ADF Code Corner

65. Active Data Service Sample –Twitter Client

twitter.com/adfcodecorner

Abstract:

Active Data Service is a push event framework in Oracle ADF Faces that allows developers to implement real time server to client notification when a server side event, like RDBMS change notification or twitter stream notification, occurs. Unlike other technologies in Oracle ADF, Actice data Services (ADS), except for the BAM Data Control, is not a declarative implementation and instead requires developers to write a fair amount of Java. This ADF Code Corner articles provides an Active Data Services sample that listens for Twitter messages of registered friends. Friends are registered within the part of the sample code that starts a Twitter4J instance to start the active service.

This sample does not teach ADS in detail but comments on the code it uses to implement the use case. You find references to additional documentation at the end of this article.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
01-DEC-2010

Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

Introduction

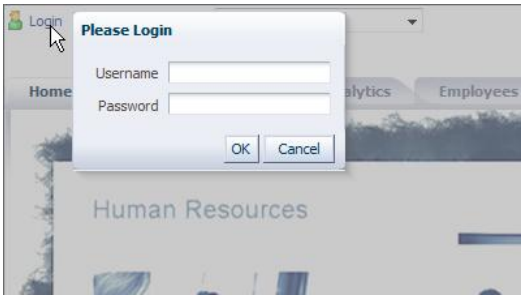
Active Data Services (ADS) in Oracle ADF Faces solves the problem of the disconnected Web that disallows servers to actively push messages to the browser client. ADS is a framework that lets developers decorate a component model, like table, to then configure the `adf-config.xml` file for whether to use streaming, poll or long-poll for checking server side events and payloads.

Twitter – http://twitter.com – provides development API for application developers to program against the site's service. Twitter4j is an open source library that simplifies the use of Twitter streaming APIs. A version of the Twitter4j runrime JAR files is shiupped with the sample.

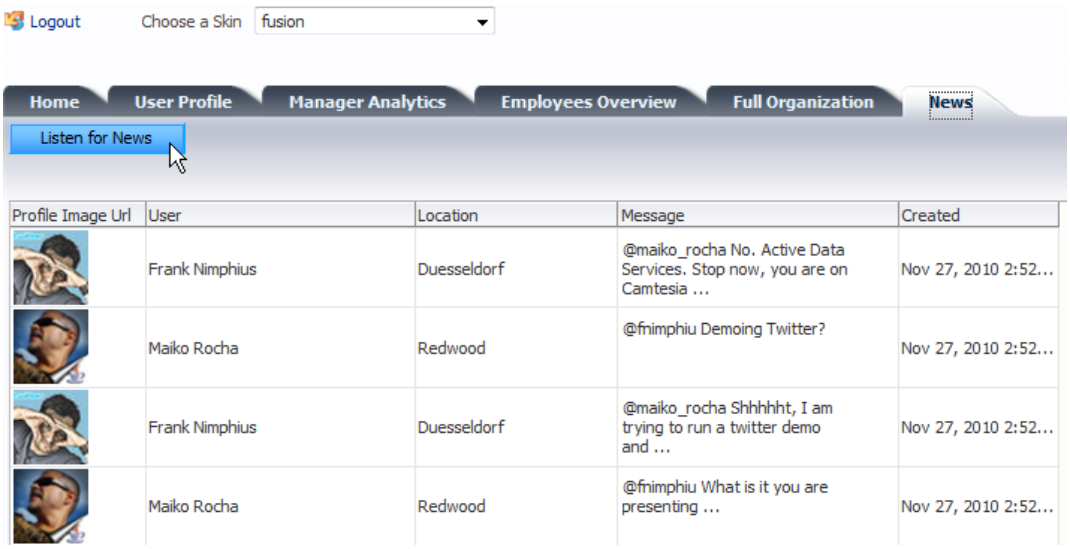
In the sample, a managed bean is used to connect to Twitter, requiring username and password of a valid Twitter account. The sample then starts listening for registered friends – which you learn how to setup – to display incoming messages in an ADF Faces table.

Note This sample neither uses the full functionality available in Active Data Services, nor does it make use of all that is possible with Twitter4J.

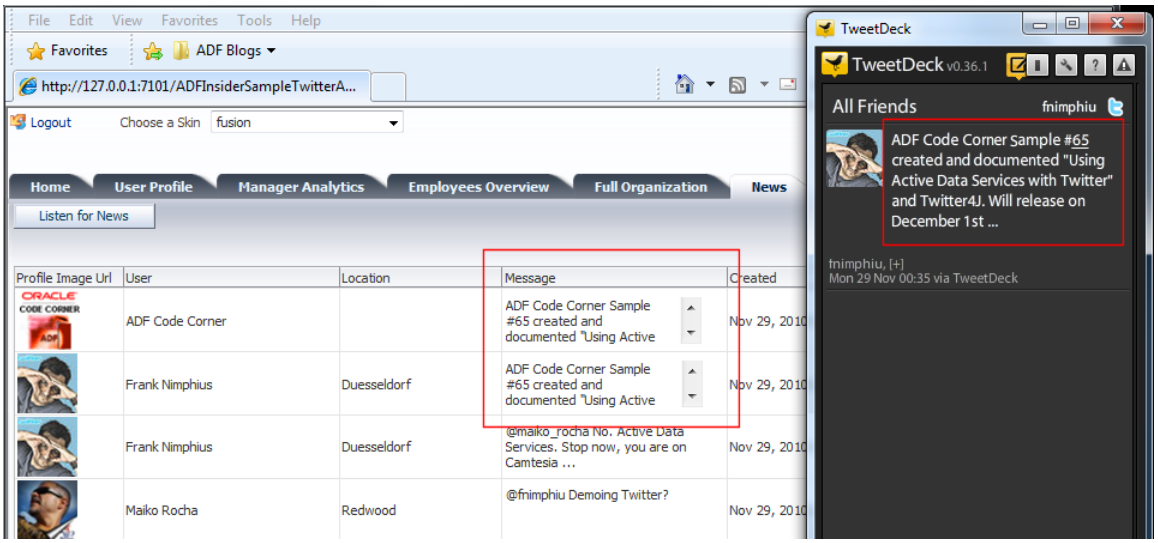
At runtime – and only after you completed reading this article - you authenticate to the sample application with your Twitter account credentials.



This does not yet create a connection to Twitter but performs web authentication and keeps the account information for later when you decide to start listening.



A tweet sent from one of the registered friends immediately shows in the ADF Faces table

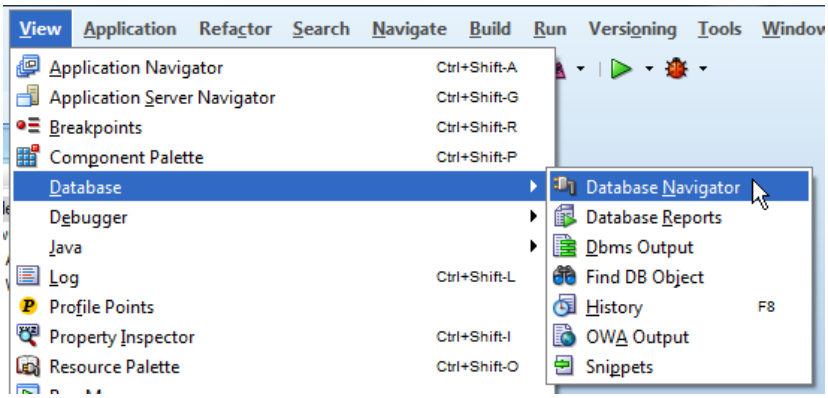


Note: The above conversation between myself and Maiko Rocha in the team is a fake and coded in the sample to not start with an empty table, which for a demo you show on stage would be just too boring. However, you can change this "initial" conversation to have you and one of your peers chatting.

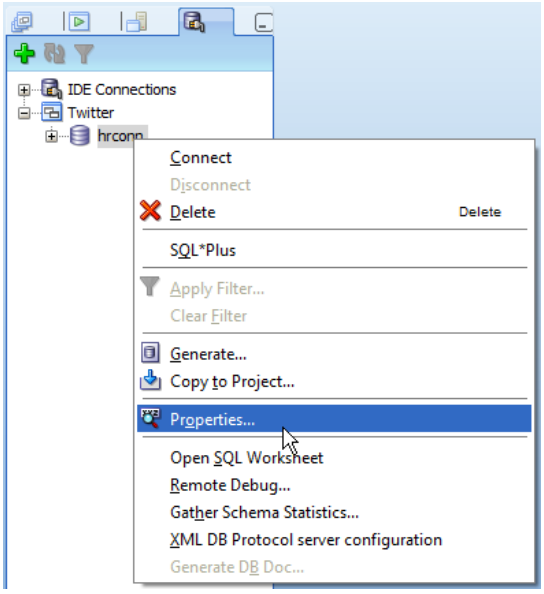
To start the listen process, click the "Listen for News" button, which then connects to Twitter – assuming Internet connection and proxy settings allow it,

Database Connection

The sample uses data of the HR schema for the other parts of it. For this you need to change the database connection to point to a database on your laptop or near you. For his, select **View | Database | Database** Navigator from the Oracle JDeveloper menu

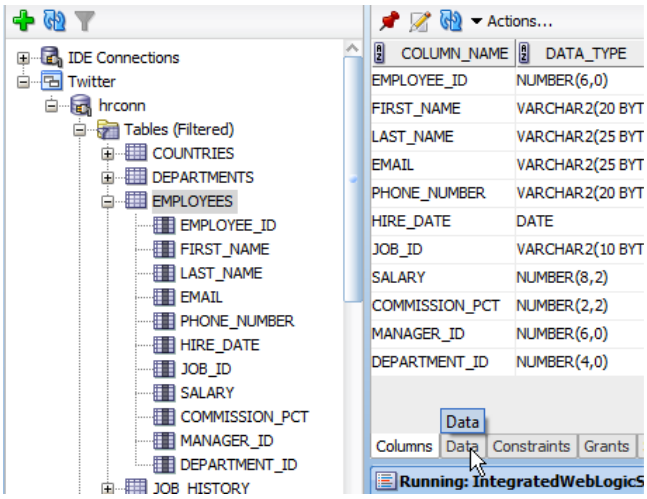


Expand the **Twitter** node and select the **hrconn** entry with the right mouse button. Choose the "Properties" option in the opened context menu to bring up the database connection dialog.

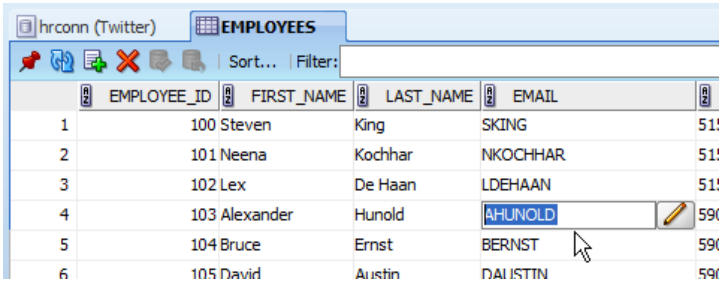


Optional: The **User Profile** tab in the application shows detail data about the authenticated user. For the perfect illusion, you can change the **EMAIL** column value of an employee, like "Alexander Hunold", to the name of your Twitter account name.

For this, expand the **hrconn** connection in the Database Navigator view and double click onto the Employees table. In the opened dialog, select the **Data** tab at the bottom



Navigate to the account of Alexander Hunold and update the EMAIL value with your Twitter account. This now makes you employee and manager at the same time, information shown in the user profile tab



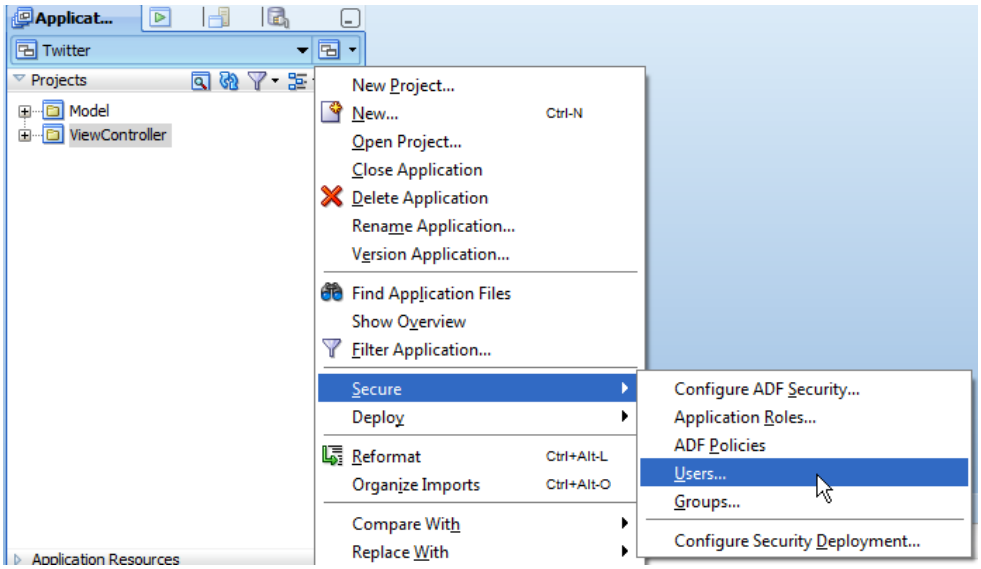
Configuring and running the Sample

Before you can run the sample, you need to

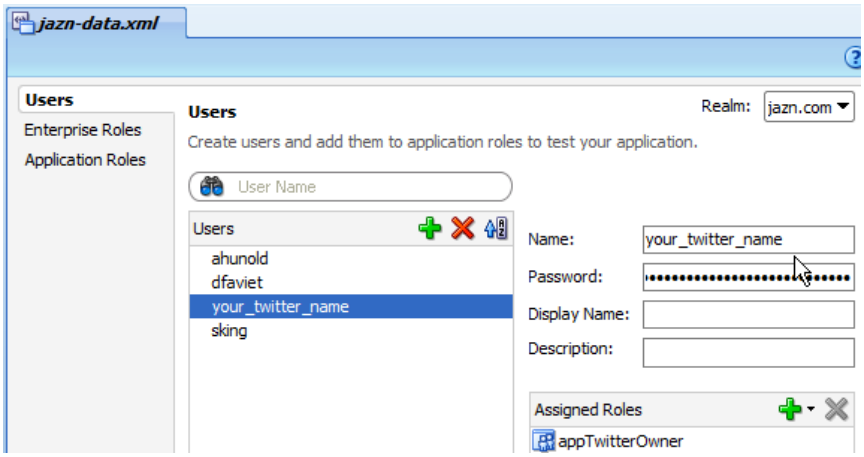
- Configure the jazn-data.xml configuration file with your Twitter username and password
- Add Twitter account Ids of users you want to listen for (this should be your account too, in case you plan to demo the sample on stage
- Deploy the Twitter4J JAR as shared library to WLS

Configuring jazn-data.xml

In the Oracle JDeveloper Application Navigator, click the folder icon next to the "Twitter" workspace name and choose Secure | Users from the popup menu.



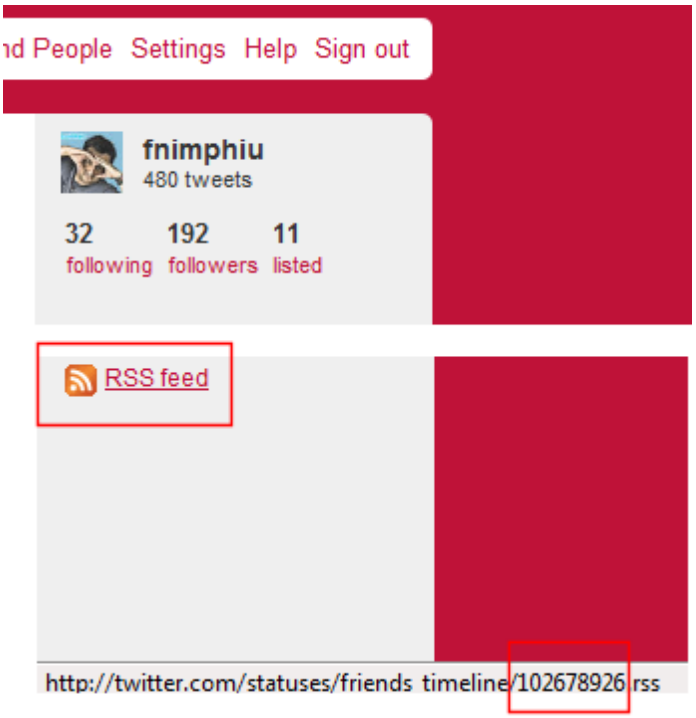
In the opened dialog, configure the "your_twitter_name" account with the username and password of your Twitter account. The password is encrypted and kept with the application afterwards Click the Save icon to ensure your changes are persisted.



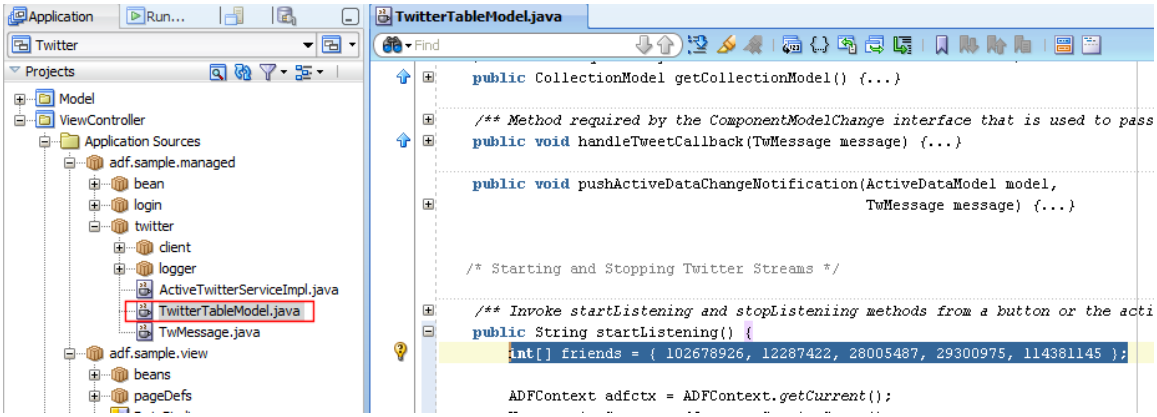
Register Twitter accounts to listen for

The users to listen for on Twitter are defined in the `TwitterTableModel.java` class.

The registration is by user Id, not the Twitter user name. To obtain the Id of a Twitter user, go to twitter.com and login. Search for the user to listen for and hover over the RSS feed icon shown on the user profile page. The RSS feed contains the user Id, which then you copy into the list of friends.



Then copy the user id into the `TwitterTableModel.java` class shown below

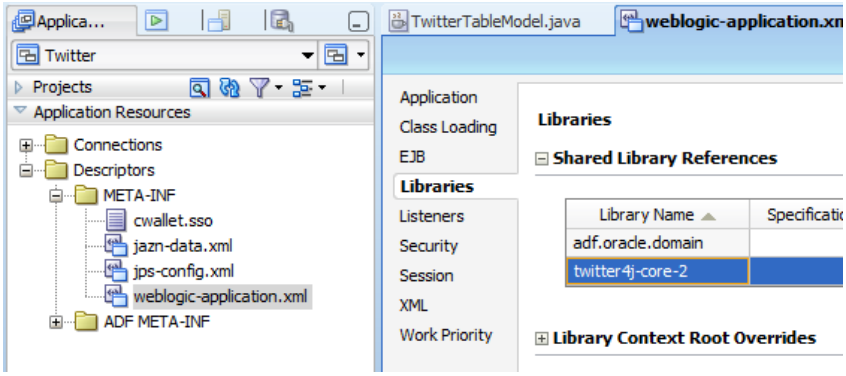


Note: The existing accounts in this class are from: Frank Nimphius, Matthias Wessendorf, Maiko Rocha and Oracle JDeveloper. All of the accounts are from Oracle JDeveloper development and management.

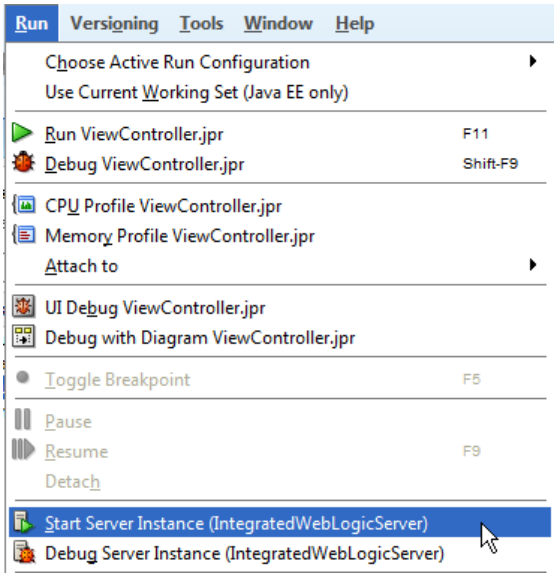
Note: I am not sure if you need to follow the accounts you want to listen for in the sample. So a save harbor is to use the Ids of friends and co-workers you are listening for anyway

Deploy Twitter4J JAR as shared library

The sample is configured to reference a shared library with the name "twitter-4j-core-2". If you deploy the Twitter4J library that comes with this sample, then you don't need to take care of providing the correct name upon deployment. It is well defined within the Twitter4J JAR file. All you need to do is to deploy the JAR.



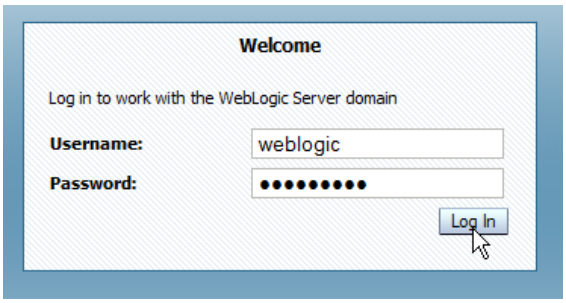
To deploy the Twitter4J JAR as a shared library, start WebLogic Server (WLS). If you run the application in the integrated WLS server, choose Run | Start Server Instance from the Oracle JDeveloper menu to start the server instance.



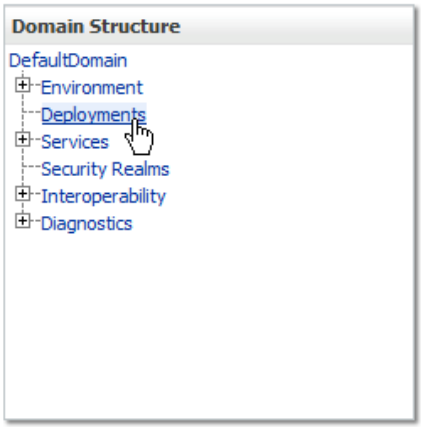
Open a browser window and type **http://host:7101/console** to start the WebLogic server console

When the console is deployed, it displays a logon screen that you connect to using weblogic/weblogic1 as username and password

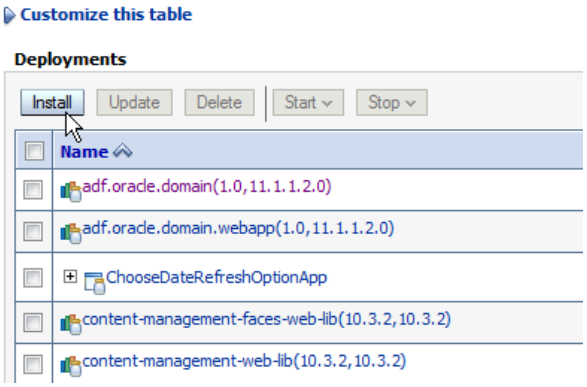
Note: production servers should have this default configuration changed. Also, starting Oracle JDeveloper 11.1.1.4 the default password policy is changed as well



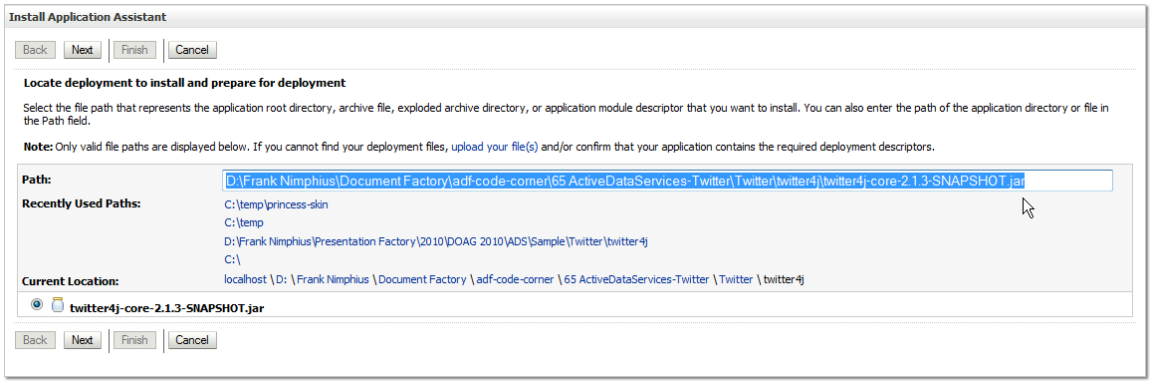
Under **Domain Structure**, select the **Deployments** node.



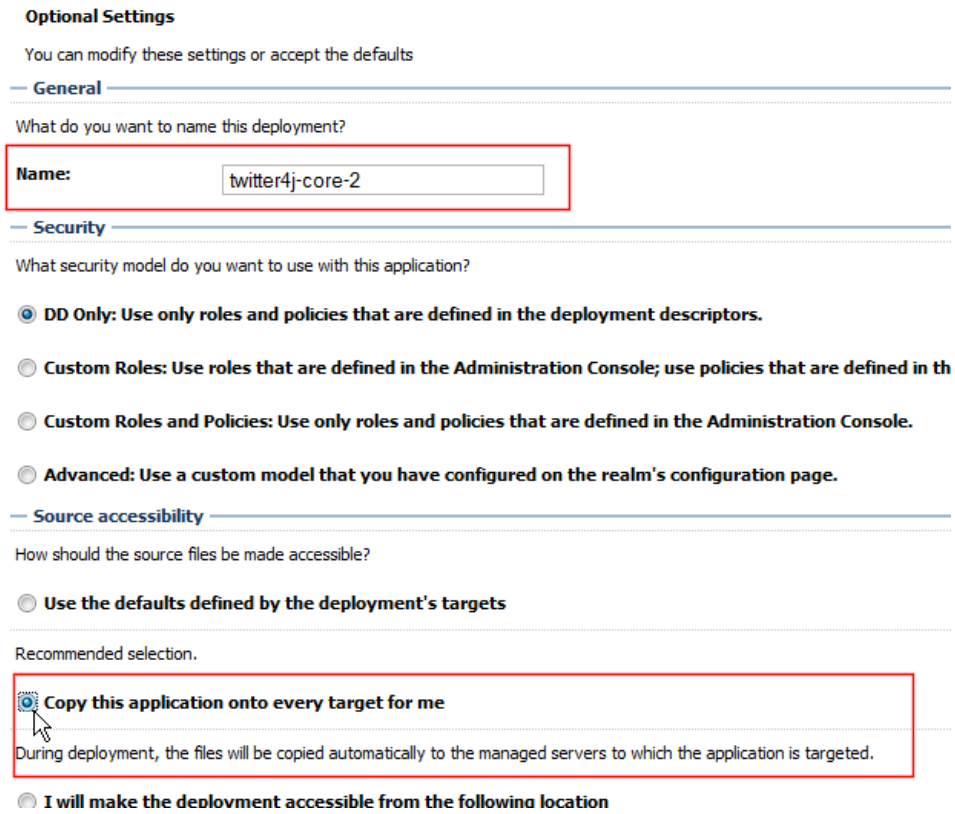
Press the **Install** button to start the library deployment



In the first screen, you can type the path of the Twitter4J JAR location into the **Path** field or navigate to it

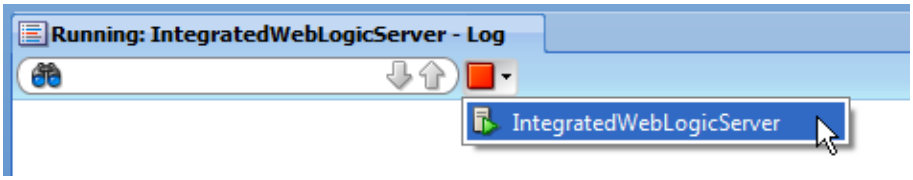


The select the JAR entry and press **Next**. You can ignore error messages if displayed in the WLS console



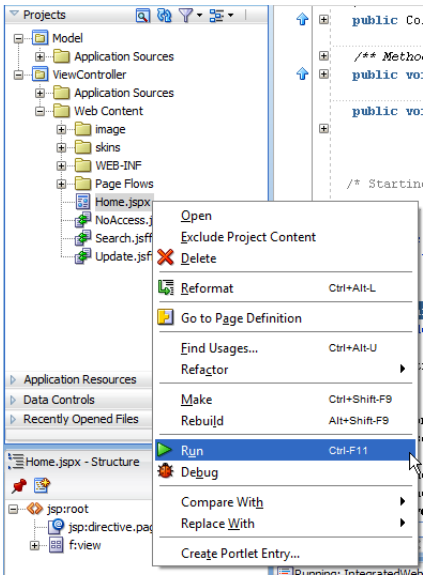
On the next screen, the **twitter4j-core-2** name is already set. Select the **Copy this application onto every target for me** option and hit **Next** or **Finish**.

Note: If the Web Logic Server console informs you to accept changes then do so. Once done, re-start the WLS server by stopping it – e.g. from the Oracle JDeveloper– and starting it up again. Do so even if the WLS console says that this is not required.



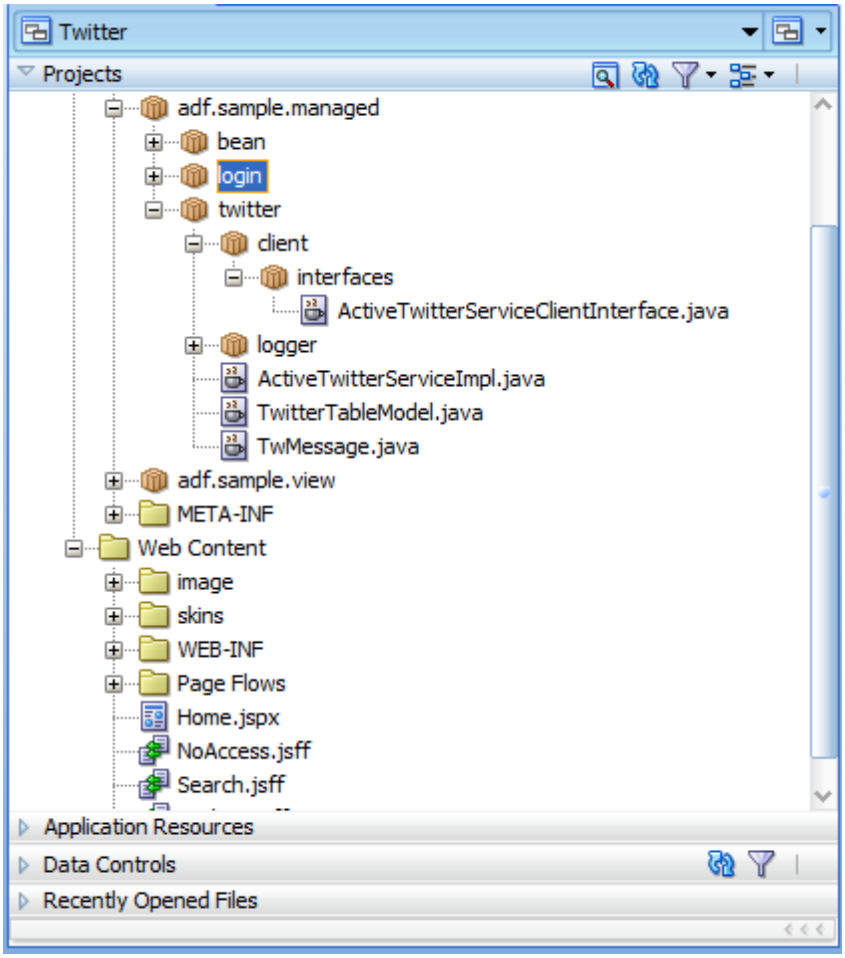
Note: If you download and deploy a newer version of the Twitter4J library, make sure its **Manfifest** file is corrected. The **Manifest** file in the JAR contains an invalid version number. Only numbers are allowed to specify a JAR version.

Run the application from the Oracle JDeveloper **Application Navigator**. Select the **Home.jspx** page and choose **Run** from the right mouse context menu.



Understanding the code

The classes related to Active Data Services in this sample are shown in the image below



ActiveTwitterServiceImpl – contains the code that interacts with Twitter through Twitter4J. It also holds the initial conversation between Maiko Rocha and me shown after application startup. This class needs to be changed only to create a different opening talk.

TwitterTableModel - This class contains the Active Data Service implementation. It decorates the ADF Faces table model and registers with the **ActiveTwitterServiceImpl** class to receive notification when a new message is posted by a friend on Twitter. The TwitterTableModel class is referenced from the ADF Faces table displaying the tweet messages.

The `TwitterTableModel` bean is configured as a managed bean in view scope because it holds the table data, which should not be re-queried with each request

```
<managed-bean id="__11">
  <managed-bean-name id="__12">tableModel</managed-bean-name>
  <managed-bean-class id="__10">
      adf.sample.managed.twitter.TwitterTableModel
  </managed-bean-class>
  <managed-bean-scope id="__9">view</managed-bean-scope>
</managed-bean>
```

In the bean constructor, the table model registers itself with the `ActiveTwitterServiceImpl` class, which is the active listener using Twitter4J.

```
public TwitterTableModel() {
  super();
  activeTwitterDataService = new ActiveTwitterServiceImpl(this);
  //get some start tweets
  tweetCollection = activeTwitterDataService.getTweets();
  Collections.reverse(tweetCollection);
  //create Trinidad collection model
  tableModel = new SortableModel(tweetCollection);
}
```

For every new Twitter message, `ActiveTwitterServiceImpl` calls the `TwitterTableModel` class's `pushActiveDataChangeNotification` method to update the table using ADS

```
public void pushActiveDataChangeNotification(ActiveDataModel model,
                                             TwMessage message) {
        TwitterActiveDataModel adm = (TwitterActiveDataModel)model;
        adm.increaseChangeCounter();
        //perform update update
        ActiveDataUpdateEvent updateEvent = null;
        ArrayList<String> attributeList = new ArrayList<String>();
        //create the list of attributes to update in the table view
        attributeList.add("profileImageUrl");
        attributeList.add("user");
        attributeList.add("location");
```

```
        attributeList.add("message");
        attributeList.add("created");

        String[] attributeListArray = new String[attributeList.size()];
        ArrayList<Object> updateValueList = new ArrayList<Object>();
        updateValueList.add(message.getProfileImageUrl());
        updateValueList.add(message.getUser());
        updateValueList.add(message.getLocation());
        updateValueList.add(message.getMessage());
        updateValueList.add(message.getCreated());

        //call the ADS proxy framework to add a new message as the
        //first row in the table
        updateEvent =
                ActiveDataEventUtil.buildActiveDataUpdateEvent(
                            ActiveDataEntry.ChangeType.INSERT_BEFORE,
                            model.getCurrentChangeCount(),
                            new Object[]{0},
                            null,
                            attributeList.toArray(attributeListArray),
                            updateValueList.toArray());
      //call the Active Data Model internal class
      adm.notifyDataChange(updateEvent);


  //synchronize the table data collection with the current state of the
  //table model
  tweetCollection.add(0,message);
  tweetCollection = activeTwitterDataService.getTweets();
}
```

TwMessage – An object representation of the Twitter message. A row in the ADF Faces table represents exact one instance of this object

The most interesting class for you to study is **TwitterTableModel.** The class contains an inner class **TwitterActiveDataModel** that extends the ADF Faces Active Data Services **BaseActiveDataModel**.

Download

The Oracle JDeveloper 11.1.1.3 workspace configured in this document can be downloaded as sample 065 from ADF Coder Corner:

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

What else do you need to know?

You should be aware that this sample only implements one aspect of Active Data Services, which is the creation of a new object. In addition ADS can be used to update screens for delete and update operations as well, which you find information about in the collateral referenced at the end of this document.

The Twitter sample code in the documented work space can be changed to adapt to any messaging service that provides a streaming API for Java.

The default push protocol used by the sample is streaming. To change this to poll or long-polling, edit the adf-config.xml file in there. However, before changing the settings in this file, I recommend reading the ADS documentation that is available.

Last but not least – this sample is a demo to get you started or for you to be able to demo Active Data Services. There are no plans to extend this demo to be a complete Twitter client that not only allows to read but also to write tweets.

RELATED DOCOMENTATION

☒	Ch 42 - Oracle Fusion Middleware Fusion Developer's Guide for Oracle Application Development Framework http://download.oracle.com/docs/cd/E15523_01/web.1111/b31974/adv_ads.htm#BEIDHJFD
☒	Ch 20 – Fusion Developer Guide, Frank Nimphius, Lynn Munsinger (McGraw Hill) – http://www.mhprofessional.com/product.php?cat=112&isbn=0071622543
☒	