extracting the IBM-supplied installation files to the work directory and creating the CUSTOM directories, you can store the user-defined files in the individual subdirectories. For example, copy AFP font files (file types .FNT and .MAP) that you want to distribute to your users to the \ONDEMAND \AFP32\CUSTOM\FONT directory.

Building the AFP Web Viewer installation file
After you finish creating directories and storing files in the CUSTOM directory tree, you must create an installation file that contains your user-defined files and the AFP Web Viewer files supplied by IBM. The installation file is typically named Setup.exe.

About this task

Several companies make software for packaging files and applications into a single, self-extracting AFP Web Viewer executable file for distribution. For example, the InstallShield Software Corporation offers a product called PackageForTheWeb.

Important: Software provided by other companies is not supported by IBM.

After you obtain the packaging software, run it and follow the instructions provided to create an AFP Web Viewer installation file that contains your user-defined files and the AFP Web Viewer files supplied by IBM.

Installing the AFP Web Viewer on a user's workstation
The next time a user activates the link to the AFP Web Viewer installation file from the server, the Setup program installs the AFP Web Viewer on the user's workstation and copies all of the user-defined files that you packaged with the AFP Web Viewer installation file to the user's workstation.

Mapping AFP fonts

AFP fonts that a document was created with need to be mapped to fonts that can be displayed by the AFP Web Viewer. ODWEK provides font definition files that map the IBM Core Interchange (Latin only) and compatibility fonts to TrueType fonts. The font definition files and font map files are stored in the FONT subdirectory of the AFP Web Viewer directory.

About this task

For any of the following situations, you must define the fonts in the font definition files so that the AFP Web Viewer can correctly display the documents:

- If your documents use fonts that are not defined to the AFP Web Viewer.
- If you or others in your organization have modified the IBM Core fonts
- If you or others in your organization have created AFP fonts.

Displaying AFP reports

The FTDPORT2.INI file, which is stored in the AFP Web Viewer installation directory, contains modifiable parameters that can affect how AFP reports display.

About this task

The following list describes these parameters and their possible values.

- Rules and lines

  If rules or lines do not display correctly when you view an AFP report, the problem might be caused by differences in display software. Use another method for displaying rules. In the Misc section of the FTDPORT2.INI file, change the parameter **RuleFix** from FALSE to TRUE.

- Text fidelity

  If fonts are not substituted correctly and the text alignment is not correct (especially when the Text Fidelity parameter is set to Character), it might be because your report was created with 300–pel metrics rather than 240–pel metrics. If you specify 240Fidelity=FALSE, the report is displayed with 300–pel metrics. If you specify 240Fidelity=TRUE, the report is displayed with 240–pel metrics. The default is 240–pel metrics.

- Print dialog box default

  When the Print dialog box displays, the default is to print the current page of the report. You can change the default to print all of the pages of the report by specifying PrintAllPages=TRUE in the Settings section of the FTDPORT2.INI file.

- User-defined page sizes

  You can define two page sizes for viewing reports that contain non-standard page sizes. These two user-defined page sizes are added to the list of other page sizes that can be selected when viewing a report. To define the two page sizes, modify the following two lines in the FTDPORT2.INI file:

  PaperSize1=*width, length*

  PaperSize2=*width, length*

Specify the width and length of each page in the report. All values must be in units of 1440ths of an inch.

- If the page size is in inches, multiple it by 1440.
- If the page size is in millimeters, multiply it by 56.7 and round the result to the nearest whole number.

If no values are set for **PaperSize1** and **PaperSize2**, the default page size for reports that contain non-standard page sizes is 8.5 x 11 inches.

- True Type fonts

  If you want to view reports with True Type fonts, add the following line to the Misc section of the FTDPORT2.INI file: TTONLY=TRUE

Displaying overlays

If a Content Manager OnDemand Windows client displays an overlay when you view an AFP data stream, but the ODWEK AFP Web Viewer does not display the overlay, the AFP Web Viewer might not have found an overlay resource. You can modify the FTDPORT2.INI file to specify the resource directory.

About this task

To configure the AFP Web Viewer to display the overlay, specify the resource directory in the file named FTDPORT2.INI. Using a text editor, like Microsoft Notepad, open the file and find the parameter **ResourceDataPath** under [Preferences], which looks similar to the following example:

```
[Preferences]
DefaultView=DEFAULT
ViewDataPath=C:\Program Files\IBM\OnDemand AFP Web Viewer\Data
ResourceDataPath=C:\Program Files\IBM\OnDemand AFP Web Viewer\Resource
FontDataPath=C:\Program Files\IBM\OnDemand AFP Web Viewer\Font
```

The parameter **ResourceDataPath** that is used for the Content Manager OnDemand client must match the same parameter used for the AFP Web Viewer. Both the Content Manager OnDemand client and the AFP Web Viewer must have an FTDPORT2.INI file.

Important: External overlay resources are not downloaded with the AFP document. If the resource is external (if it is not stored in the same file as the AFP document), then the resource must be downloaded with the AFP document. If the resource is external, it must be stored in the directory specified by the parameter **ResourceDataPath**.

The AFP Web Viewer does not download overlays to the resource directory specified by the parameter **ResourceDataPath**. If the resource cannot be downloaded to the client workstation by some other means, then the AFP data stream must be modified to include the resource so that the AFP document and the AFP resource reside in the same file.

Installing and configuring the HTML5 Line Data Viewer

If you have created a custom servlet that uses the ODWEK Java APIs for your end-users to access data stored in Content Manager OnDemand, you can configure the HTML5 Line Data Viewer to view line data.

About this task

You can launch the HTML5 Line Data Viewer from your servlet by implementing the following code in your servlet environment and then deploying the viewer.

Launching the viewer

The HTML5 Line Data Viewer is launched by redirecting the servlet request to open a document. When the servlet receives a request to open a line data document, it must forward this request to the

`Viewer.html` file, passing parameters along with the request. The parameters that need to be passed are as follows:

action
The callback servlet. It is the servlet that processes the viewerPassthru() call.

docid
The encrypted docid for the line data document you are wanting to view. Make sure the value here is encoded. (See URLEncoder class.)

folder
The name of the folder. Special characters in Content Manager OnDemand folder names require that the folder name be URL encoded. (See URLEncoder class.)

language
The language of the browser for this request. This value can be obtained by reading the Accept-Language header of the request object.

The sample code that follows shows how your redirect might look:

```
response.sendRedirect (request.getContextPath() + “LineDataViewer/Viewer.html” +
“?action=” + request.getRequestURI() +
“&docid=” + URLEncoder.encode(docId, “UTF-8”) +
“&folder=” + URLEncoder.encode (odFolder.getName(), "UTF-8") +
“&language=” + request.getHeader(“Accept-Language”));
```

This will redirect the browser to `Viewer.html` which, once downloaded, will start the process of retrieving and viewing the specified line data document.

Deploying the viewer

The viewer code ships with Content Manager OnDemand and is entirely contained in a zip file named `LineDataHTML5Viewer.zip`. In the previous code sample, this zip file was extracted to a directory called `LineDataViewer` at the root level of the web application. Your web paths may vary.

A few items to note:

- The HTML5 Line Data Viewer is supported on Apple Safari, Google Chrome, Microsoft Edge, and Mozilla Firefox browsers. It is not supported on Microsoft Internet Explorer.
- It is important that your web server be configured to return `json` files as `application/json`. If your web server is not configured to return `json` files as `application/json`, the viewer will not run. Your browser debugger console might show the following error(s):

  ```
  Font: ‘]’ not found
  Object doesn’t support property or method ‘splice’.
  ```

  You can also examine the response headers in the browser to verify the proper mime-type is being returned. See your browser documentation for more information on its browser debugger console and response headers.

Enabling the Copy Pages to File function

You must make the changes to your servlet code that are described in this section only if all of the following are true:

- You have created a custom servlet that uses the ODWEK Java APIs for your end-users to retrieve data stored in Content Manager OnDemand
- You have configured the HTML5 Line Data Viewer so that it is the viewer that launches for line data viewing
- You want to use the **Copy pages to file** functionality available from the HTML5 Line Data Viewer toolbar

If one or more of the conditions listed are not true, then you do not need to make the changes to your servlet code that are described in this section, and you will not be able to use the **Copy pages to file** function.

After the viewerPassthru() call to ODWEK, add a check to see if any HTTP headers are needed by the ODWEK code. If so, then the servlet needs to pass those HTTP headers back to the browser. A sample of how you might update your servlet to use the new functionality is as follows:

```
InputStream is = req.getInputStream();
data = odServer.viewerPassthru(query_string, is);
Hashtable<String,String> headers = odServer.getHeaders();
if (!headers.isEmpty())
{
    Set<String> keys = headers.keySet();
    for (String key: keys) {
        res.setHeader(key, headers.get(key));
    }
}
out.write(data);
```

Installing and configuring the Java Line Data Viewer

If your browser still supports applets, you can use the Java Line Data Viewer to view line data documents.

The Java Line Data Viewer is an applet that is stored on the Web server. After an administrator enables the use of the Java Line Data Viewer, it is automatically loaded into memory on the workstation when the user selects to view a line data document. Verify that the LINEVIEWING parameter in the ODConfig object specifies the viewer that your users will be using.

`ODLineDataViewer2.jar` is the Java Line Data Viewer.

Configuring the ODWEK interface to AFP2PDF

The AFP to PDF transform software can transform AFP documents to PDF documents that a user can view.

Configuring the AFP2PDF.INI file

About this task

The following is an example of an AFP2PDF `.INI` file:

```
[CREDIT-CREDIT]
OptionsFile=
ImageMapFile=creditImageMap.cfg

[default]
OptionsFile=
ImageMapFile=imagemap.cfg
AllObjects=0
```

The structure of the file is similar to a Windows `.INI` file, and contains one stanza for each AFP application and one default stanza. The title line of the stanza identifies the application group and application. For example, the following title line identifies the CREDIT application group and the CREDIT application :

```
[CREDIT-CREDIT]
```

Use the – (dash) character to separate the names in the title line. The names must match the application group and application names defined to the Content Manager OnDemand server. If the application group contains more than one application, then create one stanza for each application.

The parameters that you specify in the `[default]` stanza are used by AFP2PDF to process documents for AFP applications that are not identified in the AFP2PDF `.INI` file. The default parameters are also used if an AFP application stanza does not include one of the parameters specified.

The parameter **OptionsFile** identifies the full path name of the file that contains the transform options used by AFP2PDF. The transform options are used for AFP documents that require special processing. See the AFP2PDF documentation for details about the transform options file.

The parameter **ImageMapFile** identifies the image mapping file. The image mapping file can be used to remove images from the output, improve the look of shaded images, and substitute existing images for images created by the AFP2PDF Transform. Mapping images that are common in most of your AFP documents (such as a company logo) reduces the time required to transform documents. If specified, the image mapping file must exist in the directory that contains AFP2PDF programs. You can specify the directory that contains the programs for AFP2PDF with the ODConfig.AFP2PDF_INSTALL_DIR field in the ODConfig object. See the AFP2PDF documentation for details about the image mapping file.

The parameter **AllObjects** determines how ODWEK processes documents that are stored as large objects in Content Manager OnDemand. The default value is 0 (zero), and means that ODWEK retrieves only the first segment of a document. If you specify 1 (one), then ODWEK retrieves all of the segments and converts them before sending the document to the viewer.

Note: If you enable large object support for very large documents, then your users may experience a significant delay before they can view the document.

Viewing converted documents

About this task

To view converted documents with the Adobe Acrobat viewer, you must obtain the plug-in for the browsers used by your organization.

Chapter 7. Tools for troubleshooting

You can use log files, output, trace facilities, and the Java Console to help you debug problems with ODWEK applications.

You can use the tools listed in the following table to gather information about the system and documents. You can use the information to help solve problems when you configure ODWEK and help other people in your organization who are having problems using the applets and viewers.

Table 3. Problem determination tools

Tool	Purpose	How to start the tool
ODWEK log files	Save information about servers, errors, and users accessing information.	For instructions on creating ODWEK log files, see the following documents: • http://www.ibm.com/support/docview.wss?uid=swg21243419 • "Enabling ODWEK tracing" on page 16 **Note:** Because a significant amount of information can be written to a log file, start logging only when needed, such as when recreating a problem. If you need to log information for extended periods of time, make sure that the log file paths point to storage devices with plenty of free space. Remember to periodically delete old log files from the server.
Java Console	Display messages generated by the applets.	The applets write information to the Java Console. Reviewing this information might be helpful when you are trying to determine the cause of a problem.

AFP Web Viewer Trace Facility	Capture detailed information about AFP documents being viewed with the AFP Web Viewer.	Make sure that the following section exists in the file `FLDPORT2.INI` on the user's workstation:
		`[Misc]`
		`ViewTraceFile=d:\temp\afpplgin.log`
		`Trace=TRUE`
		Verify the path of the log file. Remember to turn off logging after you gathered the information you need.

Content Manager OnDemand System Log	Save system messages (such as log on and log off) and application group messages related to documents (such as query and retrieve) and annotations.	Do the following steps: 1. Enable system and application group logging for the Content Manager OnDemand server. Update the system parameters for the server using the administrative client. 2. Enable the specific application group messages that you want to log. Update the message logging options for the application group using the administrative client.

Java Dump

Java Virtual Machines (JVMs) can create "javadump" or "javacore", which collects diagnostic information related to the JVM and Java applications.

During the run time of a Java process, some Java virtual machines (JVMs) might not respond predictably and might seem to hang up for a long time or until a JVM shutdown occurs. It is not easy to determine the root cause of these problems.

By triggering a javacore when a Java process does not respond, you might be able to collect diagnostic information that is related to the JVM and a Java application captured at a particular point. For example, the information can be about the operating system, the application environment, threads, native stack, locks, and memory. The exact contents depend on the platform on which the application is running.

On some platforms, and in some cases, a javacore is known as a "javadump." The code that creates a javacore is part of the JVM. You can control it by using environment variables and runtime switches. By default, a javacore occurs when the JVM terminates unexpectedly. A javacore can also be triggered by sending specific signals to the JVM. Although a javacore or javadump is present in Oracle JVMs, much of the content of a javacore is added by IBM and, therefore, is present only in IBM JVMs.

The IBM Thread and Monitor Dump Analyzer for Java analyzes javacore and diagnoses monitor locks and thread activities to identify the root cause of hangs, deadlocks, and resource contention or monitor bottlenecks.

IBM Thread and Monitor Dump Analyzer

The IBM Thread and Monitor Dump Analyzer for Java analyzes javacore and diagnoses monitor locks and thread activities to identify the root cause of hangs, deadlocks, and resource contention or monitor bottlenecks.

This technology analyzes each thread and provides diagnostic information, including the following information:

- current thread information
- the signal that caused the javacore
- Java heap information (maximum Java heap size, initial Java heap size, garbage collector counter, allocation failure counter, free Java heap size, and allocated Java heap size)
- number of runnable threads
- total number of threads
- number of monitors locked
- deadlock information

In addition, IBM Thread and Monitor Dump Analyzer for Java Technology provides the recommended size of the Java heap cluster (applicable only to IBM SDK Version 1.5 and Version 1.3.1 SR7 or above) based on the heuristic analysis engine.

IBM Thread and Monitor Dump Analyzer for Java compares each javacore and provides the following information:

- process ID information for threads
- time stamp of the first javacore
- time stamp of the last javacore
- number of garbage collections per minute
- number of allocation failures per minute
- time between the first javacore and the last javacore
- number of hang suspects
- list of hang suspects

This technology also compares all monitor information in javacore and detects deadlock and resource contention or monitor bottlenecks, if there are any.

Related information

https://www.ibm.com/support/knowledgecenter/en/SS3KLZ/com.ibm.java.diagnostics.dbda.isav4.doc/docs/overview.html

Java diagnostic commands

jmap

jmap is a command-line utility that is included in Linux (but not Windows) releases of the Java Development Kit (JDK). This utility prints memory–related statistics for a running JVM or core file.

If jmap is used without any command-line options, then it prints the list of shared objects loaded. For more specific information, you can use the **–heap**, **–histo**, or **–permstat** options.

-heap

Use the **–heap** option to obtain the following information:

- the name of the garbage collector
- algorithm–specific details (such as the number of threads used for parallel garbage collection)
- heap configuration information
- a heap usage summary

-histo

Use the **-histo** option to obtain a class–wise histogram of the heap. For each class, it prints the number of instances in the heap, the total amount of memory used by those objects in bytes, and the fully qualified class name. The histogram is useful when you want to understand how the heap is used.

-permstat

Configuring the size of the permanent generation can be important for applications that dynamically generate and load many classes (for example, Java Server Pages and web containers). If an application loads too many classes, then an `OutOfMemoryError` exception is thrown. Use the **–permstat** option to the jmap command to get statistics for the objects in the permanent generation.

jstat

The jstat utility uses the built–in instrumentation in the HotSpot JVM to provide information about performance and resource consumption of running applications.

You can use the jstat utility to diagnose performance issues, especially issues that are related to heap sizing and garbage collection. Some of its many options can print statistics regarding garbage collection behavior and the capacities and usage of the various generations.

HPROF: Heap Profiler

HPROF is a simple profiler agent that is shipped with JDK Version 5.0. It is a dynamically–linked library that interfaces to the JVM by using the Java Virtual Machine Tools Interface (JVM TI). It writes out profiling information either to a file or to a socket in ASCII or binary format. This information can be further processed by a profiler front–end tool.

HPROF can present processor usage, heap allocation statistics, and monitor contention profiles. In addition, it can output complete heap dumps and report the states of all the monitors and threads in the Java virtual machine. HPROF is useful when you analyze performance, lock contention, memory leaks, and other issues.

HAT: Heap Analysis Tool

The Heap Analysis Tool (HAT) helps debug unintentional object retention. This term is used to describe an object that is no longer needed but is kept alive due to references through some path from a live object.

HAT provides a convenient means to browse the object topology in a heap snapshot that is generated using HPROF. The tool allows a number of queries, including “show me all reference paths from the rootset to this object".

Diagnostic Tool for Java Garbage Collector

The Diagnostic Tool for Java Garbage Collector is a diagnostic tool for optimizing parameters affecting the garbage collector when using the IBM Java virtual machine (JVM).

Applications that run under Java use what is known as a Java heap for garbage collection, which serves as a storage manager in Java development kits provided by IBM and runtime environments.

An analysis of data reflecting the activity of the garbage collector in Java enterprise or stand-alone applications is critical to optimizing tasks running under a JVM. For example, you must consider the following issues when you optimize parameters for Java applications and try to prevent bottlenecks:

- the frequency of the garbage collection cycle
- the time spent in different phases of the garbage collection
- the quantities of heap memory involved in the process
- the characteristics of the allocation failures from which the garbage collection originate
- the unwanted presence of stack overflows

Diagnostic Tool for Java Garbage Collector helps to examine the characteristics of the garbage collection for an application running under a JVM by reading the output of the "verbose" garbage collection and producing textual and graphical visualizations and related statistics. This tool is well suited for looking at the garbage collector activity of a frequently-accessed enterprise application hosted by a WebSphere® Application Server. The tool includes a "multiple file analysis" modality that allows the loading of two or more files simultaneously, thereby enabling you to compare the behavior of two or more application servers in a WebSphere cluster.

Diagnostic Tool for Java Garbage Collector Version 1.3 comes with the following built-in parsers and sample files so you can evaluate features of the tool and review documentation:

- IBM JVM Version 1.5.0
- IBM JVM Version 1.5
- IBM JVM Version 1.2.2

Each parser is likely to be able to parse the data that is produced by some other versions of the IBM JVM; for instance, the JVM Version 1.5 parser also works well with the JVM Version 1.3.*x*. For unsupported versions of the IBM JVM, the tool contains detailed documentation that allows one to code a different parser in Java by implementing the GCParser interface. The documentation describes in detail the assumptions and the rules necessary for coding of the new parser

Related information

https://www.ibm.com/support/knowledgecenter/en/SS3KLZ/com.ibm.java.diagnostics.visualizer.doc/introduction.html

HeapAnalyzer

HeapAnalyzer finds a possible Java heap leak area through its heuristic search engine and analysis of the Java heap dump in Java applications.

Java heap areas define objects, arrays, and classes. When the Garbage Collector allocates areas of storage in the heap, an object continues to be live while a reference to it exists somewhere in the active state of the JVM; therefore, the object is reachable. When an object ceases to be referenced from the active state, it becomes garbage and can be reclaimed for reuse. When this reclamation occurs, the Garbage Collector must process a possible finalizer and also ensure that any internal JVM resources that are associated with the object are returned to the pool of such resources. Java heap dumps are snap shots of Java heaps at specific times.

HeapAnalyzer analyzes Java heap dumps by parsing the Java heap dump, creating directional graphs, transforming them into directional trees, and running the heuristic search engine.

The following are examples of features of HeapAnalyzer:

- List of Java heap leak suspects
- Recommendation of the size of kCluster
- List of gaps among allocated objects/classes/arrays
- Java objects/classes/arrays search engine
- List of objects/classes/arrays by type name
- List of objects/classes/arrays by object name
- List of objects/classes/arrays by address
- List of objects/classes/arrays by size
- List of objects/classes/arrays by size of child
- List of objects/classes/arrays by number of child
- List of objects/classes/arrays by frequency
- List of available heap spaces by size
- Tree view of Java heap dump Loading/saving processed Java heap dumps

HeapRoots

HeapRoots is a tool for debugging memory leaks in Java applications through analysis of "heap dumps."

The Java virtual machine (JVM) maintains a runtime data area (called a heap) for the allocation of all class instances and array objects. The heap storage for objects is automatically reclaimed by a storage management system known as the garbage collector. If an application requires more heap space than can be made available by the garbage collector, the JVM throws an OutOfMemoryError.

HeapRoots analyses "heap dumps," which are files (typically text files) containing information about the objects in the JVM garbage collected heap.

Some IBM VMs (contained in the IBM Developer Kits for Windows, Java Edition) can produce heap dumps on demand; heap dumps can also be triggered by out-of-memory situations.

HeapRoots loads these heap dump files and provides commands for analyzing the data. These commands run algorithms on the data or query for information about the data. HeapRoots provides a command-line interactive interface where one enters commands and gets results. The following list shows examples of analysis:

- Searching or filtering of individual objects
- Summary or tabulation of the various types of objects
- Statistics on heap address space (such as gaps between objects)
- Inward and outward references of an object
- Paths between two objects
- Exploring the heap from source or root objects by following references
- Calculation of objects reachable by an object
- Calculation of objects kept alive by an object.

Notices

This information was developed for products and services that are offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBMproducts. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).
Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM’s Privacy Policy at www.ibm.com/privacy and IBM’s Online Privacy Statement at www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM

Software Products and Software-as-a-Service Privacy Statement" at www.ibm.com/software/info/product-privacy.

Index

Product Number: 5697-N93
5724-J33
5770-RD1

SC19-3353-03