

Develop applications using the IBM Enterprise Content Management Java APIs with IBM Rational Application Developer

A fast start to managing your enterprise content using the Java APIs for IBM Content Manager, IBM FileNet P8 Content Manager, and IBM Information Integrator Content Edition

Skill Level: Intermediate

[Tony Hulme \(tony_hulme@uk.ibm.com\)](mailto:tony_hulme@uk.ibm.com)
Consulting IT Specialist
IBM

17 Apr 2008

Get started with the following IBM® Enterprise Content Management (ECM) Java application programming interfaces (APIs): IBM Content Manager, IBM FileNet® P8 Content Manager, and IBM Information Integrator Content Edition. Set up the IBM Rational® Application Developer environment for each of the APIs covered, and start writing simple code to log on, search, retrieve, and view documents using each API.

Section 1. Before you start

About this tutorial

ECM solutions increasingly need to capture and manage all forms of unstructured content, such as images, forms, fax, office documents, e-mail, video, and audio. To be most effective, this content needs to be integrated across diverse business processes and applications such that it can be delivered on demand to users. This integration typically involves using the API of the ECM solution to allow for in-flight

searching of relevant content. For example, you might want to search for all documents relating to the customer number the user is working with.

It is precisely this level of integration that this tutorial aims to cover. This is an introduction to programming for IBM ECM. This tutorial covers the Java API of the IBM Content Manager product, as well as the IBM FileNet P8 Content Manager product. In addition, the federated Java API of IBM Information Integrator Content Edition is covered — which is capable of accessing either repository (or both).

There are, of course, APIs other than Java, for example Web Services — which are not in the scope of this tutorial.

Prerequisites

This tutorial assumes that you are an experienced IBM Content Manager user or IBM FileNet P8 Content Manager user with Java application development experience using Rational Application Developer.

Required development platform

This tutorial is based on three software environments:

- IBM Content Manager API
 - IBM Content Manager
 - IBM Rational Application Developer
- IBM FileNet P8 Content Manager API
 - IBM FileNet P8 Content Manager
 - IBM Rational Application Developer
- IBM Information Integrator Content Edition API
 - IBM Content Manager
 - IBM Information Integrator Content Edition
 - IBM Rational Application Developer

The software versions used are as follows:

- IBM Content Manager
 - Microsoft® Windows® 2000 Server SP4

- IBM WebSphere® Application Server 5.1.1.2
- IBM Content Manager 8.3
- IBM FileNet P8 Content Manager
 - Microsoft Windows 2003 Enterprise Server SP1
 - IBM WebSphere Application Server 6.2.0.13
 - IBM FileNet P8 4.0 Content Engine
 - IBM FileNet P8 4.0 Application Engine
- IBM Information Integrator Content Edition
 - Microsoft Windows 2000 Server SP4
 - IBM WebSphere Application Server 5.1.1.2
 - IBM Content Manager 8.3
 - IBM Information Integrator Content Edition 8.3
- IBM Rational Application Developer
 - IBM Rational Application Developer 7.0.0

Development scenario

The typical business context for a user who needs access to documents in the ECM repository is to show a list of all documents that relate to the entity they are currently working with, for example: customer, supplier, product, or project. From the list of matching documents, they then select one or more and choose to view them.

The application you will write satisfies these basic requirements by performing the following, using each API in turn:

- Log-on to the repository
- Perform a parametric search
- Retrieve the results set
- View the document metadata
- View the documents

Important: The sample code in this exercise assumes that you will be retrieving “simple” documents, that is, single part document objects with no annotations. The ECM products do allow for a more complex content model, which is beyond the

scope of this tutorial.

Section 2. IBM Content Manager API

Set up the development environment

Set up the IBM Rational Application Developer environment

Java build path

For a newly created project in IBM Rational Application Developer, in addition to the Java Run-time Environment (JRE), the following Java Archive (JAR) files are required (see Figure 1 below):

- [IBMCMROOT] \lib\cmb81.jar
- [IBMCMROOT] \lib\cmb81sdk.jar
- [IBMCMROOT] \lib\cmbview81.jar
- [IBMCMROOT] \lib\log4j-1.2.8.jar
- [DB2ROOT] \java\ db2java.zip

In this test environment, the [IBMCMROOT] directory is: C:\IBM\db2cm8

In this test environment, the [DB2ROOT] directory is: C:\IBM\SQLLIB

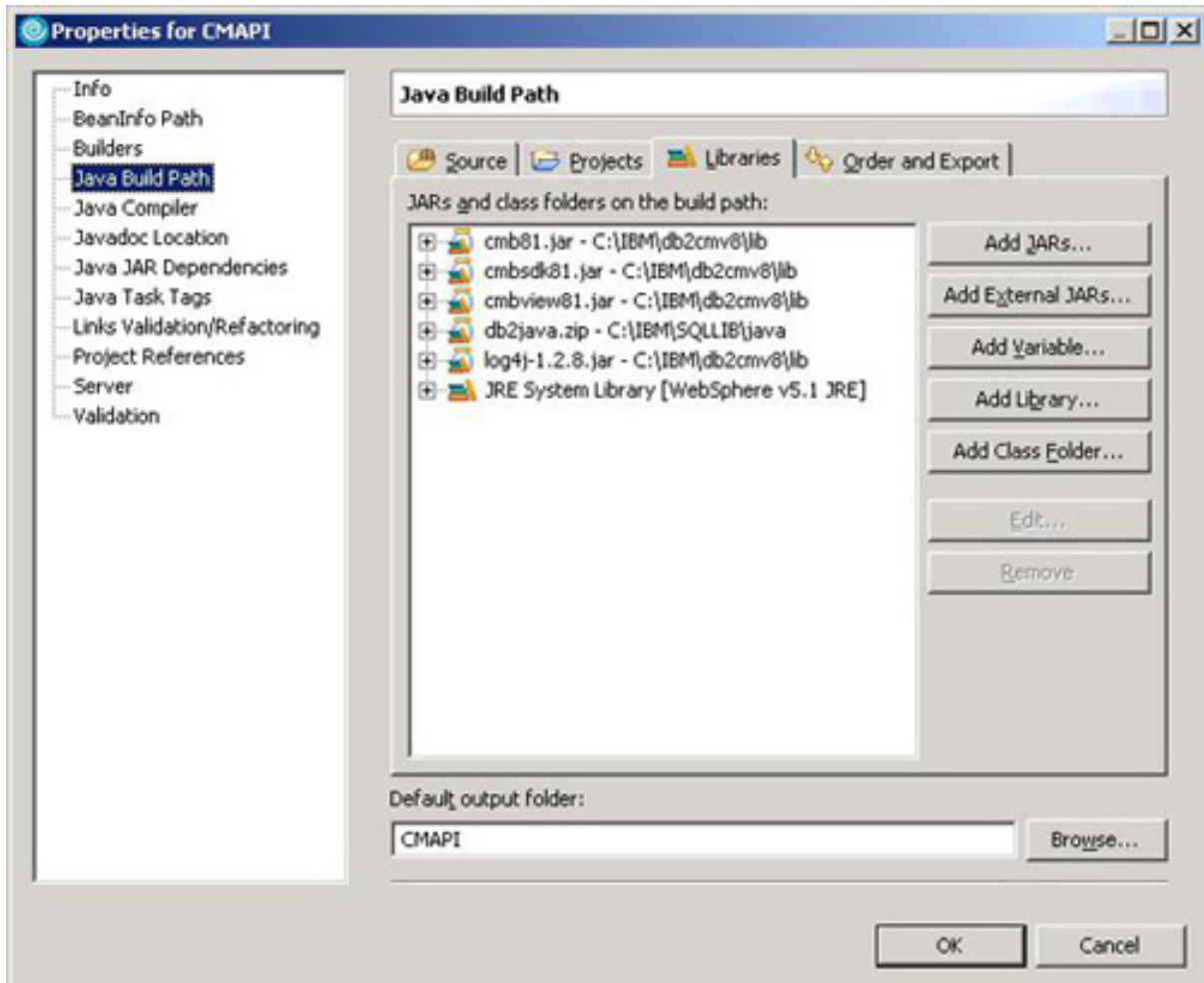
Note: In IBM Content Manager 8.4 (recently available), the required JAR files are slightly different:

There is **no** requirement for db2java.zip.

Instead, the following JAR files must be included:

- [IBMCMROOT] \lib\db2jcc.jar
- [IBMCMROOT] \lib\db2jcc_license_cu.jar
- [IBMCMROOT] \lib\db2jcc_license_cisuj.jar

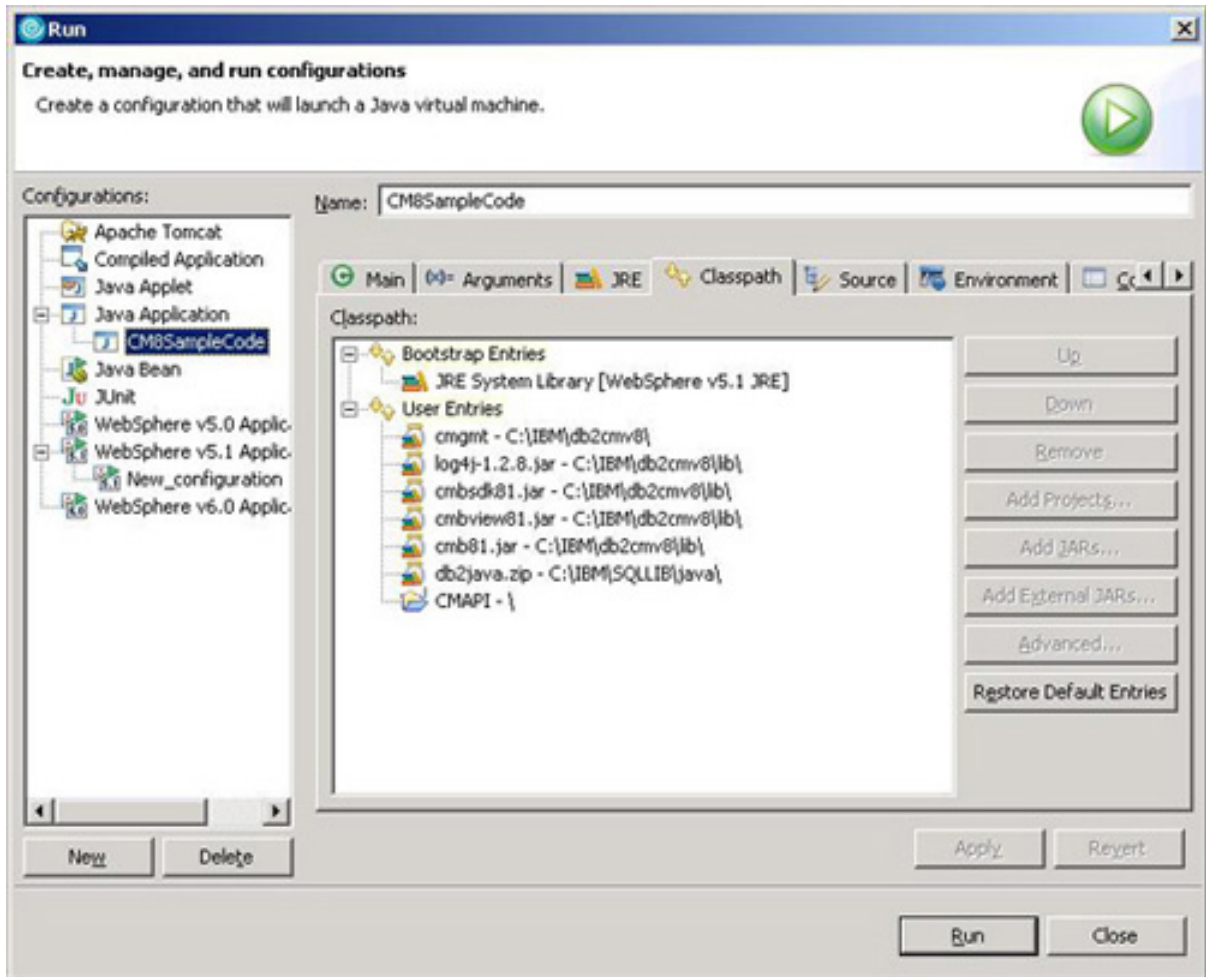
Figure 1. Java build path



These JAR files must be in the class path (see Figure 2 below).

Also, add the directory containing cmbicmenv.properties to the class path — in this test environment the file was installed in: C:\IBM\db2cm8\cmgmt

Figure 2. Class path



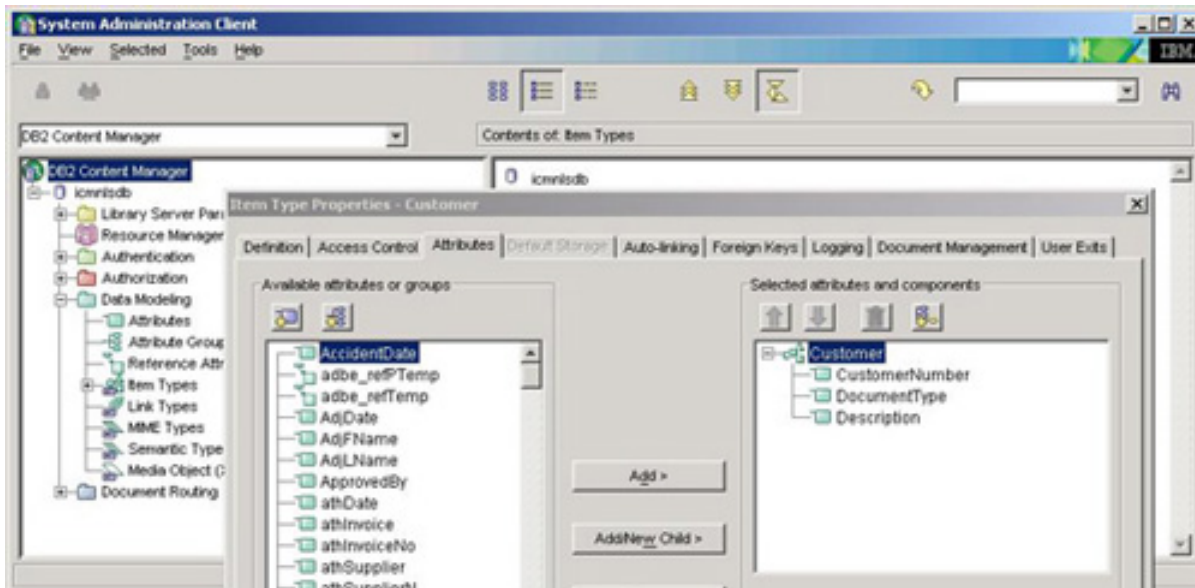
IBM Content Manager application environment

The Library server used in this sample code is "icmnlbdb" — as seen in the IBM Content Manager Administration Client (see Figure 3 below).

The Item Type holding the documents is "Customer" — and the document attribute being used for the search is "Customer Number" — as seen in the IBM Content Manager Administration Client (see Figure 3 below).

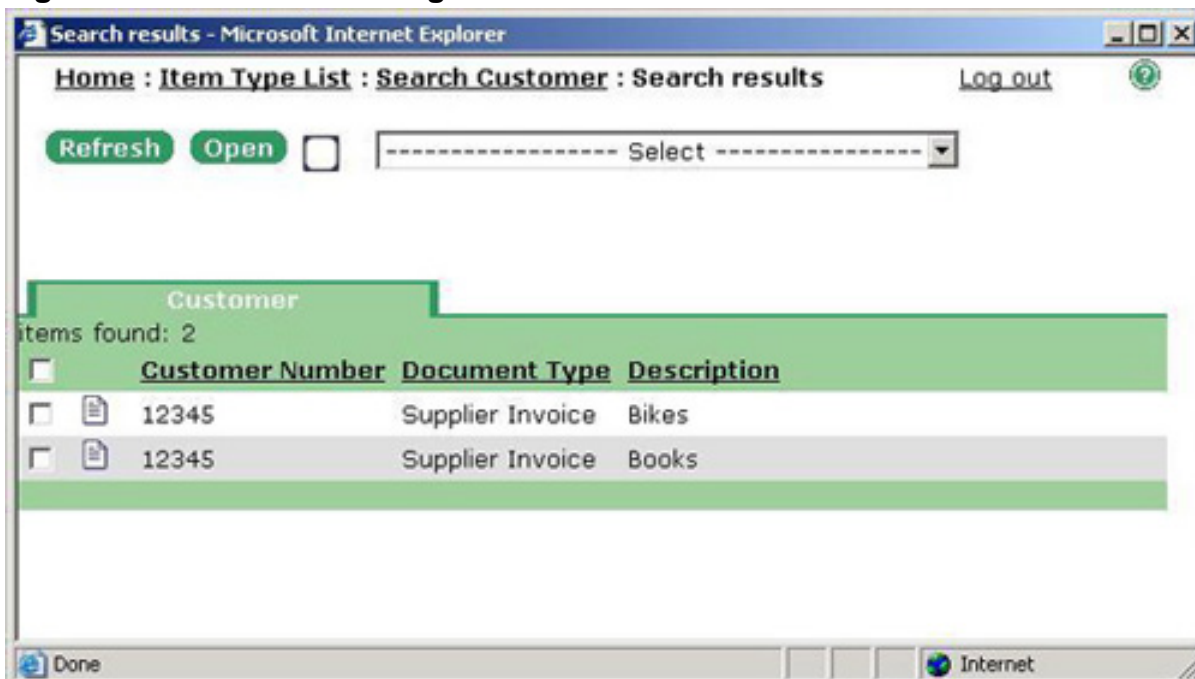
Two documents (a JPEG and a GIF) were added to this Item Type with a customer number of "12345."

Figure 3. IBM Content Manager Administration Client



If the IBM Content Manager application environment is set up correctly, using the eClient client you should be able to sign on as administrator and search for customer number = "12345", as shown in Figure 4.

Figure 4. IBM Content Manager eClient



Listing 1. IBM Content Manager sample code

```
package developerWorks;

import java.io.*;
import java.io.File;
```

```

import com.ibm.mm.beans.*;

public class CMSampleCode {

    public static void main(String[] args)
    {
        try{
            CMSampleCode cm8 = new CMSampleCode();
            // Log on to the CM Library Server
            CMConnection connection = cm8.getCMConnection();
            // Search for documents
            CMSearchResults documents = cm8.searchDocuments(connection);
            // Iterate through results set (documents) to retrieve each document
            for (int i = 0; i < documents.getCount(); i++)
            {
                // Get the document item
                CMItem item = documents.getItem(i);
                // Get document metadata attributes
                cm8.getDocumentMetaData(item);
                // Retrieve the document
                String fileName = cm8.retrieveDocument(connection, item);
                // View the document using Explorer
                cm8.viewDocument(fileName);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public CMConnection getCMConnection() throws Exception
    {
        // Create connection bean
        CMConnection connection = new CMConnection();
        // Set properties on connection bean
        // Set the DataStore type to indicate Content Manager
        connection.setDsType("ICM");
        // Set the Library Server name
        connection.setServerName("icmnlbdb");
        // Set the user id and password for authentication
        connection.setUserid("cmuser");
        connection.setPassword("password");
        // Get the connection
        connection.connect();
        System.out.println("Connected to CM server");
        return connection;
    }

    public CMSearchResults searchDocuments(CMConnection connection)
        throws CMInvalidQueryException, CMConnectFailedException, CMException
    {
        // Search across all CM Item Types
        String entity = "/*";
        // Search where customer number is equal to 12345
        String condition = "[@CustomerNumber = \"12345\"]";
        // Create a query string to hold our search criteria
        String queryString = entity + condition;
        // Get an instance of query service bean
        CMQueryService queryService = connection.getQueryService();
        // Set properties on the query service bean
        short queryType = CMBBaseConstant.CMB_QS_TYPE_XPATH;
        queryService.setQueryString(queryString, queryType);
        queryService.setAsynchSearch(false);
        // Perform search and create results set
        queryService.runQuery();
        // Get an instance of search results bean
        CMSearchResults documents = new CMSearchResults();
        documents.setConnection(connection);
        documents.newResults(queryService.getResults());
        // Return results set object
    }
}

```

```
        return documents;
    }

    public void getDocumentMetaData(CMBItem item)
        throws CMBException, Exception
    {
        // Get document metadata attributes
        String[] names = item.getAttrNames();
        String[] values = item.getAttrValues();
        // For each attribute, show the attribute name and value
        for (int i = 0; i < names.length; i++) {
            System.out.println(names[i] + ": " + values[i] + " ");
        }
    }

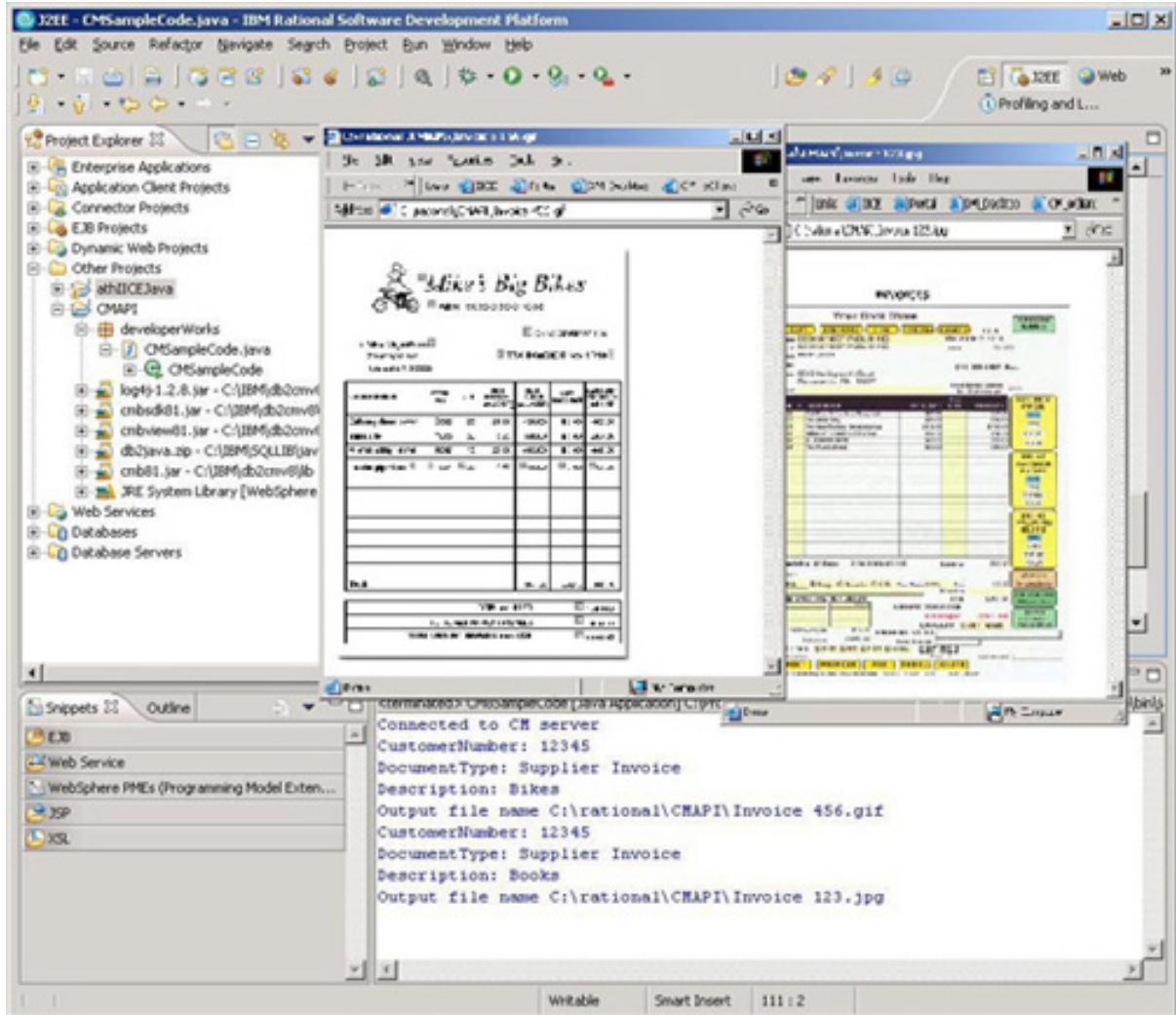
    public String retrieveDocument(CMBConnection connection, CMBItem item)
        throws CMBException, IOException, Exception
    {
        // Get an instance of data management bean
        CMBDataManagement dataManagement = connection.getDataManagement();
        // Set the current data item
        dataManagement.setDataObject(item);
        // Retrieve the original file name
        CMBObject object = dataManagement.getContent(0);
        String inputFileName = object.getOriginalFileName();
        // Parse the file name from the full path
        int pos=inputFileName.lastIndexOf("\\");
        inputFileName = inputFileName.substring(pos+1);
        // Write the document content to a new file
        String fileName = System.getProperty("user.dir")
            + File.separator + inputFileName;
        System.out.println("Output file name " + fileName);
        FileOutputStream fileoutstream = new FileOutputStream(fileName);
        fileoutstream.write(dataManagement.getContent(0).getData());
        fileoutstream.close();
        // Return file name
        return fileName;
    }

    public void viewDocument(String fileName) throws Exception
    {
        // Use Explorer.exe to view the documents
        String VIEWER = "Explorer.exe ";
        Runtime rt = Runtime.getRuntime();
        // Launch the document with explorer
        String cmd = VIEWER + fileName;
        rt.exec(cmd);
    }
} // end main
```

Results from running the sample Code

Running the sample code in the IBM Rational Application Developer environment launches the two documents found in IBM Content Manager, as shown in Figure 5.

Figure 5. Runtime results



Section 3. IBM FileNet P8 Content Manager API

Set up the development environment

Edit the properties file in IBM WebSphere Application Server

The sample code in this tutorial uses Java Remote Method Invocation technology run over Internet Inter-Orb Protocol (RMI-IIOP) to connect to WebSphere Application Server. This requires Common Object Request Broker Architecture (CORBA) security support to be configured.

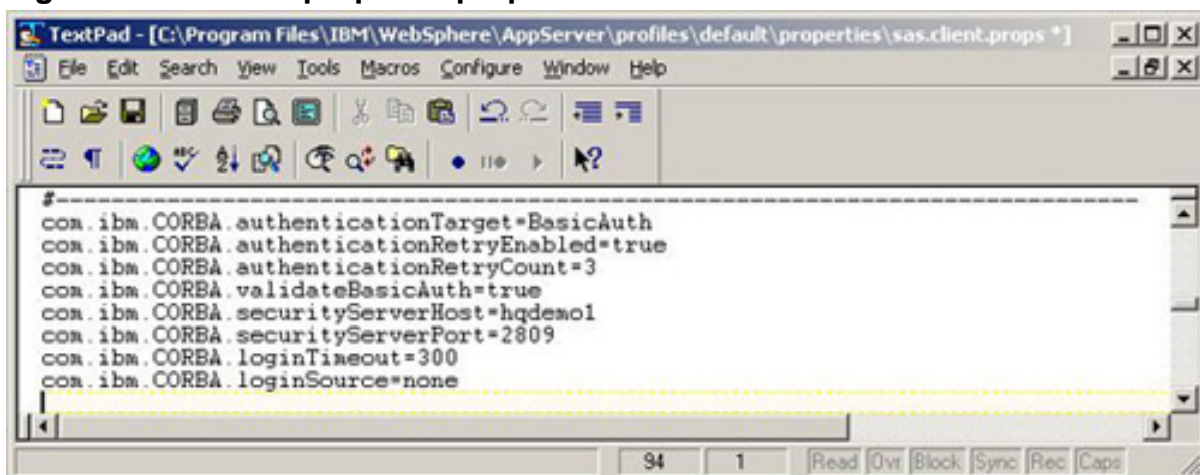
Note: This step would be different if you were using the Web Services Interoperability (WSI) transport, which is also supported.

Edit the `sas.client.props` file — which, by default, can be found in: `C:\Program Files\IBM\WebSphere\AppServer\profiles\default\properties`.

Ensure that the following properties are set (see Figure 6 below):

- `com.ibm.CORBA.securityServerHost=hqdemo1`
- `com.ibm.CORBA.securityServerPort=2809`
- `com.ibm.CORBA.loginSource=none`

Figure 6. sas.client.props file properties



Set up the IBM Rational Application Developer environment

Java Build Path

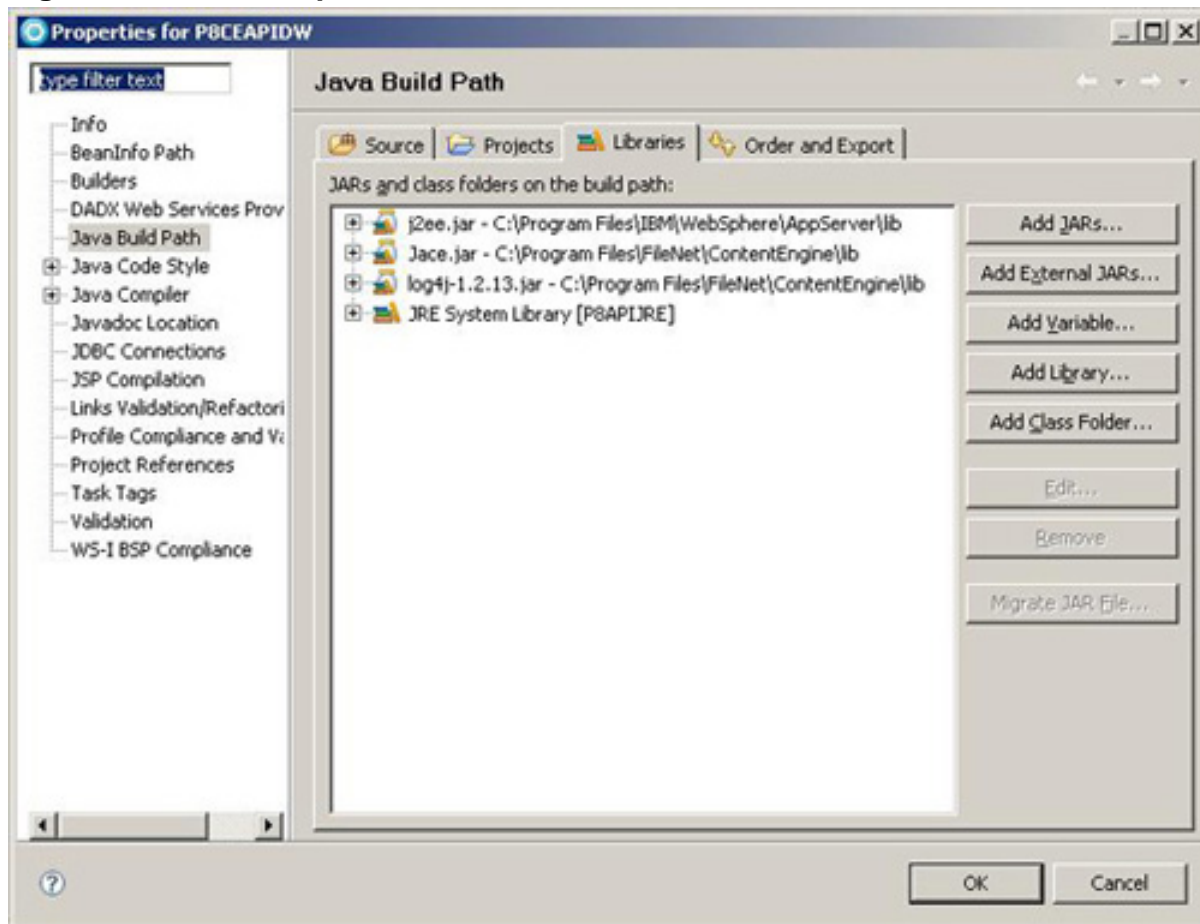
For a newly created project in IBM Rational Application Developer, in addition to the JRE, the following JAR files are required (see Figure 7 below):

- `Jace.jar`
- `log4j-1.2.13.jar`
- `J2EE.jar`

In this test environment, the `Jace.jar` and `log4j-1.2.13.jar` were installed in `C:\Program Files\FileNet\ContentEngine\lib`.

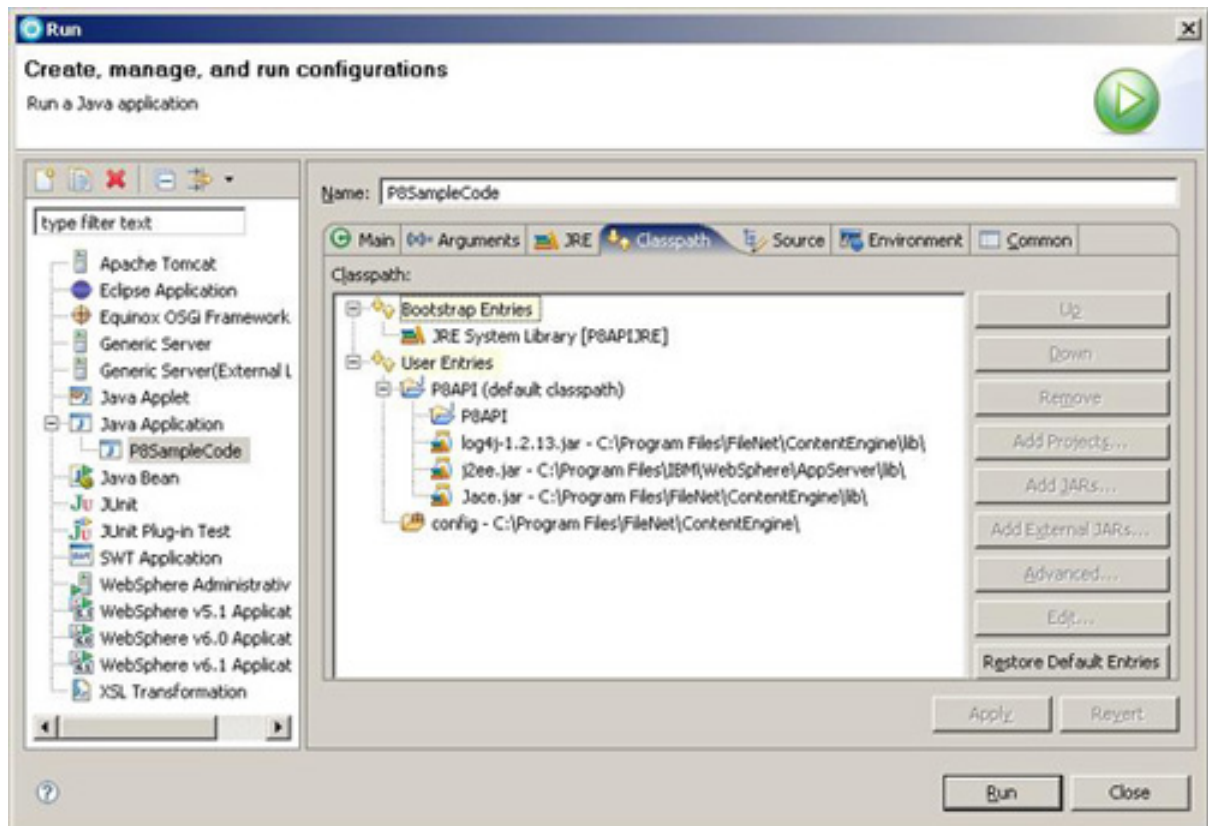
In this test environment, the `J2EE.jar` was installed in `C:\Program Files\IBM\WebSphere\AppServer\lib`.

Figure 7. Java build path



These JAR files must be in the class path — also the log4j.xml file must be in the class path in order for logging to successfully initialize (see Figure 8):

Figure 8. Class path



VM arguments

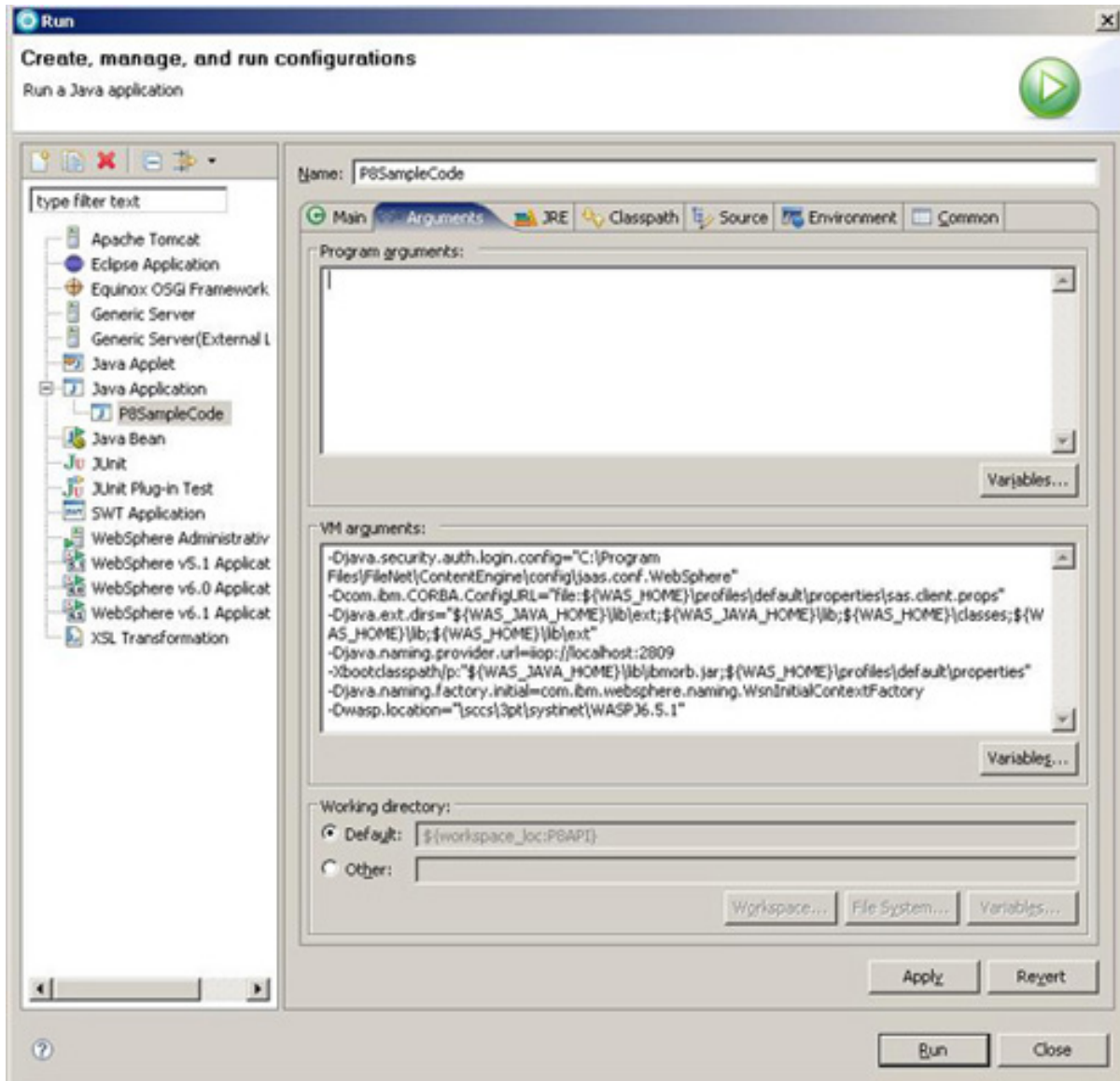
For a newly created class for the project above, set the VM arguments in the IBM Rational Application Developer runtime parameters.

Copy the text from Listing 2 into the VM arguments box shown in Figure 9.

Listing 2. VM Arguments

```
-Djava.security.auth.login.config=
"C:\Program Files\FileNet\ContentEngine\config\jaas.conf.WebSphere"
-Dcom.ibm.CORBA.ConfigURL="file:${WAS_HOME}\profiles\default\properties\sas.client.props"
-Djava.ext.dirs="${WAS_JAVA_HOME}\lib\ext;${WAS_JAVA_HOME}\lib;
${WAS_HOME}\classes;${WAS_HOME}\lib;${WAS_HOME}\lib\ext"
-Djava.naming.provider.url=iio://localhost:2809
-Xbootclasspath/p:"${WAS_JAVA_HOME}\lib\ibmorb.jar;
${WAS_HOME}\profiles\default\properties"
```

Figure 9. VM arguments



Variables

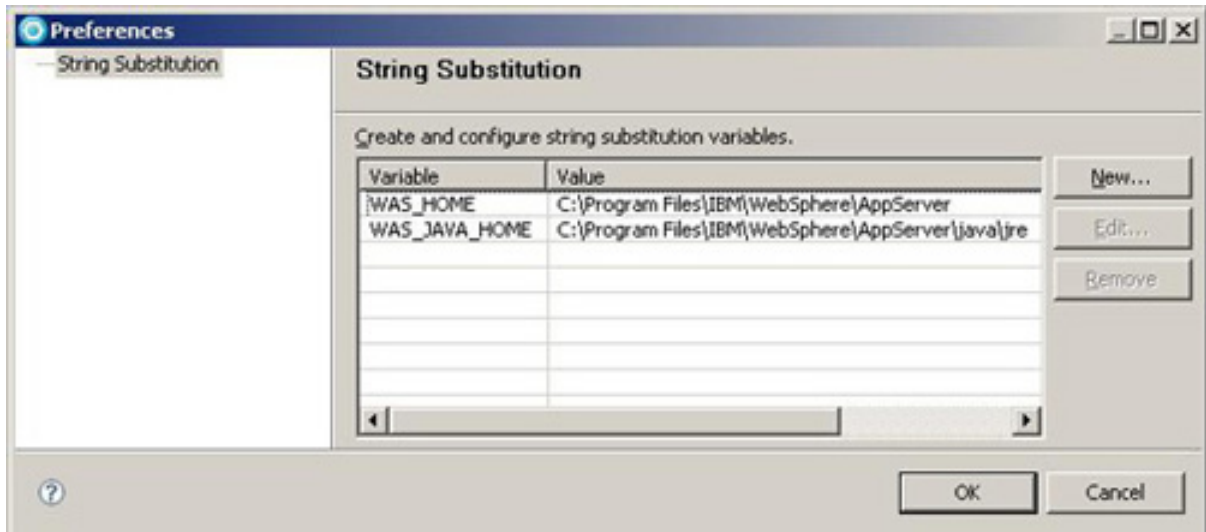
For the same class, add the variables from Listing 3 in the IBM Rational Application Developer runtime parameters shown in Figure 10 below.

Listing 3. VM Variables

```

WAS_HOME           C:\Program Files\IBM\WebSphere\AppServer
WAS_JAVA_HOME      C:\Program Files\IBM\WebSphere\AppServer\java\jre
    
```

Figure 10. Program argument variables



IBM FileNet P8 Content Manager Application Environment

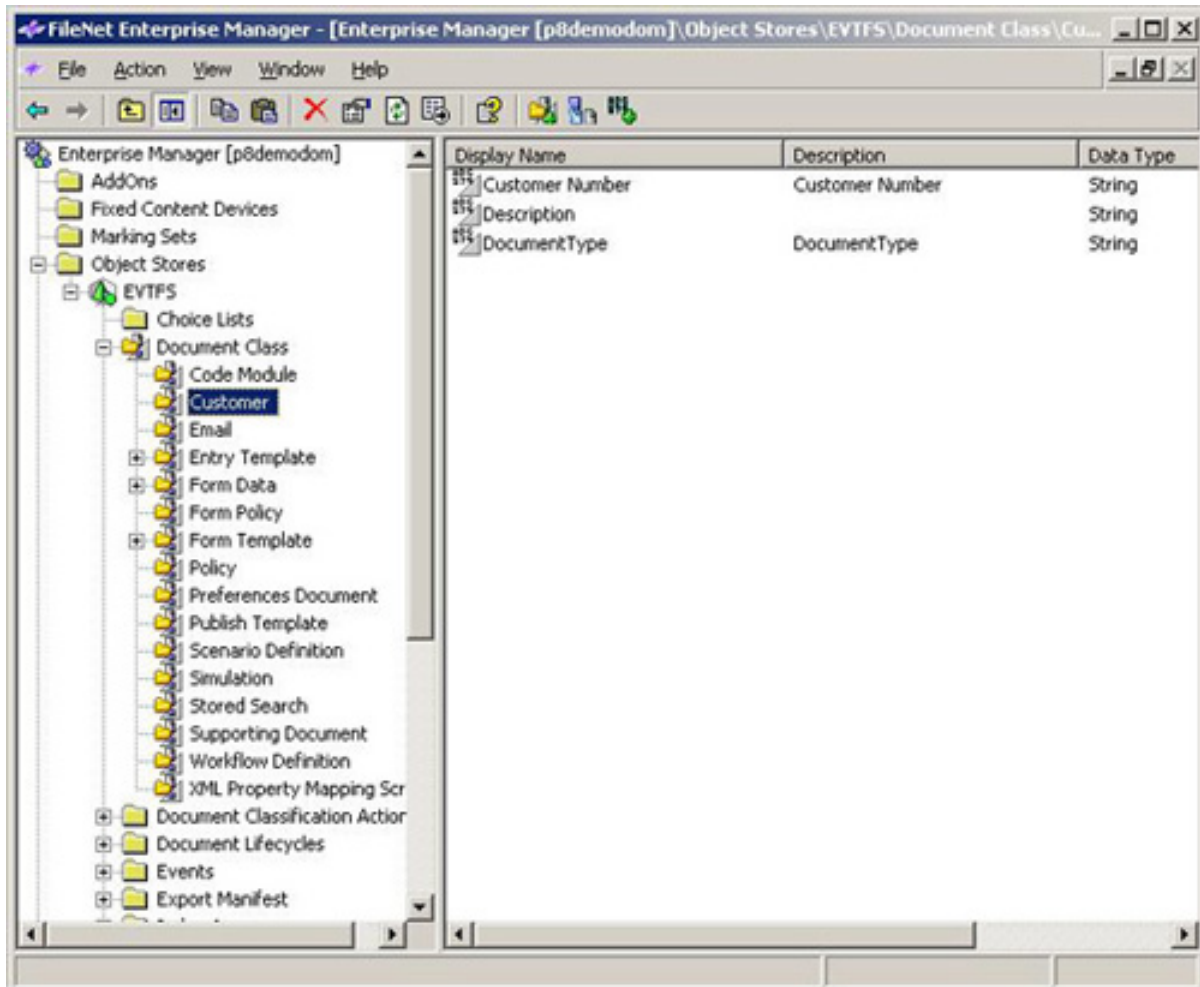
The domain used in this sample code is "p8demodom", as seen in the IBM FileNet P8 Enterprise Manager (Figure 11).

The object store used in this sample code is "EVTFS", as seen in the IBM FileNet P8 Enterprise Manager (Figure 11).

The document class holding the documents is "Customer", and the document property being used for the search is "Customer Number", as seen in the IBM FileNet P8 Enterprise Manager (Figure 11).

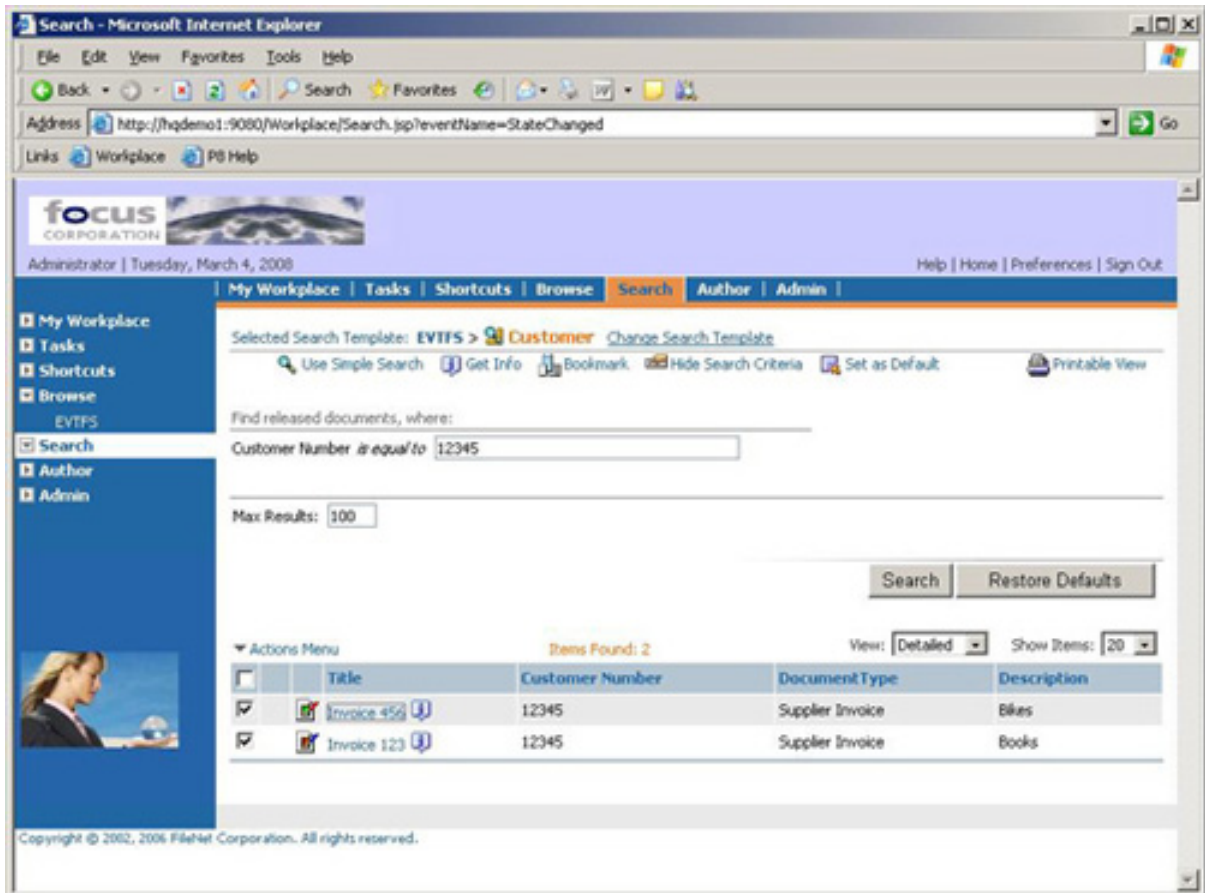
Two documents (a JPEG and a GIF) were added to this document class with a customer number of "12345".

Figure 11. FileNet Enterprise Manager



If the FileNet P8 Content Manager application environment is set up correctly, using the Workplace client you should be able to sign on as administrator and search for customer number = "12345", as shown in Figure 12.

Figure 12. FileNet Workplace Client



Listing 4. IBM FileNet P8 Content Manager sample code

```

package developerWorks;

import java.io.*;
import java.util.Iterator;
import javax.security.auth.Subject;
import com.filenet.api.core.*;
import com.filenet.api.util.UserContext;
import com.filenet.api.collection.*;
import com.filenet.api.query.*;

public class P8SampleCode {

    public static void main(String[] args)
    {
        try{
            P8SampleCode p8 = new P8SampleCode();
            // Log on to the P8 Content Engine
            ObjectStore store = p8.getP8Connection();
            // Search for documents
            DocumentSet documents = p8.searchDocuments(store);
            // Iterate through results set (documents) to retrieve each document
            Iterator it = documents.iterator();
            while (it.hasNext())
            {
                // Get the document item
                Document document = (Document)it.next();
                // Get document metadata attributes
            }
        }
    }
}

```

```

        p8.getDocumentMetaData(document);
        // Retrieve the document
        String fileName = p8.getDocumentContent(document);
        // View the document using Explorer
        p8.viewDocument(fileName);
    }
} catch (Exception e) {
    e.printStackTrace();
}
}

public ObjectStore getP8Connection()
{
    // The connection URI includes the transport protocol (connection type),
    // host name, and port number that are used for server communication
    // Note these are the default P8 configuration parameters
    String uri = "iiop://hqdemol:2809/FileNet/Engine";
    // Set the user id and password for authentication
    String username = "p8user";
    String password = "password";
    // Get the connection
    Connection conn = Factory.Connection.getConnection(uri);
    // The next 3 lines authenticate with the application server using the JAAS API
    Subject subject = UserContext.createSubject( conn, username, password, null);
    UserContext uc = UserContext.get();
    uc.pushSubject(subject);
    // Retrieve the specific Domain Object P8demodom
    Domain domain = Factory.Domain.fetchInstance(conn, "P8demodom", null);
    System.out.println("Domain Name is: " + domain.get_Name());
    // Get the specific object store EVTFS
    ObjectStore store =
        Factory.ObjectStore.fetchInstance(domain, "EVTFS", null);
    System.out.println("Objectstore is: " + store.get_Name());
    // Return the Object Store
    return store;
}

public DocumentSet searchDocuments(ObjectStore os)
{
    // Instantiate a search scope to search our object store
    SearchScope search = new SearchScope(os);
    // Instantiate an SQL object to hold our search criteria
    SearchSQL sql = new SearchSQL();
    // When searching, retrieve certain document
    sql.setSelectList("DocumentType, DocumentTitle, Description,
        ContentElements");
    // Search for all documents
    sql.setFromClauseInitialValue("Document", "d", true);
    // Search where customer number is equal to 12345
    sql.setWhereClause("CustomerNumber='12345'");
    // Perform search and create results set
    DocumentSet documents = (DocumentSet)search.fetchObjects(sql, new
        Integer(50),null, Boolean.valueOf(true));
    // Return results set object
    return documents;
}

public void getDocumentMetaData(Document document)
{
    // Get document metadata attributes
    System.out.println("Document type = " +
        document.getProperties().getStringValue("DocumentType"));
    System.out.println("Document title = " +
        document.getProperties().getStringValue("DocumentTitle"));
    System.out.println("Document description = " +
        document.getProperties().getStringValue("Description"))
}

```

```

public String getDocumentContent(Document document) throws Exception
{
    // Initialize the output file name with the path where to store the document
    String fileName = System.getProperty("user.dir") + File.separator;
    // Get the content elements
    ContentElementList contents = document.get_ContentElements();
    ContentElement content;
    Iterator itContent = contents.iterator();
    // iterate on the elements to retrieve the document
    while (itContent.hasNext())
    {
        content = (ContentElement)itContent.next();
        // Get the document file name
        fileName = fileName+((ContentTransfer)content).get_RetrievalName()
        System.out.println("fileName = " + fileName);
        // Get an input stream for reading document data
        InputStream inputStream =
            ((ContentTransfer)content).accessContentStream();
        // Get an output stream for writing document data
        OutputStream outputStream = new FileOutputStream(fileName);
        // Retrieve document content to the new file
        byte[] nextBytes = new byte[64000];
        int nBytesRead;
        while ((nBytesRead = inputStream.read(nextBytes)) != -1)
        {
            outputStream.write(nextBytes, 0, nBytesRead);
            outputStream.flush();
        }
        // Return the newly created document file
        return fileName;
    }
}

public void viewDocument(String fileName) throws Exception
{
    // Use Explorer.exe to view the documents
    String VIEWER = "Explorer.exe ";
    Runtime rt = Runtime.getRuntime();
    // Launch the document with explorer
    String cmd = VIEWER + fileName;
    rt.exec(cmd);
}

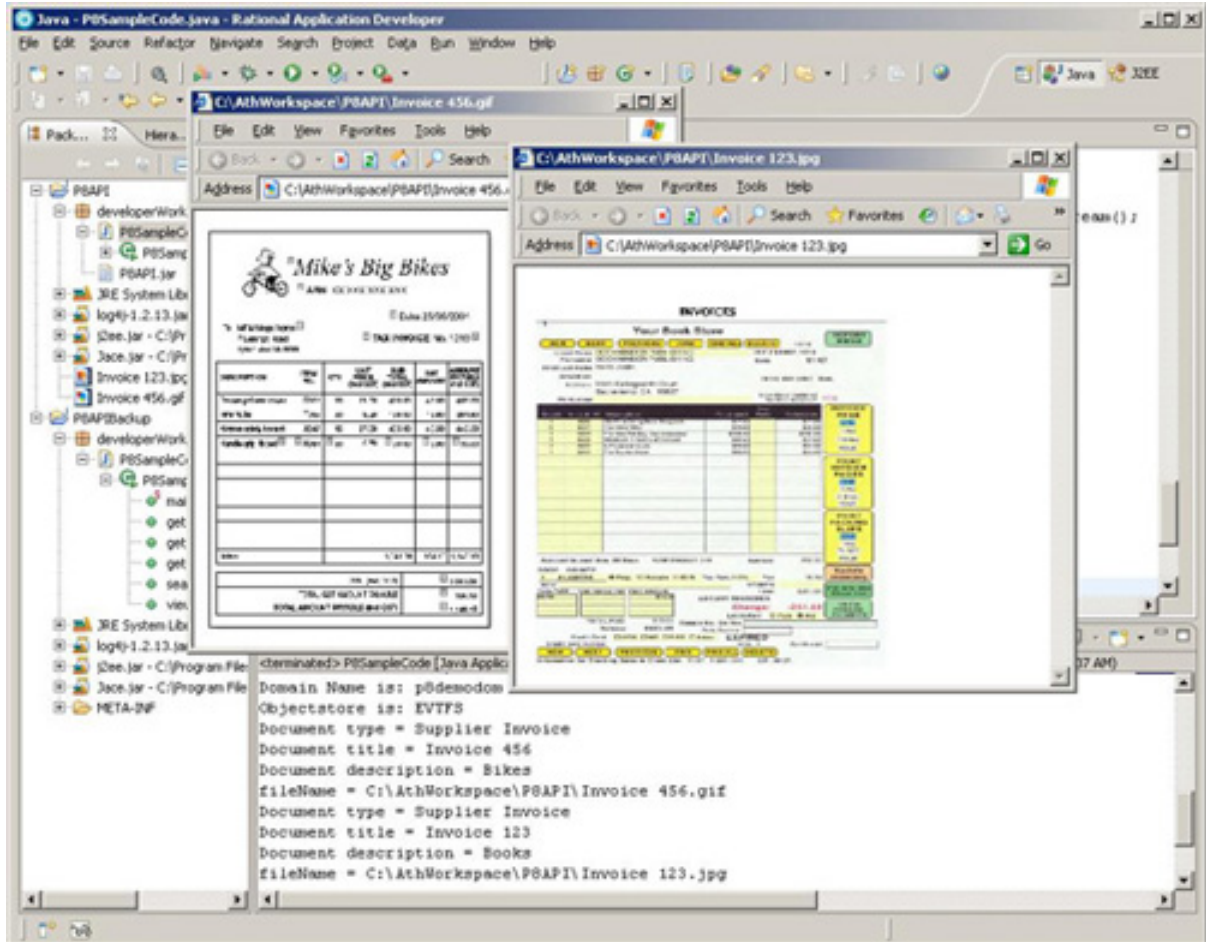
} // end main

```

Results from running the sample code

Running the sample code in the IBM Rational Application Developer environment launches the two documents found in IBM FileNet P8 Content Manager, as shown in Figure 13:

Figure 13. Runtime results



Section 4. IBM Information Integrator Content Edition API

In this section, you will use the IBM Information Integrator Content Edition API to retrieve documents from IBM Content Manager.

Set up the development environment

Set up the IBM Rational Application Developer environment

Java build path

For a newly created project in IBM Rational Application Developer, in addition to the

JRE the following JAR files are required (see Figure 14 below):

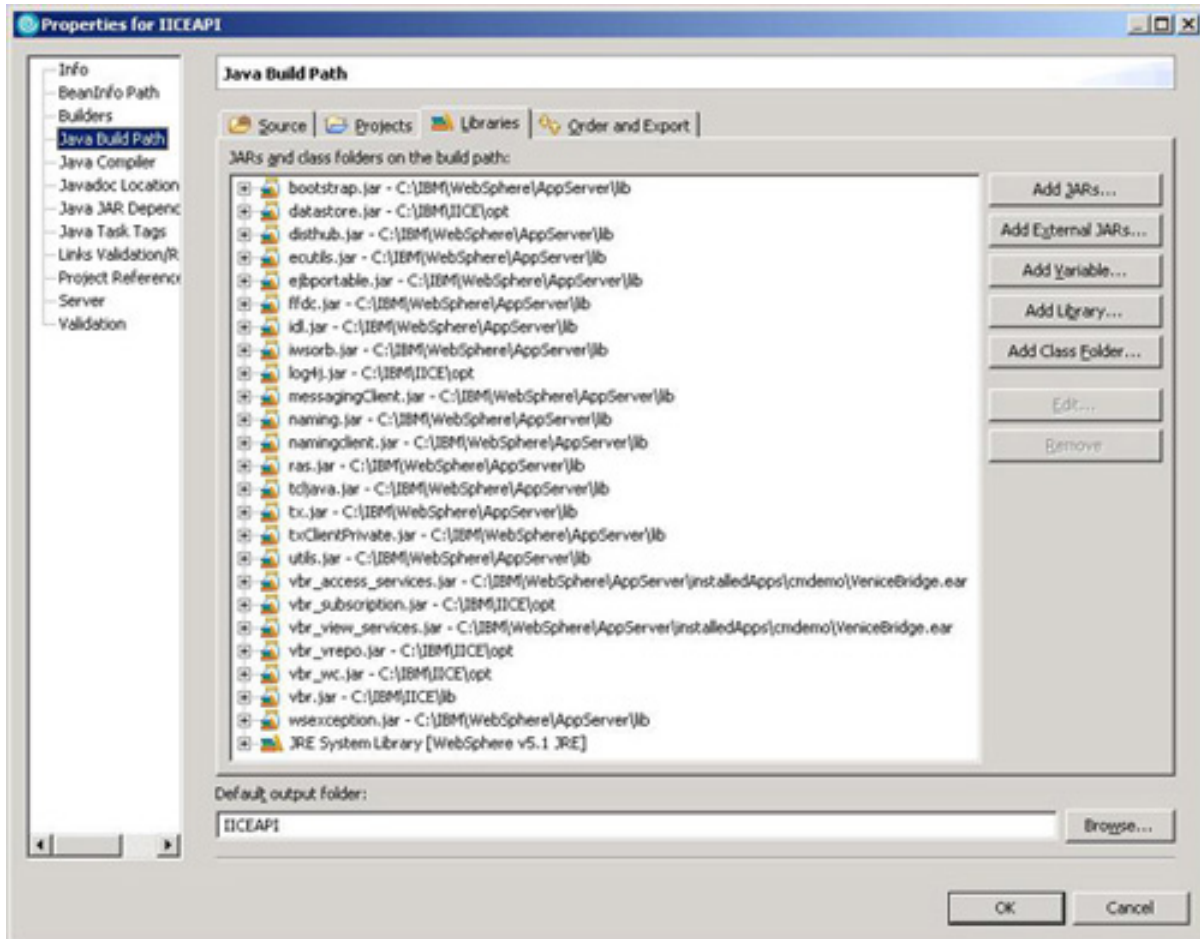
- [IICE_HOME] \lib vbr.jar
- [IICE_HOME] \opt\datastore.jar
- [IICE_HOME] \opt\log4j.jar
- [IICE_HOME] \opt\vbr_subscription.jar
- [IICE_HOME] \opt\vbr_vrepo.jar
- [IICE_HOME] \opt\vbr_wc.jar
- [WAS_HOME] \lib\bootstrap.jar
- [WAS_HOME] \lib\disthub.jar
- [WAS_HOME] \lib\ecutils.jar
- [WAS_HOME] \lib\ejbportable.jar
- [WAS_HOME] \lib\ffdc.jar
- [WAS_HOME] \lib\idl.jar
- [WAS_HOME] \lib\iwsorb.jar
- [WAS_HOME] \lib\messagingClient.jar
- [WAS_HOME] \lib\naming.jar
- [WAS_HOME] \lib\namingclient.jar
- [WAS_HOME] \lib\ras.jar
- [WAS_HOME] \lib\tcljava.jar
- [WAS_HOME] \lib\tx.jar
- [WAS_HOME] \lib\txClientPrivate.jar
- [WAS_HOME] \lib\utils.jar
- [WAS_HOME] \lib\utils.jar
- [WAS_HOME] \lib\wsexception.jar
- [WAS_HOME] \installedApps\< node> \VeniceBridge.ear\vbr_access_services.jar
- [WAS_HOME] \installedApps\< node> \VeniceBridge.ear\vbr_view_services.jar

In this test environment, the [IICE_HOME] directory is C:\IBM\IICE.

In this test environment, the [WAS_HOME] directory is C:\IBM\WebSphere\AppServer.

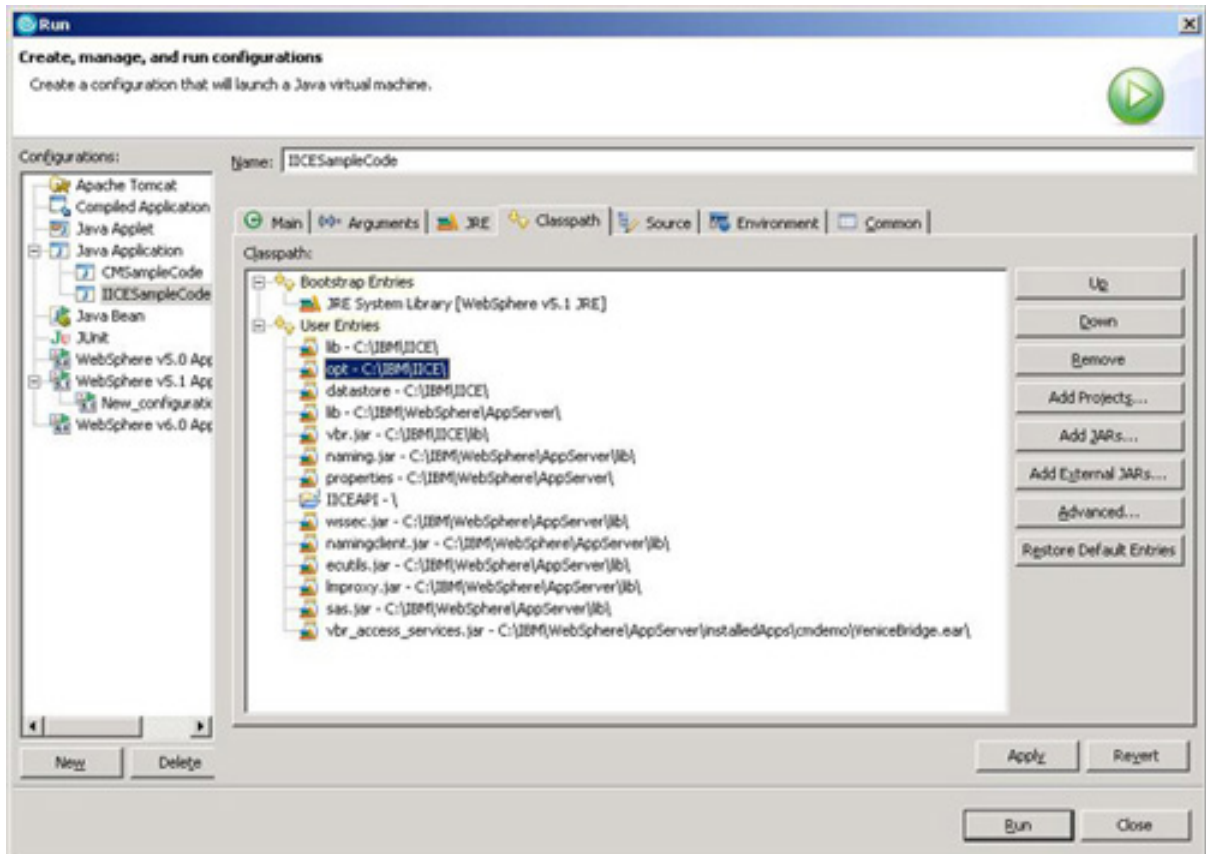
In this test environment, the <node> directory is cmdemo.

Figure 14. Java build path



These JAR files and paths must be in the class path (see Figure 15):

Figure 15. Class path



VM arguments

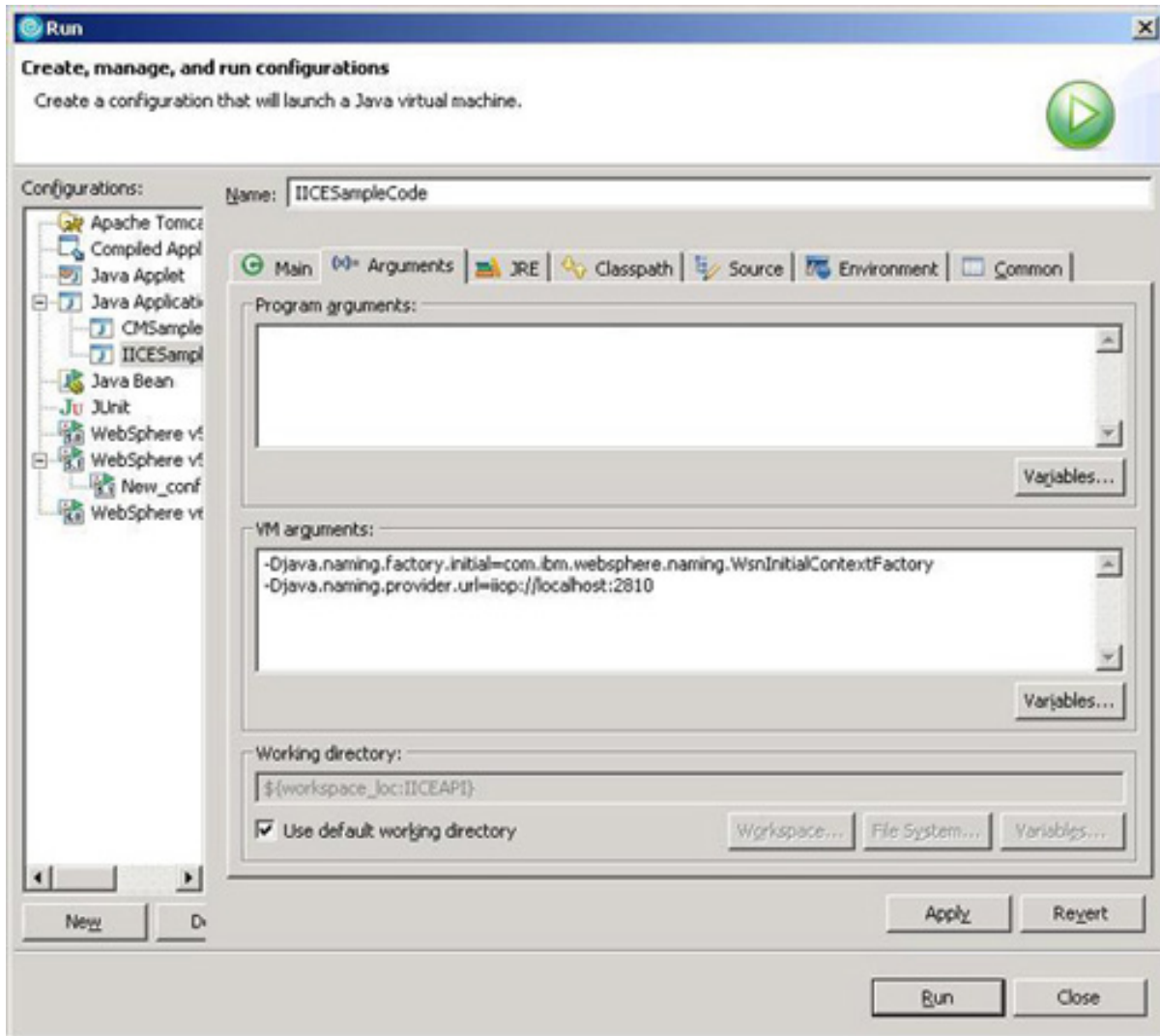
For a newly created class for the project above, set the VM arguments in the IBM Rational Application Developer runtime parameters.

Copy the text from Listing 5 into the VM arguments box, shown in Figure 16.

Listing 5. VM arguments

```
-Djava.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory
-Djava.naming.provider.url=iiop://localhost:2810
```

Figure 16. VM arguments



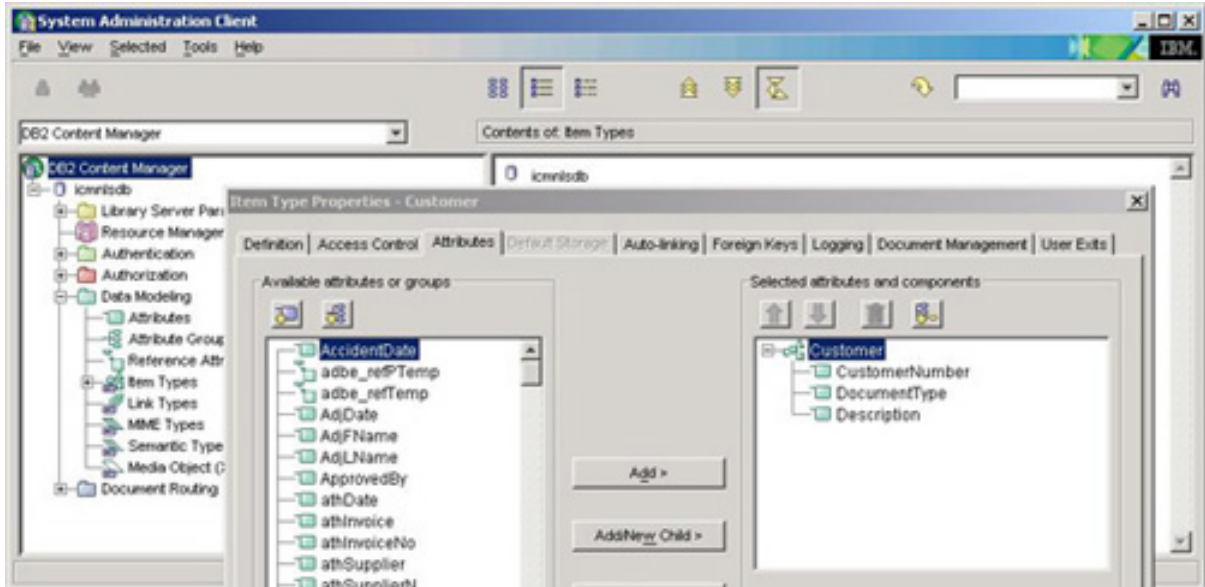
IBM Content Manager Application Environment

The library server used in this sample code is "icmnlbdb", as seen in the IBM Content Manager Administration Client (see Figure 17 below).

The item type holding the documents is "Customer", and the document attribute being used for the search is "Customer Number", as seen in the IBM Content Manager Administration Client (see Figure 17 below).

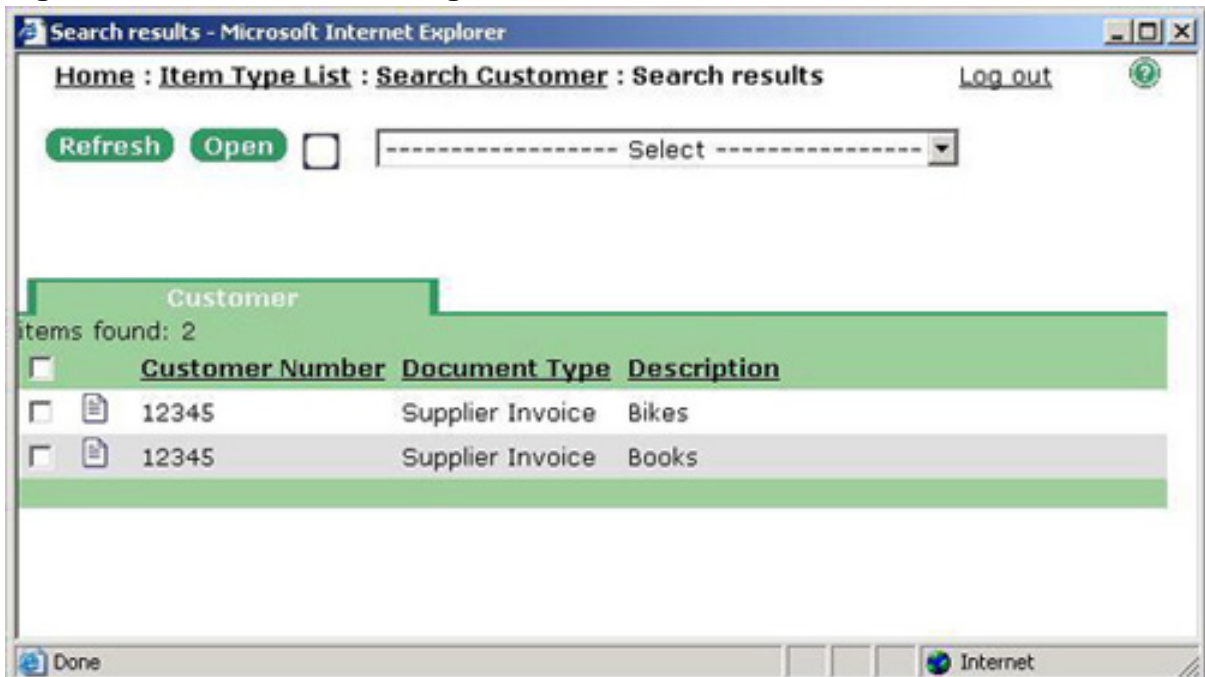
Two documents (a JPEG and a GIF) were added to this item type with a customer number of "12345."

Figure 17. IBM Content Manager Administration Client



If the IBM Content Manager application environment is set up correctly, using the eClient client you should be able to sign on as administrator and search for customer number = "12345", as shown in Figure 18.

Figure 18. IBM Content Manager eClient



IBM Information Integrator Content Edition Application Environment

The sample code in this section assumes that you have IBM Content Manager and IBM Information Integrator Content Edition configured with a working connection (see Figure 19).

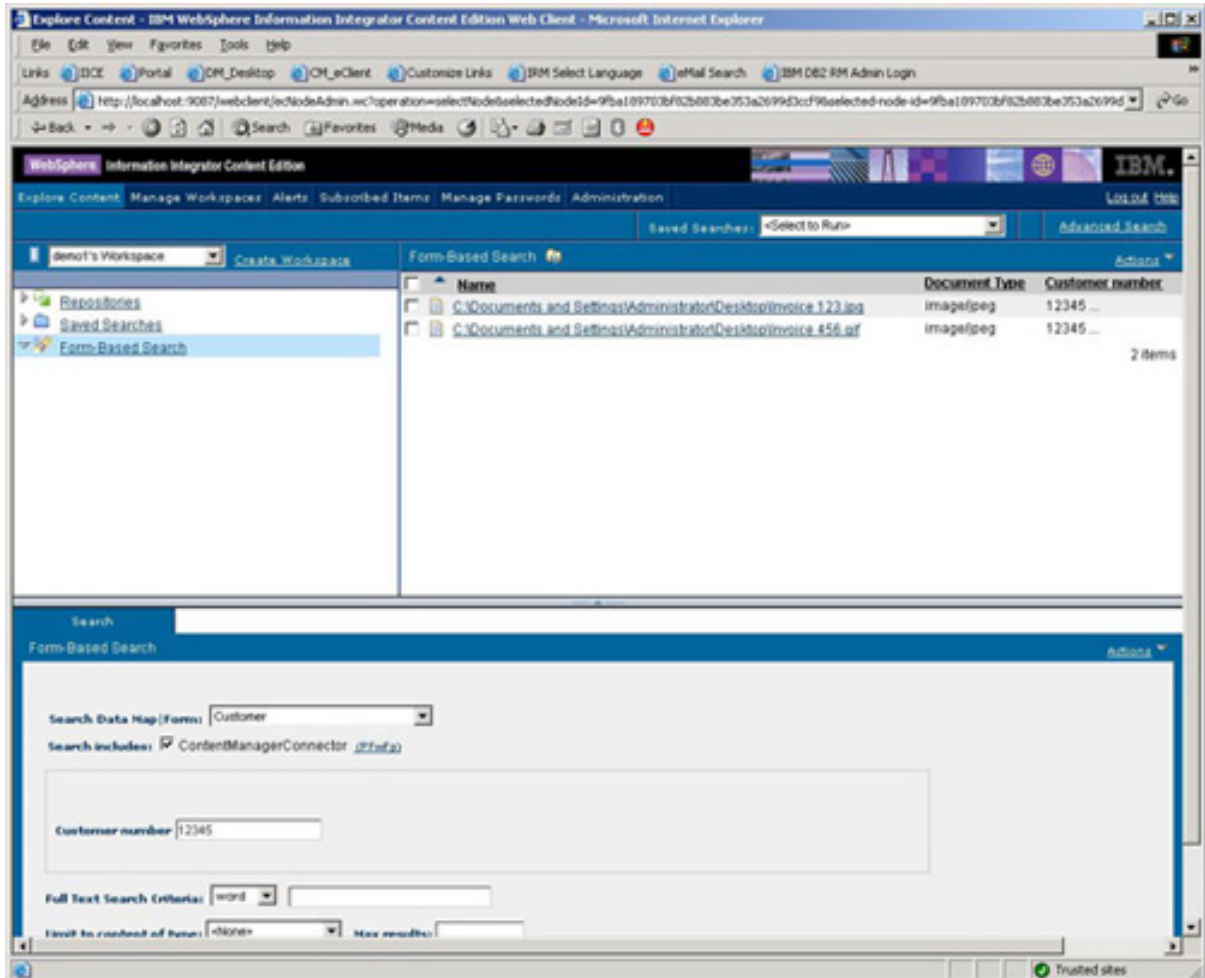
Figure 19. IBM Content Manager Connector

The screenshot shows the Administration Tool interface for the ConfigurationServer. The left pane displays a tree view of configuration categories, with 'ContentManagerConnector' selected under the 'Connectors' folder. The right pane shows a detailed configuration table for this connector.

Name	ContentManagerConnector
Persistent Identifier	a608
Description	
JNDI Information	ContentManagerConnector...
Datastore Name	ICMNLSDB
Database Schema	icmadmin
ICM Servers File	C:\IBM\db2cmv8\cmgmt\cmbicmsrvs.ini
ICM Servers URL	
ICM Environment File	C:\IBM\db2cmv8\cmgmt\cmbicmenv.ini
ICM Environment URL	
Browse Option	Folder hierarchy
Use RMI Proxy Connector	true
RMI Proxy Connector URLs	rmi://localhost:1251/RMIBridgeServer...
Use SOAP Proxy Connector	false
SOAP Proxy Connector URI	vbr_wsb
SOAP Proxy Connector URLs	http://localhost:9081/vbr_axis/services
SOAP Proxy Connector Timeout in Seconds	60
Log off on Deactivate	false
HTTP Access Information	\$VBRHOME/httpcache...
Disable the Connector	false
Logging Information	errors...
Custom Properties	{TitleProperty-XYZ_ClaimFolder=XYZ_Claim

If the IBM Information Integrator Content Edition connection to IBM Content Manager is set up correctly, using the IBM Information Integrator Content Edition Web Client you should be able to sign on as administrator and search for customer number = "12345", as shown in Figure 20.

Figure 20. IBM Information Integrator Content Edition Web Client



Listing 6. IBM Information Integrator Content Edition sample code

```
package developerWorks;

import java.io.*;
import com.venetica.vbr.client.*;

public class IICESampleCode {

    public static void main(String[] args)
    {
        try{
            IICESampleCode iice = new IICESampleCode();
            // Log on to the CM Library Server
            Repository repository = iice.getIICEConnection();
            // Search for documents
            ResultSet documents = iice.searchDocuments(repository);
```

```

        // Iterate through results set (documents) to retrieve each document
        for (int i = 0; i < documents.getRowCount(); i++)
        {
            // Get the document item
            ResultRow item = documents.getRowAt(i);
            Content content = repository.getContent(item.getID(),
                Content.LATEST_VERSION);
            // Get document metadata attributes
            iice.getDocumentMetaData(documents, item);
            // Retrieve the document
            String fileName = iice.retrieveDocument(content);
            // View the document using Explorer
            iice.viewDocument(fileName);
        }
    } catch (Exception ex) {
        ex.printStackTrace(System.err);
    }
}

public Repository getIICEConnection() throws Exception
{
    // Get the repository id for our Content Manager IICE connection
    User user = new User();
    user.initialize();
    Repository repository =
        user.getRepositoryByID("ContentManagerConnector");
    // Set the user id and password for authentication
    String userid = "cmuser";
    String password = "password";
    // Get the connection
    repository.logon(userid, password, "");
    System.out.println("Connected to CM server");
    return repository;
}

public IResultSet searchDocuments(Repository repository) throws Exception
{
    // Create a query string to hold our search criteria
    Query queryString = repository.createQuery();
    // Return only specific document attributes
    queryString.addSelectionProperties("CustomerNumber");
    queryString.addSelectionProperties("DocumentType");
    queryString.addSelectionProperties("Description");
    // Search where customer number is equal to 12345
    queryString.setSelectionCriteria("CustomerNumber = '12345'");
    // Perform search and create results set
    IResultSet documents = queryString.executeContentQuery();
    // Return results set object
    return documents;
}

public void getDocumentMetaData(IResultSet documents, ResultRow item)
    throws Exception
{
    // Get document metadata attributes
    String[] names = documents.getColumnNames();
    // For each attribute retrieved
    for (int i = 0; i < documents.getColumnCount(); i++) {
        // Show the attribute name and value
        System.out.println(names[i] + ": " + item.getColumnValue(i) + " ");
    }
}

public String retrieveDocument(Content content) throws Exception
{
    // Retrieve the original file name
    String inputFileNames = content.getDefaultFileName();
    // Parse the file name from the full path
    int pos=inputFileNames.lastIndexOf("\\");
}

```

```
        inputFileName = inputFileName.substring(pos+1);
        // Write the document content to a new file
        String fileName =
        System.getProperty("user.dir") + File.separator + inputFileName;
        System.out.println("Output file name " + fileName);
        content.getNativeContentAsFile(fileName);
        // Return file name
        return fileName;
    }

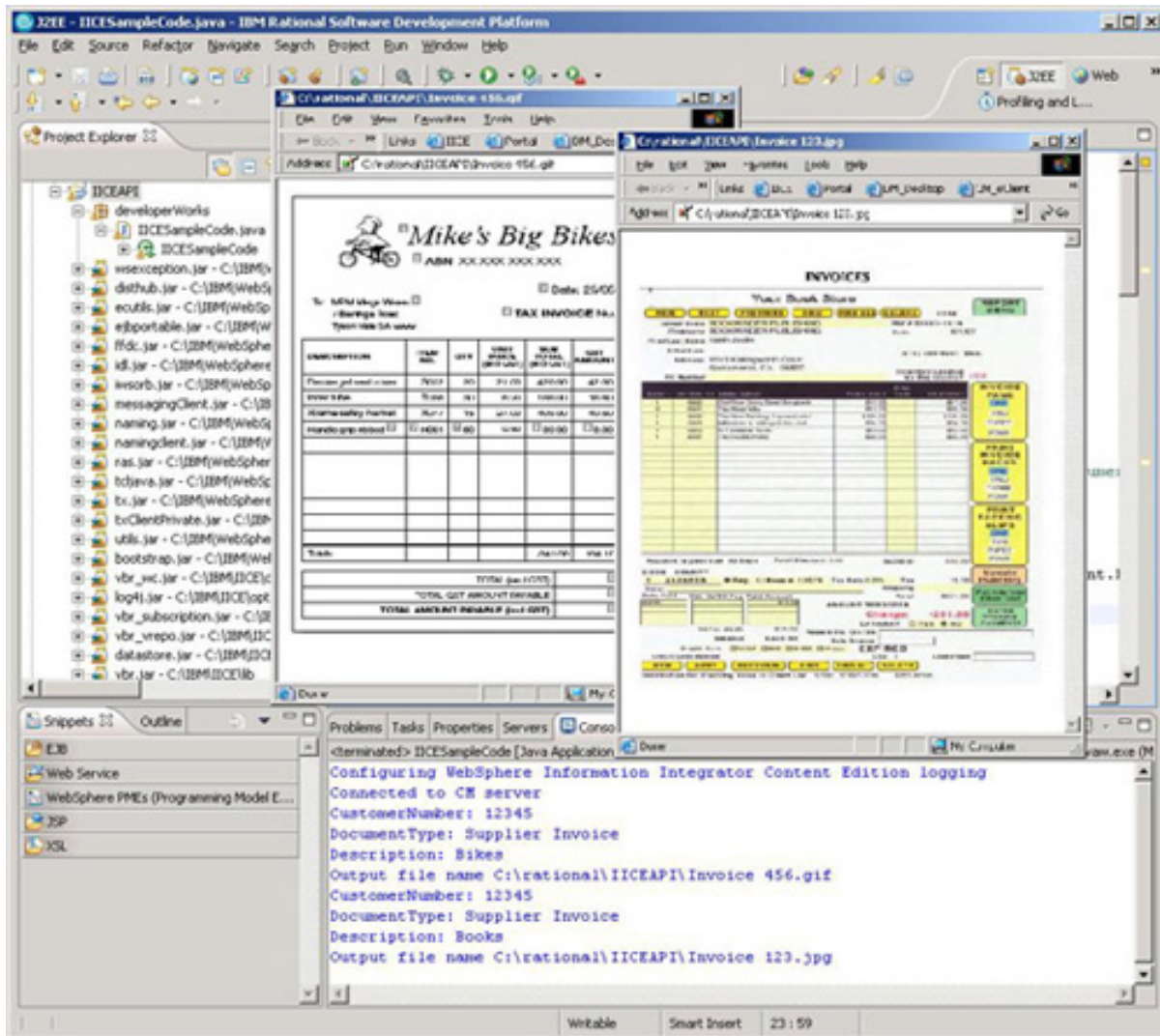
    public void viewDocument(String fileName) throws Exception
    {
        // Use Explorer.exe to view the documents
        String VIEWER = "Explorer.exe ";
        Runtime rt = Runtime.getRuntime();
        // Launch the document with explorer
        String cmd = VIEWER + fileName;
        rt.exec(cmd);
    }
}

} // end main
```

Results from running the sample code

Running the sample code in the IBM Rational Application Developer environment launches the two documents found in IBM Content Manager, as shown in Figure 21.

Figure 21. Runtime results



Section 5. Conclusion

The Java APIs of IBM Content Manager, IBM FileNet P8 Content Manager, and IBM Information Integrator Content Edition provide the key elements to integrate Enterprise Content Management capability within business applications. IBM Rational Application Developer is a great tool for developing simple or complex J2EE applications. This tutorial showed you how to configure the Rational Application Developer environment ready to develop using these Java APIs. It also showed how to satisfy the most common ECM integration requirements of being able to log on, search, retrieve, and view documents matching specific selection criteria.

Downloads

Description	Name	Size	Download method
Sample Code for the IBM Content Manager API	CM8API.zip	5KB	HTTP
Sample Code for the IBM FileNet API	P8API.zip	12KB	HTTP
Sample Code for the IICE API	IICEAPI.zip	5KB	HTTP

[Information about download methods](#)

Resources

Learn

- [IBM Enterprise Content Management](#): Locate information relating to IBM Content Manager and IBM FileNet P8 Content Manager.
- [IBM Rational Software Development](#): Find information relating to IBM Rational Application Developer.
- [Enterprise Content Management area on developerWorks](#): Get the resources you need to advance your skills with IBM Enterprise Content Management products.
- [Java area on developerWorks](#): Discover many resources to help you grow your Java skills.
- [developerWorks Information Management zone](#): Learn more about DB2. Find technical documentation, how-to articles, education, downloads, product information, and more.
- Stay current with [developerWorks technical events and webcasts](#).

Get products and technologies

- Build your next development project with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- Participate in [developerWorks blogs](#) and get involved in the developerWorks community.

About the author

Tony Hulme



With more than 25 years of experience in IT, Tony Hulme is a certified IT Specialist in the Information Management brand of the IBM Software Business. Tony specializes in Enterprise Content Management solutions, where he has more than 15 years of experience.