Using the sample external data service

You can implement an external data service for IBM® Case Manager, Version 5.1 to access data from a different repository or other data source. The sample external data service accesses data from an XML configuration file. The sample is a web application that must be deployed to WebSphere® Application Server. The sample was developed by using WebSphere Application Server, Version 7.0.0.17.

File structure

The top-level files and directories of the sample application are as follows:

`UsingSampleExternalDataService.pdf`
: This document, which describes how to use the sample external data service.

`ConfiguringSampleExternalDataXML.pdf`
: A document that describes how to configure the XML file that determines the case types and properties that the sample service manages.

`build.xml`
: An Apache Ant script file that builds the sample application.

`env.properties`
: A file that defines specific file locations and other values that the build.xml script references. This file must be modified to match your specific build environment.

`src/`
: The Java source files are located under this directory.

`config/`
: Configuration files that are included in the built application.

Modifying the Build Environment Properties

The env.properties file in the top-level directory is referenced by the build.xml script file. Before the sample is built, certain locations and values in this file must be modified to match the specific build environment. See the properties in this file and modify the values accordingly.

Several JAR files are needed to build the sample. The env.properties file defines the following properties that reference JAR file locations:

- 3pt.j2ee.jar. The location where j2ee.jar is located. The JAR file is part of Java EE.
- 3pt.apache.json4j.jar. The location where an Apache JSON4J JAR file is located.
- 3pt.ce.api.jace.jar. The location where the Content Engine API file jace.jar is located.
- 3pt.other.jar. This is the location of another JAR file as described below.

All of the JAR files other than j2ee.jar can be referenced from the location where IBM Case Manager, V5.1 is installed. The env.properties file is preconfigured to reference the typical location where IBM Case Manager is installed on a Windows system.

```
      C:/Program Files/IBM/CaseManagement
```

The j2ee.jar file should be referenced at a location where Java EE is installed. Java EE can be downloaded from the Oracle Java download website. The j2ee.jar file is needed during compilation, but it is not packaged in the WAR file. The WebSphere Application Server environment provides the necessary classes at runtime after the application is deployed.

The 3pt.other.jar property is configured by default to reference the pe3pt.jar file. This JAR file contains the relevant 3rd party classes needed by the sample. Currently the only classes needed in pe3pt.jar are the Apache Commons Logging classes in the package org.apache.commons.logging. As an alternative to using pe3pt.jar, the Apache Commons Logging component can be downloaded directly from http://commons.apache.org.

Modifying the External Data Configuration

The file config/ExternalData.xml is embedded in the WAR file, which is located in the WEB-INF directory. This file can be modified to match the sample external data that you want to configure. See the file ConfiguringSampleExternalDataXML.pdf for information about the structure and options of this XML file.

To update the file after the application is deployed, you can rebuild the application, repackage the WAR, and then update the application from the WebSphere Application Server administrative console.

Alternatively, you can update the file at the following paths on the application server:

```
      <was-root>/profiles/<profile-name>/config/cells\<cell-
         name>/applications/<application-
         name>.ear/deployments/<application-
         name>/sampleexternaldataservice.war/WEB-INF
      <was-root>/profiles/<profile-name>/installedApps/<cell-
         name>/<application-
         name>.ear/sampleexternaldataservice.war/WEB-INF
```

After the files are updated, you must restart the application from the administrative console.

Building the sample

The build.xml file in the top-level directory is a script for use with the Apache Ant application. If necessary, download a version of Ant from http://ant.apache.org. The sample was developed by using Ant version 1.7.

The Java SDK must also be installed on the local file system. If necessary, download a Java SDK. The sample was developed by using Java version 6. The JAVA_HOME environment variable must point to the top-level directory of the Java SDK.

After a version of Ant and the Java SDK are installed on the local file system, set up the environment variables for running Ant. In addition to the JAVA_HOME environment variable, configure the command path to include the Ant binaries directory.

On Windows, configure the JAVA_HOME and Path environment variables by opening the control panel, opening System, and clicking Environment Variables.

Alternatively, you can temporarily configure the variables by opening a new command prompt window and running commands such as:

```
Set JAVA_HOME=c:\JDK-6-Windows\sdk
Set Path=%Path%;c:\Apache-Ant-1.7.0\bin
```

After you configure the environment variables, run the following command If necessary, change the current working directory to the top-level directory of the sample application.

```
cd \SampleExternalDataService
ant
```

Some build messages are output to the console window. An ant.log file is also generated with detailed messages about the build.

If the build is successful, a WAR file is generated in the following location:

```
lib/sampleexternaldataservice.war
```

Deploying the WAR file

After the sample is built, a WAR file will be created under lib/sampleexternaldataservice.war. Deploy this WAR file from the WebSphere Application Server administrative console by navigating to **Applications** > **Application Types** > **WebSphere enterprise applications** > **Install**. Specify the context root of your choice, for example: sampleservice.

After the application is deployed and started, it will be accessible at the following base URL:

```
http://<yourserver>:<portnumber>/<context-root>/ICMEDREST
```

The service responds to two paths after the base URL. You can use the *base-url*/types path to quickly verify that the service is running successfully. It responds with a JSON payload that includes the names of the case types that are configured in the ExternalData.xml file. The request to the path is a GET method and can be called by entering the URL into the address field of a browser, for example:

```
GET http://localhost:9080/sampleservice/ICMEDREST/types
# Response
[{"symbolicName":"SOL_MyCase"}]
```

The Case REST API calls the *base-url*/type path when properties are processed when an external data URI is registered for a solution. The request to the path is a POST method and requires payloads as defined by the External Data Service Protocol. See the product documentation for this protocol if you want to call it directly while testing your external data service. The following example shows a sample request:

```
POST
http://localhost:9080/sampleservice/ICMEDREST/type/SOL_MyCase
# Content
{
    // JSON input payload format as defined by the
    // external data service protocol
}
# Response
{
    // JSON response format as defined by the
    // external data service protocol
}
```

Registering the external data URI

You register the external data URI for a solution by using IBM Case Manager administration client. You must register the URI by using the base URL of the sample service, up to and including ICMEDREST, for example:

```
http://localhost:9080/sampleservice/ICMEDREST
```

Authenticating against an LDAP repository

You can also configure the sample service to be authenticated to the same LDAP repository that the Case REST API application is using. Modify the file config/web.xml file according to comments in that file:

1) Uncomment the section that defines a security constraint and security role.
2) To verify that the authenticated subject can be used inside the Java code, also uncomment the section that defines a context parameter for the Content Engine URI. Modify the URI to the valid URI for your Content Engine domain.
3) Repackage the WAR file with the modified web.xml file and redeploy to WebSphere Application Server.
4) After redeploying the application, you must map the security role from the WebSphere Application Server administration console:
    a. Navigate to the configuration settings for the web application.
    b. Go to **Security role to user/group mapping**.
    c. Map the All Authenticated role to the special subject "All Authenticated in Application's Realm."

Java code in the sample

The Java code of the sample service consists of two packages:

- com.ibm.casemgmt.sampexterndata.rest. These classes implement the RESTful interface to the sample external data service. They are responsible for processing the HTTP requests, for example, converting to and from JSON.
- com.ibm.casemgmt.sampexterndata.api. These classes implement a lower level API layer that is responsible for processing the external data. They include classes to parse the XML and process the configuration data for the various properties.

Some of the main classes in the com.ibm.casemgmt.sampexterndata.rest package are:

- SampleServiceServlet. This is the main Servlet class. It derives from javax.servlet.http.HttpServlet and overrides the various servlet methods: init(), doGet(), doPost(), and so on.
- SampleServiceHandler. The actual processing of a particular request is encapsulated in this class. There is only one handler class in this sample, and it processes both the GET method to the …/types path and the POST method to the …/type path.
- ResourceRequest. This class represents the HTTP request. It further encapsulates the HttpServletRequest object that is passed to the servlet, and it provides additional information to the handler.
- Several classes are used to represent the response from the REST call.
    - IResourceResponse. This defines a generic interface to a response object.
    - BaseResourceResponse. This is a base implementation of the IResourceResponse interface. There is only one subclass of this base class: JSONResourceResponse. However, the intent is that different response types can be represented by different subclasses.
    - JSONResourceResponse. This represents a response with JSON content.
- PropertiesJSONConverter. This is a utility class that has methods for converting the list of properties from the JSON passed as input to the external data service or to the JSON returned as the result.
- Several classes are used for exceptions that are thrown from the REST layer. The classes derive from com.ibm.casemgmt.sampexterndata.api.SEDException.
    - ErrorStatusException. This represents an error when a status code, such as “404 Not Found” or “400 Bad Request,” is returned from the REST call.
    - NotAllowedException. This derives from ErrorStatusException. It represents a “405 Not Allowed” exception and includes the allowed methods that are returned with that status code.

Some of the major classes in the com.ibm.casemgmt.sampexterndata.api package are:

- FileBasedExternalData. This class parses the XML file and generates a hierarchy of objects that corresponds to that XML file.
- The following classes make up the hierarchy of objects after the XML is parsed:
    - ExternalSystemConfiguration. This is the top level of the hierarchy. When a request comes into the external data service, the REST layer code calls this object to process the data and return the relevant list of properties.
    - ExternalCaseTypeConfiguration. This represents the configuration for a particular case type. It references a PropertySet for that case type. It represents a <CaseType> element in the XML.

  - PropertySet. This represents a set of properties that, when processing a request to the external data service, will return the data for the specific properties that are applicable to the request. It represents a <PropertySet> element in the XML.
  - DynamicPropertySet. This object dynamically selects the data for properties based on the property values supplied as input to the external data service. It represents a <DynamicPropertySet> element in the XML.
  - ConditionalPropertySet. This object represents a specific condition when dynamically selecting properties. It represents a <ConditionalPropertySet> element in the XML.
  - ConditionalCriteria. This represents a specific type of condition, for example, Equals and Includes, that is part of a ConditionalPropertySet object and determines whether a specific ConditionalPropertySet matches.
  - ExternalPropertyConfiguration. This represents a specific configuration for a property. It represents a <Property> element in the XML.
- The following classes represent the data for the properties that are passed from or returned to the REST layer code.
  - InputProperty. This generic interface represents a property that is passed as input to the external data service. It provides the basic information in the input payload such as symbolic name and property type and the current property value.
  - ExternalProperty. This represents the external data for a particular property. This is converted to the JSON for a property that is returned in the payload.
  - ExternalChoiceList. If the data defines a choice list for a property, this object represents the choice list information.
  - ExternalChoice. This represents a particular choice in a choice list.
- PropertyValueHolder. This is a utility class that encapsulates the handling of property values of the various types and single and multi cardinality.
- SEDException. This is the type of exception thrown when an error occurs.
- SEDLogger. This is a utility class that is used for logging messages to the application log.

Logging

Using Apache Commons Logging, some logging messages from the sample external data service are written to the WebSphere Application Server application log. The logging can be enabled from the WebSphere Application Server administrative console. Navigate to **Application servers / *server_name* / Process definition / Logging and Tracing / Change log detail levels**. You can configure the logging level for the com.ibm.casemgmt.sampexterndata.* categories.

By default detailed log messages are written to the trace.log log file.