IBM Integration Bus

MobileFirst Integration

Featuring:

- Explore REST API service in Integration Bus Toolkit
- Explore a Mobile application in MobileFirst Studio
- Connect the Mobile application to the REST API service
  - Create an adapter
  - Test Mobile application in Mobile Browser simulator

June 2015

Hands-on lab built at product
Version 10.0.0.0

Content:

1. Introduction

IBM Integration Bus V10 allows easy integration with the IBM MobileFirst platform. It enables the building of mobile applications for a number of mobile Operating Systems. This integration uses an adapter which provides access to Web and REST services running on IBM Integration Bus.

The IBM MobileFirst platform simplifies mobile application development and utilizes features such as security and administration for mobile applications.

This lab will demonstrate a typical use case for integrating a mobile app with a REST API service running on IBM Integration Bus. The general steps in this use case are outlined below:

- Develop an IBM Integration Bus (IIB) REST API service, for querying data from a database;
- Develop a MobileFirst Application which will invoke the IIB REST API;
- Develop a MobileFirst adapter in the MobileFirst Studio which will connect the Mobile application to the IIB REST API Service.

The focus on this lab is to demonstrate the steps required for creating the MobileFirst adapter and confirming successful integration with the IIB application. Creating the IIB REST API service is the subject of a separate lab.

1.1 Lab preparation

To run this lab you will use IBM MobileFirst Studio. If you are using the pre-supplied VMware image, this will already be available. The image contains an installation of Eclipse Juno v4.2.2 and IBM MobileFirst v6.3.

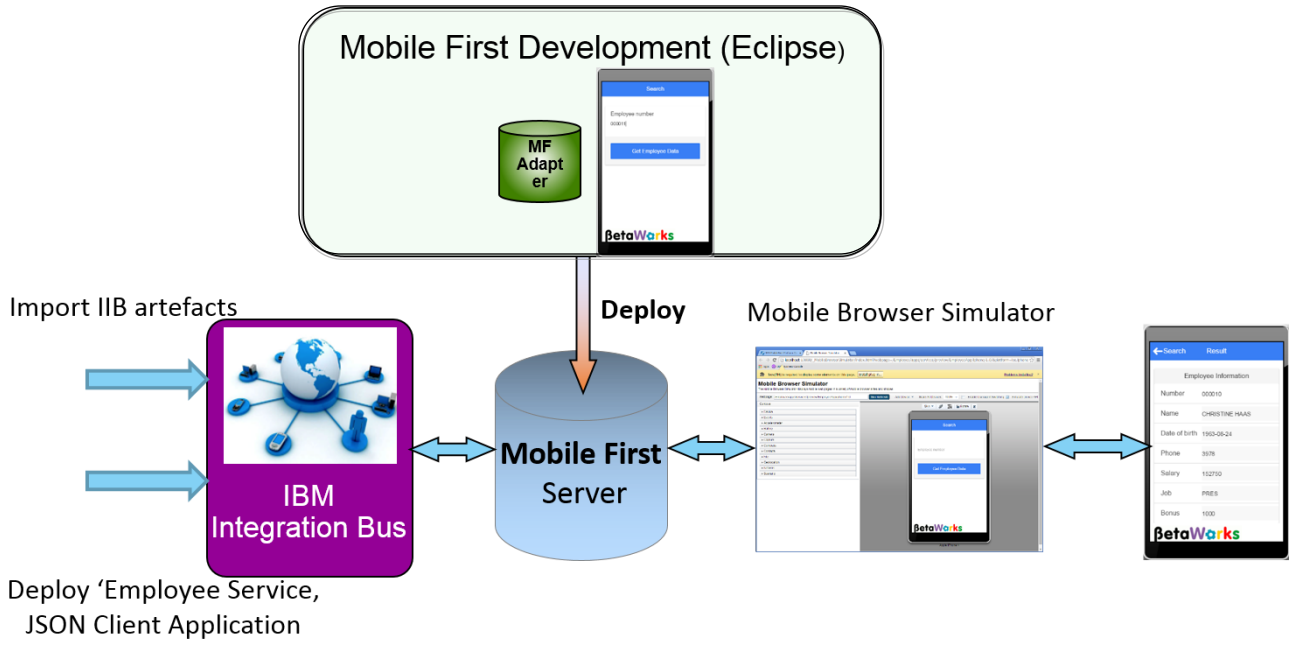
This lab is based on the solution of the REST API Service. This lab uses an IIB node called RESTNODE. You should have it already created. If not, refer to the Appendix at the end of this lab guide.

Start the RESTNODE.

1.2 Outline of Lab

This lab will show you the following functions:

- Explore REST API service in Integration Bus Toolkit
- Deploy REST API service
- Explore a Mobile application in MobileFirst Studio
- Connect the Mobile application to the REST API service
  - Create an adapter
  - Test Mobile application in Mobile Browser simulator
- Explore Error Handling



2. Prepare the Provider applications

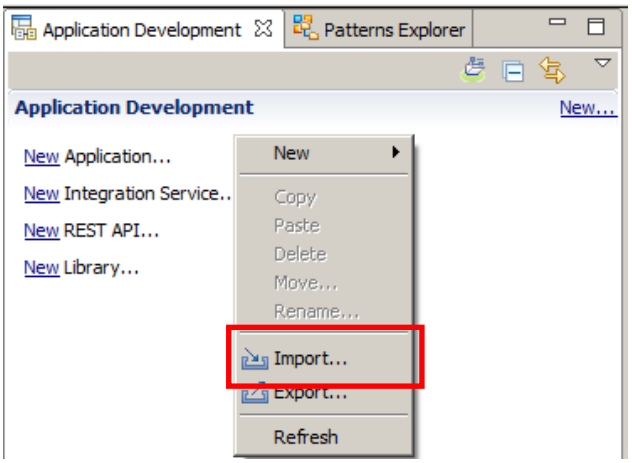
In this part of the lab exercise, you will import the solutions IIB shared library and REST API Service.

2.1 Application Deployment in IBM Integration Bus

1. To avoid naming clashes with earlier labs, this lab will be developed using a new workspace.

   If you already have a workspace open, click File, Switch Workspace. Give the new workspace the name "MobileIntegration", or similar.

2. Right-click in the 'Application Development' pane and click on 'Import':

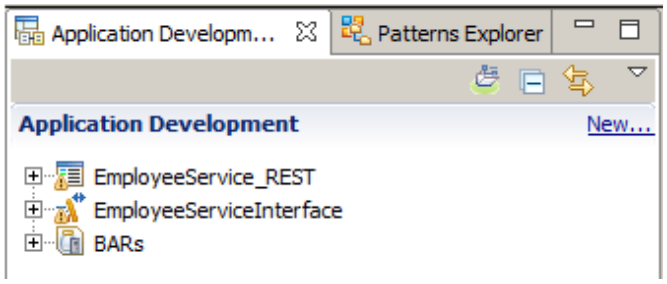


   Import the following Project Interchange (PI) files:

   - **EmployeeServiceInterface.V10.zip** from **C:\student10\Integration_service\solution** folder;
   - **EmployeeService_REST.V10.zip** from **C:\student10\REST_service\solution** folder;

   Note: Make sure that all projects in these PI files are selected for import.

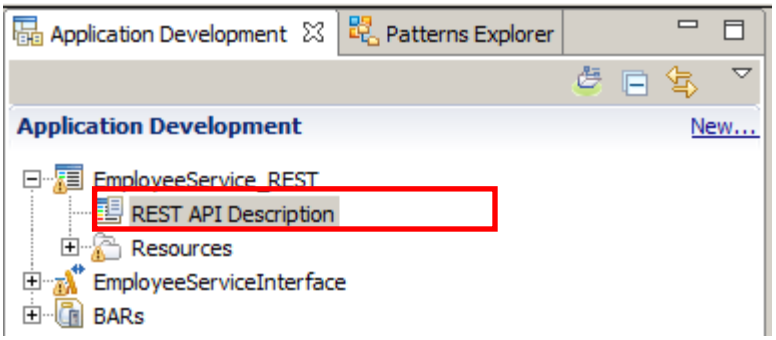
3. Once completed, you should have in your workspace the Employee REST API Service and a shared library that is referenced by the service.



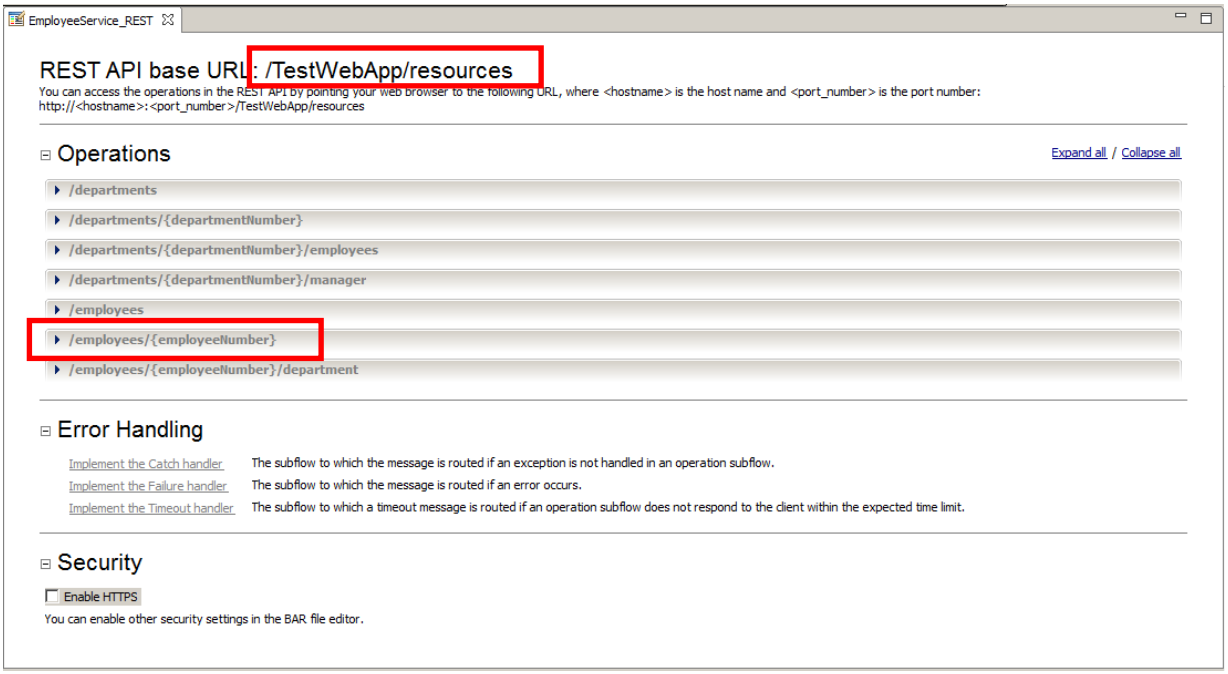
2.2 Application Deployment

In this section, you will view the IIB REST API service and where in the Mapping node one of the error messages is generated. This will be used later in this lab guide.

1. Expand the REST API Service and double-click on REST API Description:



2. The details of the REST API Service are shown:



On the top of the page you will see the REST API base URL as '**/TestWebApp/resources'** which identifies the access path to the service over HTTP.

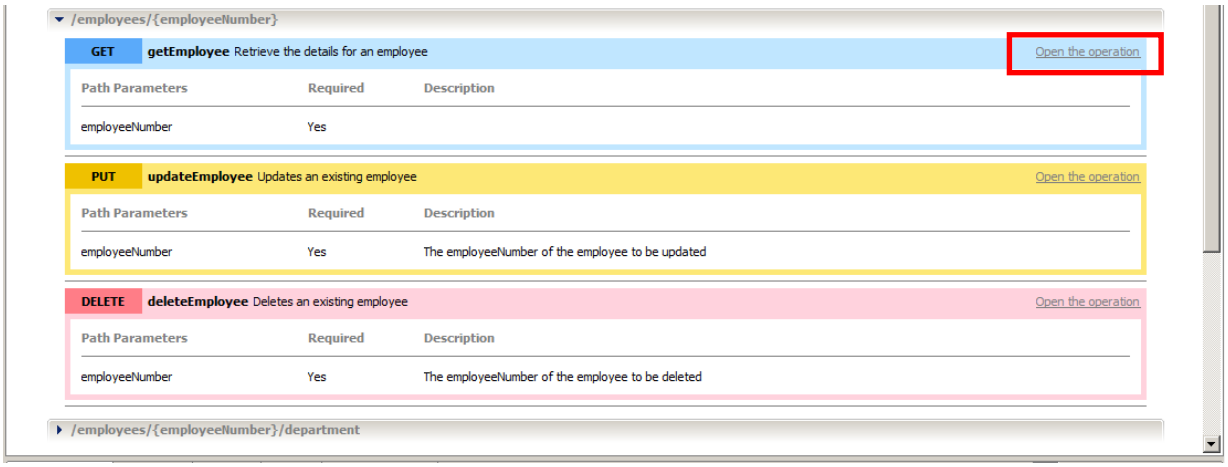
In addition, you can see a number of operations. Each operation is accessed by adding the specified operation extension to the base URL and adding an input parameter (if required). For example, for our Mobile app we will be using the **/employees/{employeeNumber}** operation providing their employee number.

This operation has already been implemented in this provided service.

Expand the **/employees/{employeeNumber}** operation by clicking it.

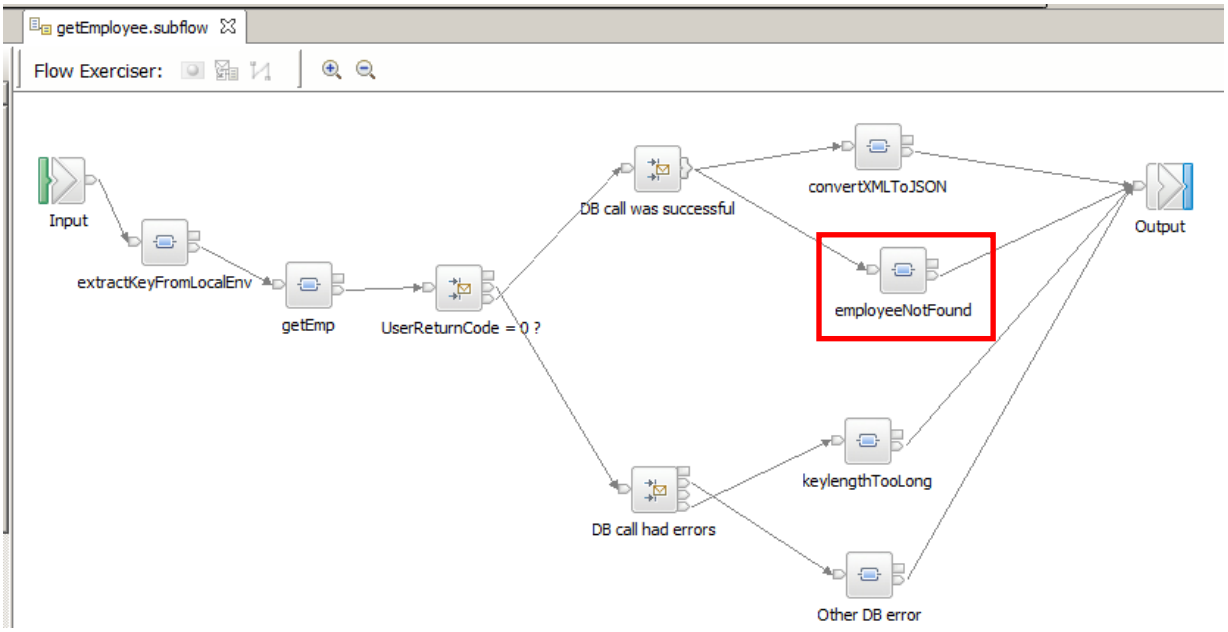
3. You will see that there are three operations – Get, Put and Delete.

   For the GET method, click 'Open the operation':



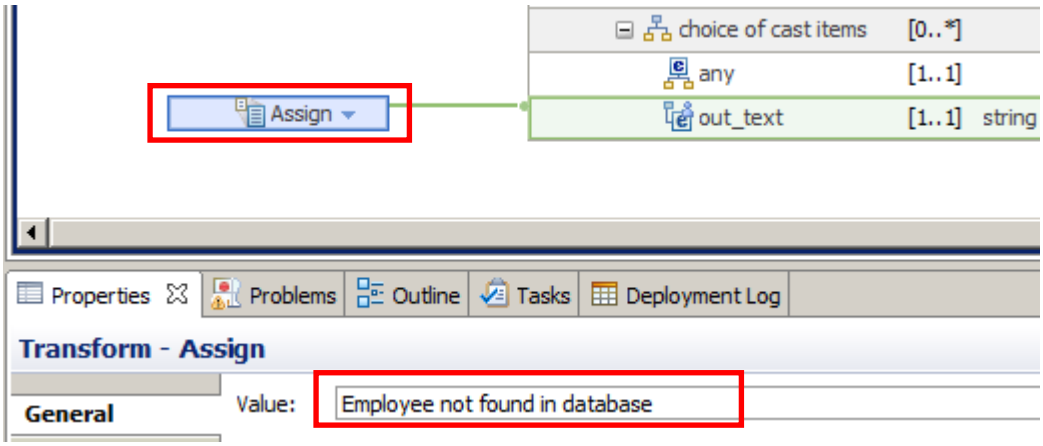
4. This opens the implementation subflow, which is probably familiar.

   Double-click to open the mapping node 'employeeNotFound'.

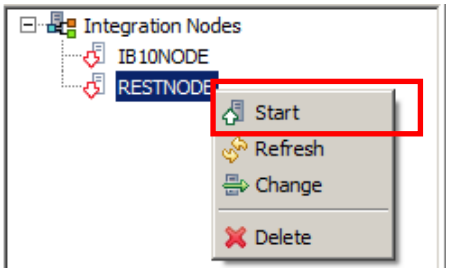


5. Once the map has opened, click the second 'Assign' transformation.

   You will see the message that will be returned from IIB if the employee is not found in the database. This will be tested in a later section of this lab.



6. In the IBM Integration Bus Toolkit, start the RESTNODE Integration Node (if not started):

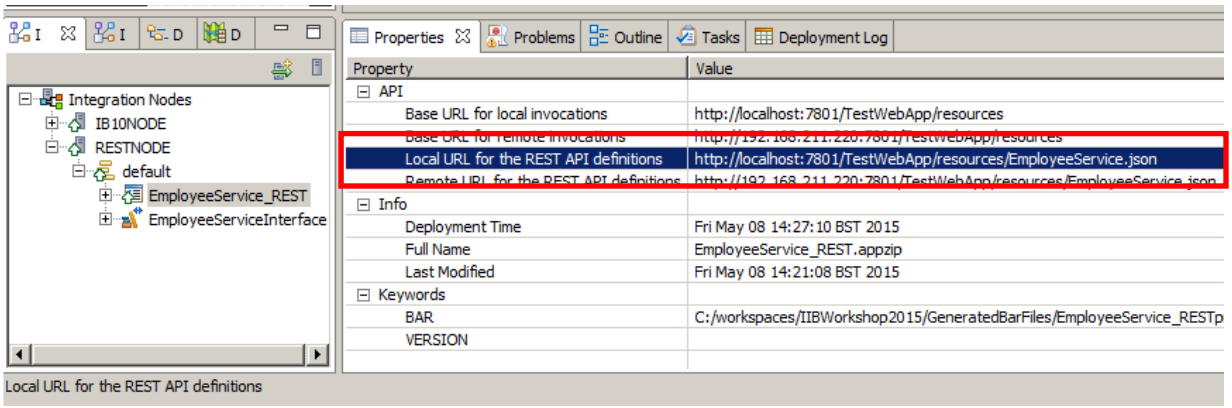


7. Deploy your workspace resources by 'drag-and-drop' to the default integration server on RESTNODE in the following order:

   - Shared Library – EmployeeServiceInterface.
   - EmployeeService_REST.

   The order of deployment is important because the REST API service references the shared library and the deployment will fail if it does not find the shared library in the runtime when they are deployed.

8. Once the deployment has completed, under the 'default' Integration server, click the deployed REST API Service.



In the Properties tab you can see details about the deployed REST API Service including URLs for remote and local invocation. In addition, a URL to the REST API definitions is provided.

Write down the port number of the Integration Server, to which the REST API Service has been deployed (7801 in the screen capture above), as it will be needed later in the lab.

REST API port number: _____

3. IBM MobileFirst Integration with IBM Integration Bus

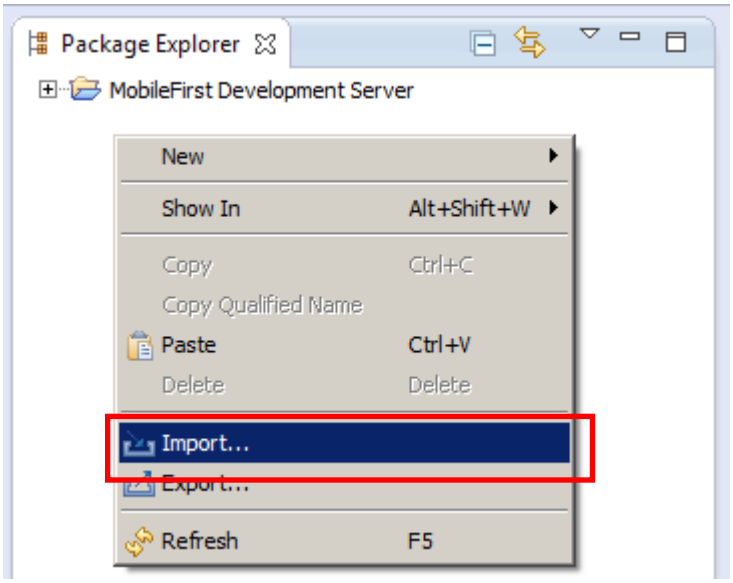
In this part of the lab, you will build an integration adapter in the IBM MobileFirst Studio, which will connect to your application running on IBM Integration Bus.

3.1 Import the Project in the MobileFirst Studio

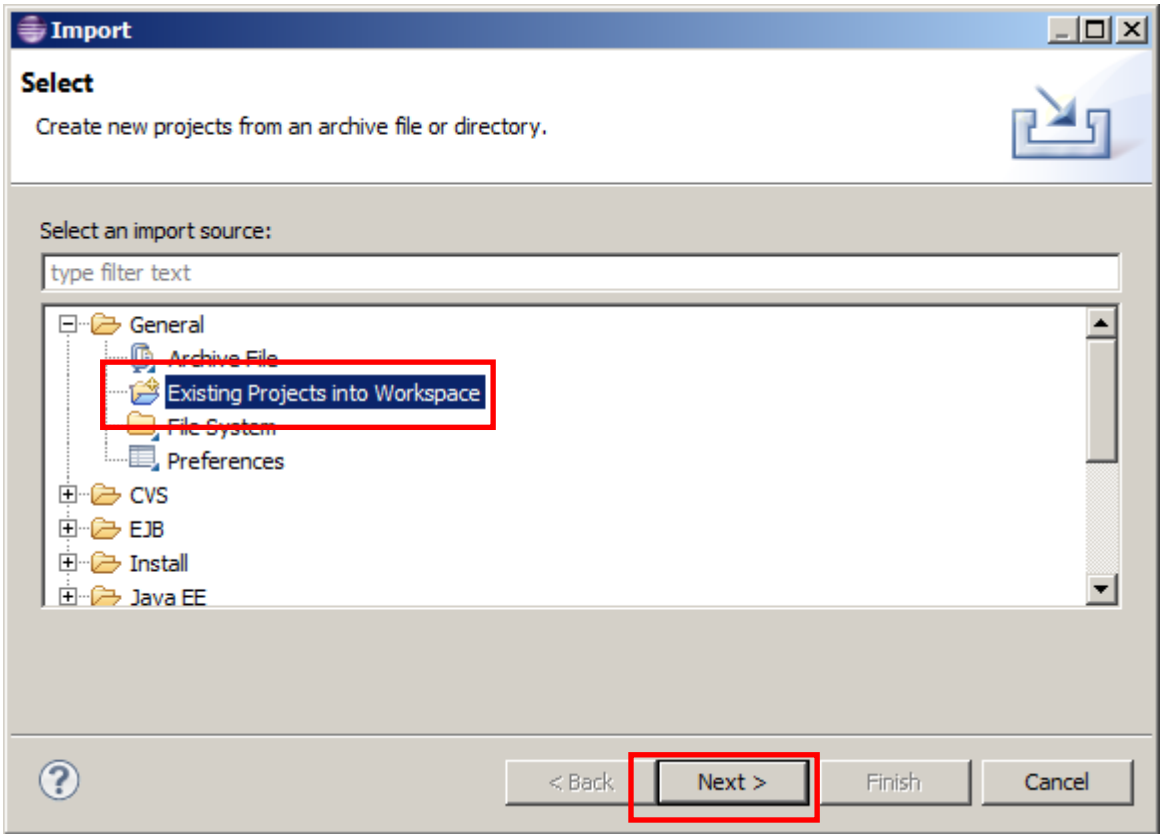
1. Open the MobileFirst Studio (from the Windows Start menu).

2. In this workshop the Mobile Application has already been built. The only missing part is an adapter, required to connect to the Integration Node and the IIB REST API Service.

   In MobileFirst Studio right-click in the Package Explorer pane and select Import.

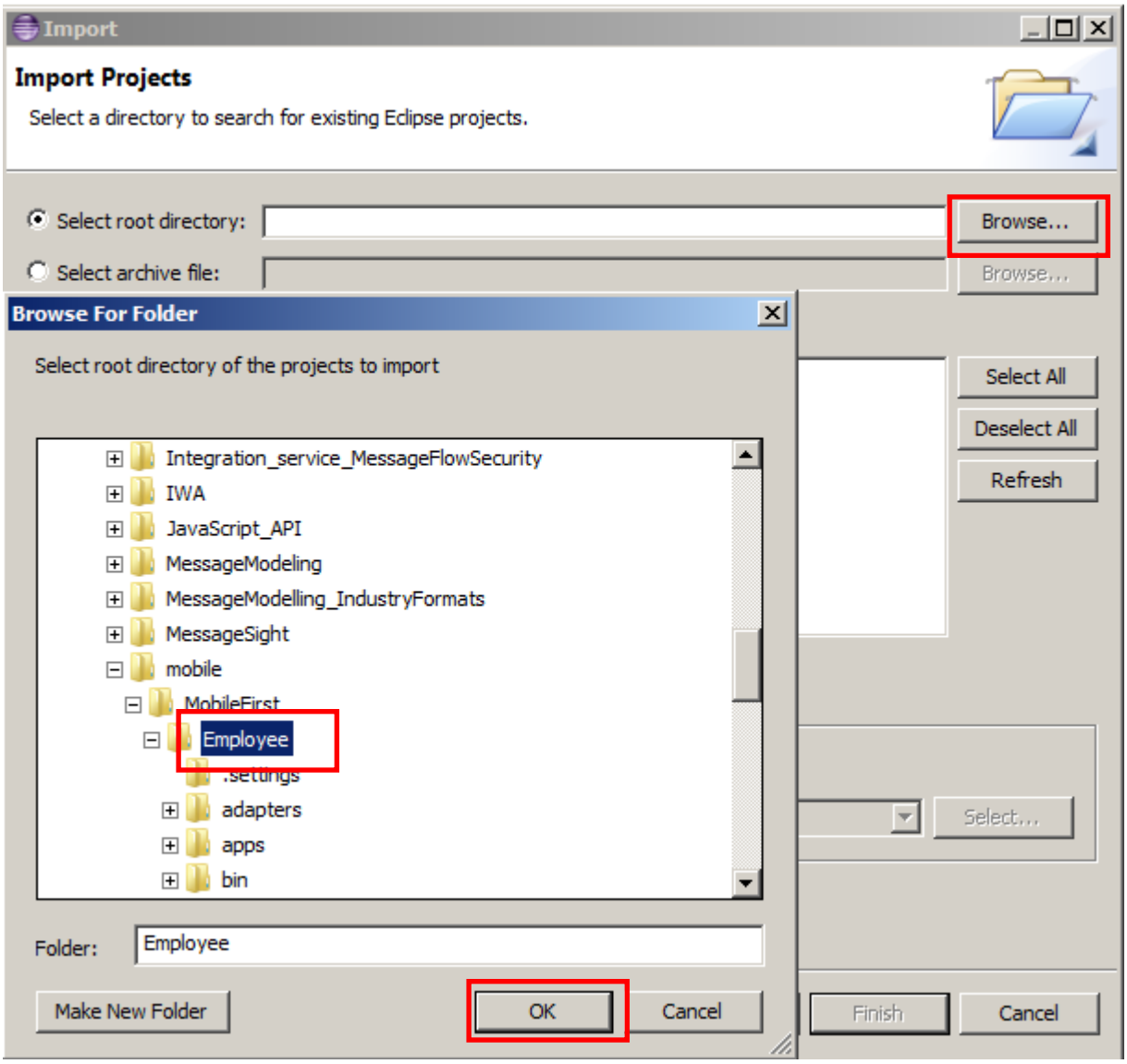


3. In the opened dialog, expand ‘General’ and select ‘Existing Projects into Workspace’. Click ‘Next’.



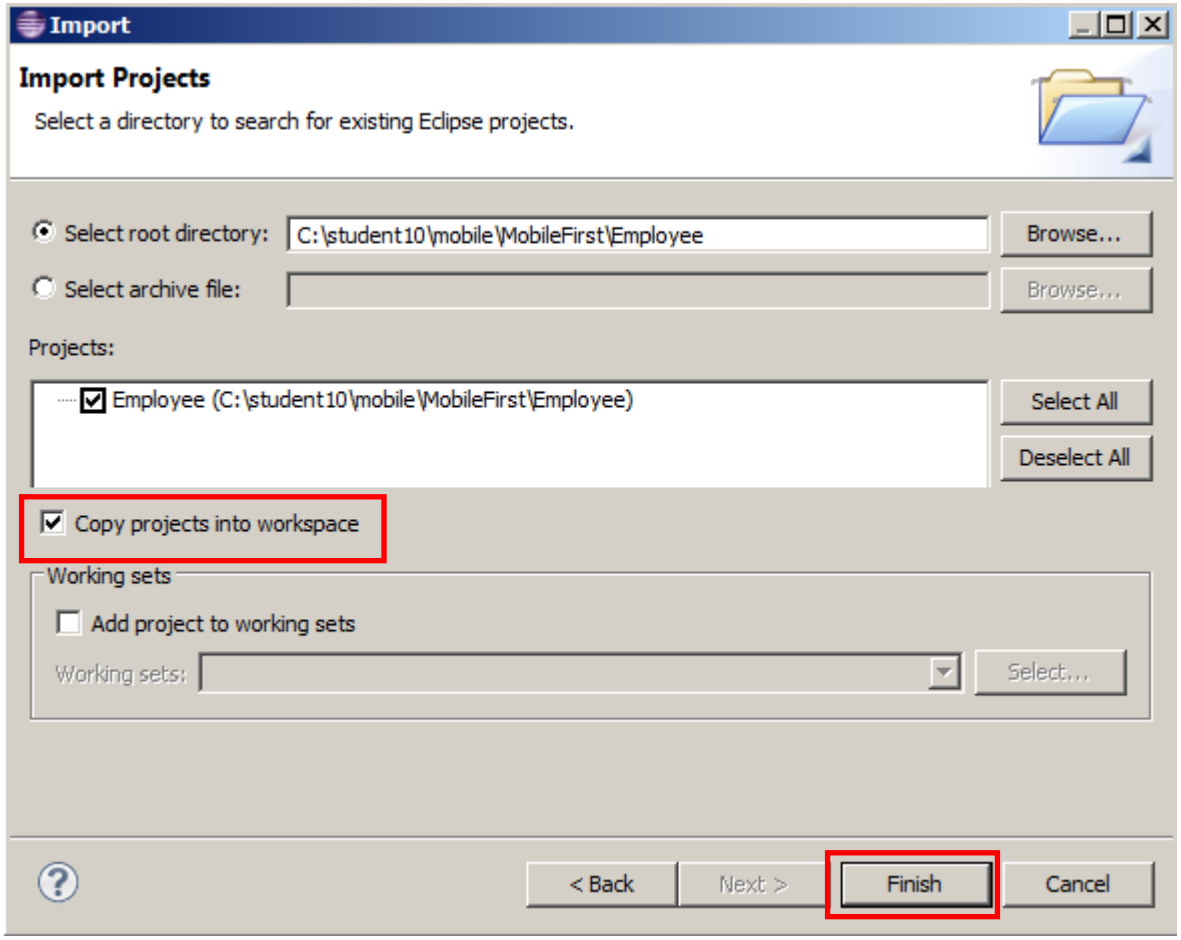
4. In the root directory, click 'Browse' and then select 'Employee' folder in **C:\student10\mobile\MobileFirst**.

   Click 'OK'.

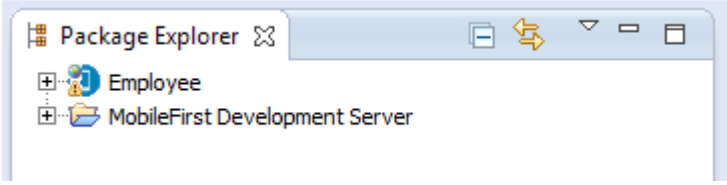


5. Tick the box next to 'Copy projects into workspace and click 'Finish' to complete the import of the Project.

.



6. Once the import has completed you will see the 'Employee' project in the Package Explorer pane.

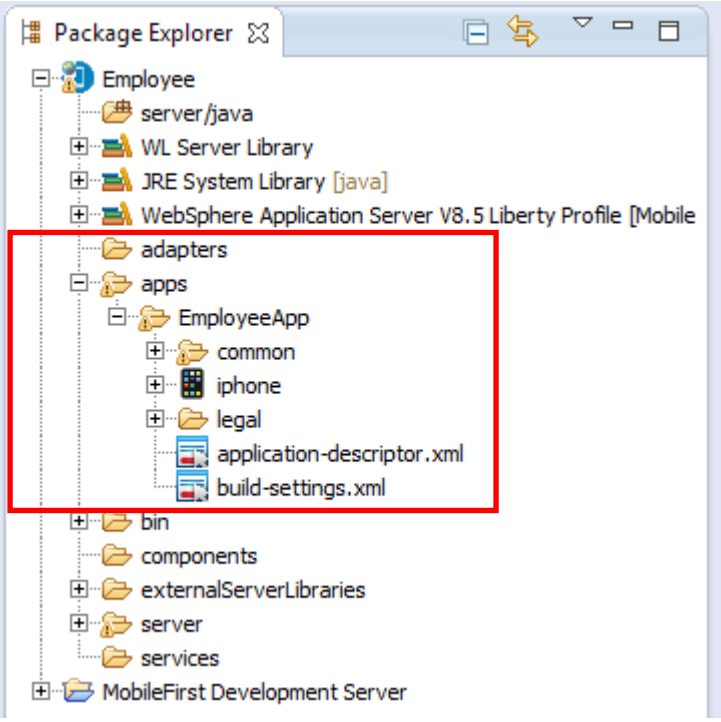


3.2 Mobile Application for querying database data

The Mobile application that will be used in this lab has already been built. The application is the front end in querying the database for an employee's record. This would be built by a MobileFirst developer, based on the data that needs to be send to the REST API Service and the returned records information.

You will first explore the main components of the project to get familiar with its structure.

1. Expand the Employee project, then 'apps' and 'EmployeeApp':

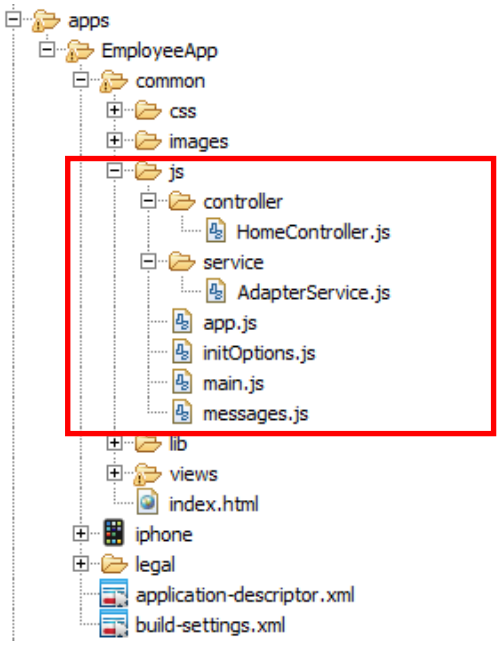


The directory contains 3 folders – common, iphone and legal:

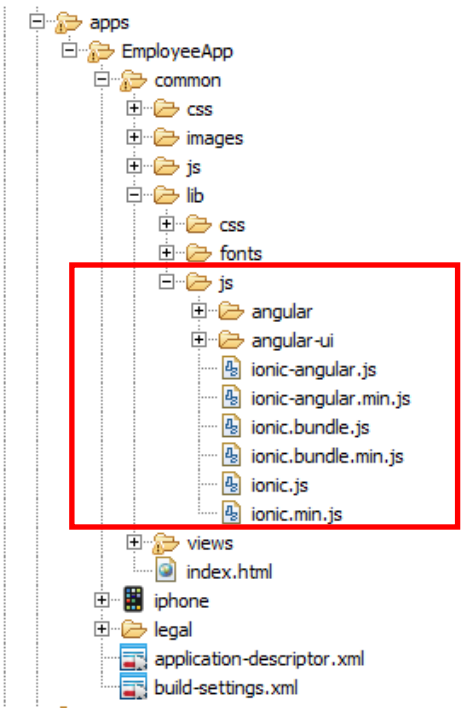
- common – contains the main artefacts for the Mobile app, which are required for any OS
- iphone - artefacts for the mobile app deployed on iOS
- legal – IBM legal agreements

These folders are created from the MobileFirst Studio when the MobileFirst developer generates the project. Then they are customized by including resources for the Mobile application and its logic (i.e. HTML files, JavaScript files, libraries and images).

2. Expand the ‘common’ folder, then ‘js’ and all the subfolders. The JavaScript file ‘HomeController.js’ is the AngularJS controller, which is used to update the data to the HTML pages. The ‘app.js’ file contains the logic for the transition between the different views of the mobile app.

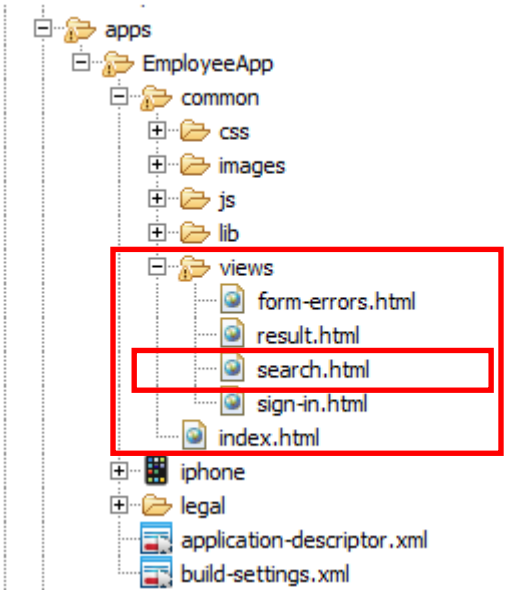


3. If you expand the ‘lib’ subfolder and then ‘js’, you will see the required AngularJS and Ionic libraries included in the project. These libraries provide the templates for the mobile pages views when the Mobile application is deployed.



4. Under the 'views' folder are the Mobile app HTML pages. You will see that there are four HTML pages defined. However, for this lab you will use only two of them:
   - 'search.html' - which represents the view to search for the Employee records;
   - 'result.html' - which is the view for displaying the returned data from the database.

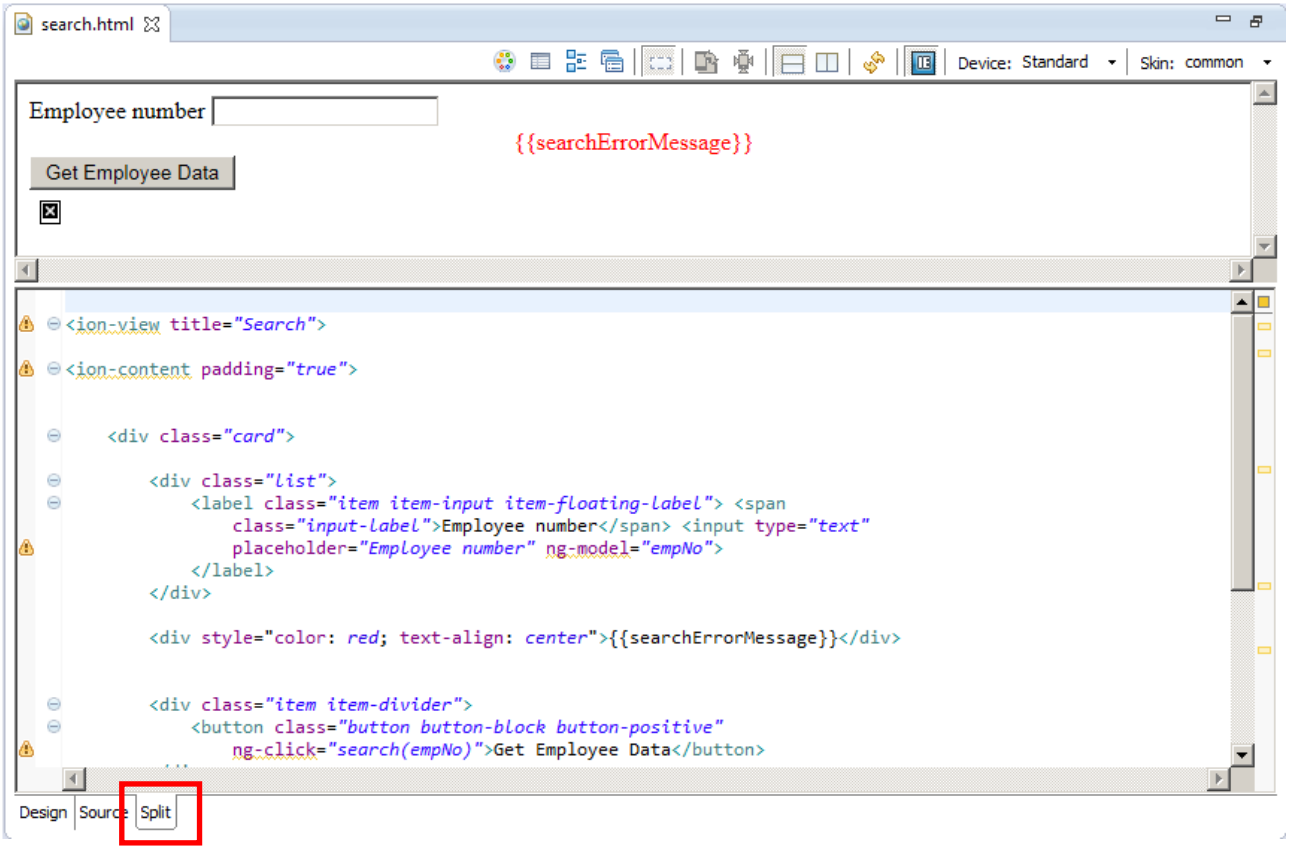
   Double-click 'search.html':



5. You may see a popup message saying 'Some pages may not render correctly when using Internet Explorer as the embedded browser,

   Click OK to dismiss the message.

6. In the editor, click the 'Split' tab (may be already on focus).



The editor is now open and the screen is split – in the upper part you can see the 'design', while in the bottom is the HTML code for this view.

The Mobile application's HTML pages have been created by the MobileFirst developer, based on the requirements from the Integration developer.

Please note that the design that is shown in the MobileFirst Studio editor is not the exact presentation of how the page will look on the mobile device. This is because the Mobile app is using the Ionic framework, which currently does not render completely in the MobileFirst Studio editor. You will see the full app later in the lab using the Mobile Simulator.

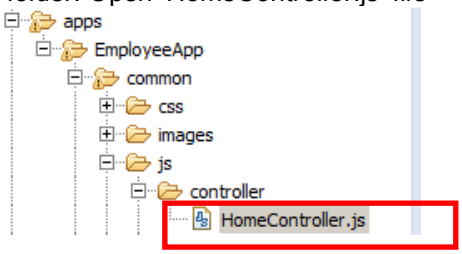
When finished reviewing, close the file without making any changes.

3.3 Data definition in 'HomeController.js'

The 'HomeController.js' contains a JavaScript construction function which is used in Angular.JS to augment an Angula.JS object that refers to the application model. This controller is created by the MobileFirst developer.

In the 'HomeController.js file, you can see the JSON output data structure.

1. Expand the 'common/js/controller folder. Open 'HomeController.js' file



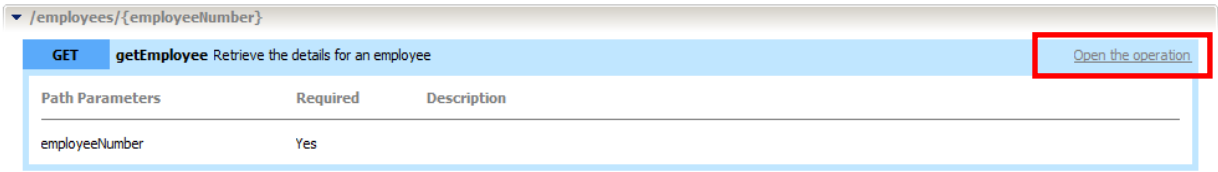
2. In this controller an 'EmployeeError' variable is included to provide you with the error message returned from Integration Bus.
   The Error message is generated in the IIB REST API Service if the employee does not exist, or if the employee number requested has more than six numbers.
   The IIB error message from IIB REST API service is '**data.out_text**'

3. The EmployeeResponse JSON data structure follows the error handling code:

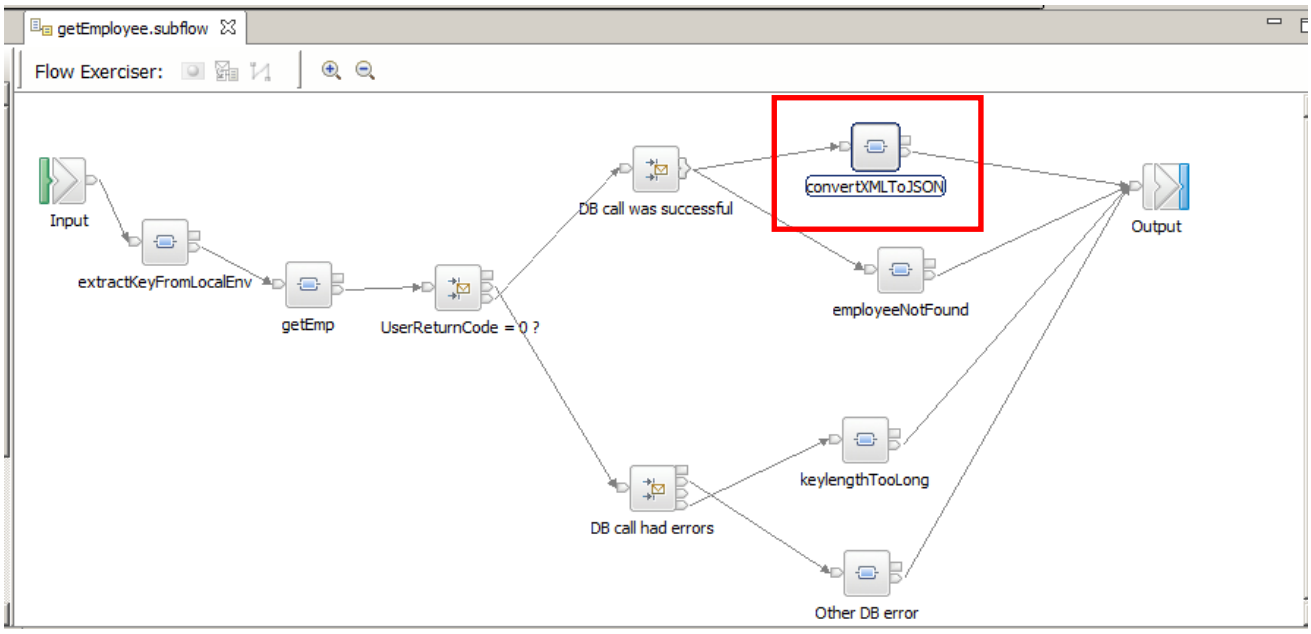
```
search.html    HomeController.js
            console.log("Got data");

            //Test got valid response
            //if not valid stay on page and produce error message

            EmployeeError = data.out_text;
            $scope.searchErrorMessage = EmployeeError;
            $scope.$apply();

            EmployeeResponse = data.EmployeeResponse.EMPLOYEE;
            $scope.name = EmployeeResponse.FIRSTNME + " " + EmployeeResponse.LASTNAME;
            $scope.dob = EmployeeResponse.BIRTHDATE;
            $scope.phone = EmployeeResponse.PHONENO;
            $scope.salary = EmployeeResponse.SALARY;
            $scope.job = EmployeeResponse.JOB;
            $scope.hiredate = EmployeeResponse.HIREDATE;
            $scope.bonus = EmployeeResponse.BONUS;
            $scope.searchErrorMessage = "";

            $state.go("result");

            busyIndicator.hide();

        },

        function(error) { //fail
```

3.4 IIB data structure

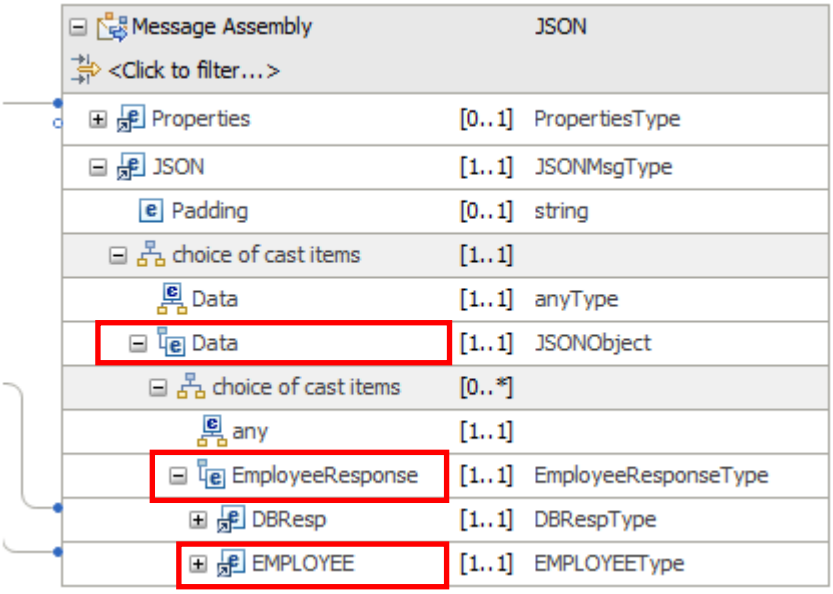
1. Back in the IIB Toolkit, the JSON output data structure was generated by the IIB REST API Service. To see this, open the operation 'getEmployee' in the IIB Toolkit:



2. Open the 'convertXMLToJSON' map:



3. Expand the output message assembly on the right hand side:



You will see the JSON object data tree as highlighted above as:

data.EmployeeResponse.EMPLOYEE

Look back in the 'HomeController.js' file to see where it is used.

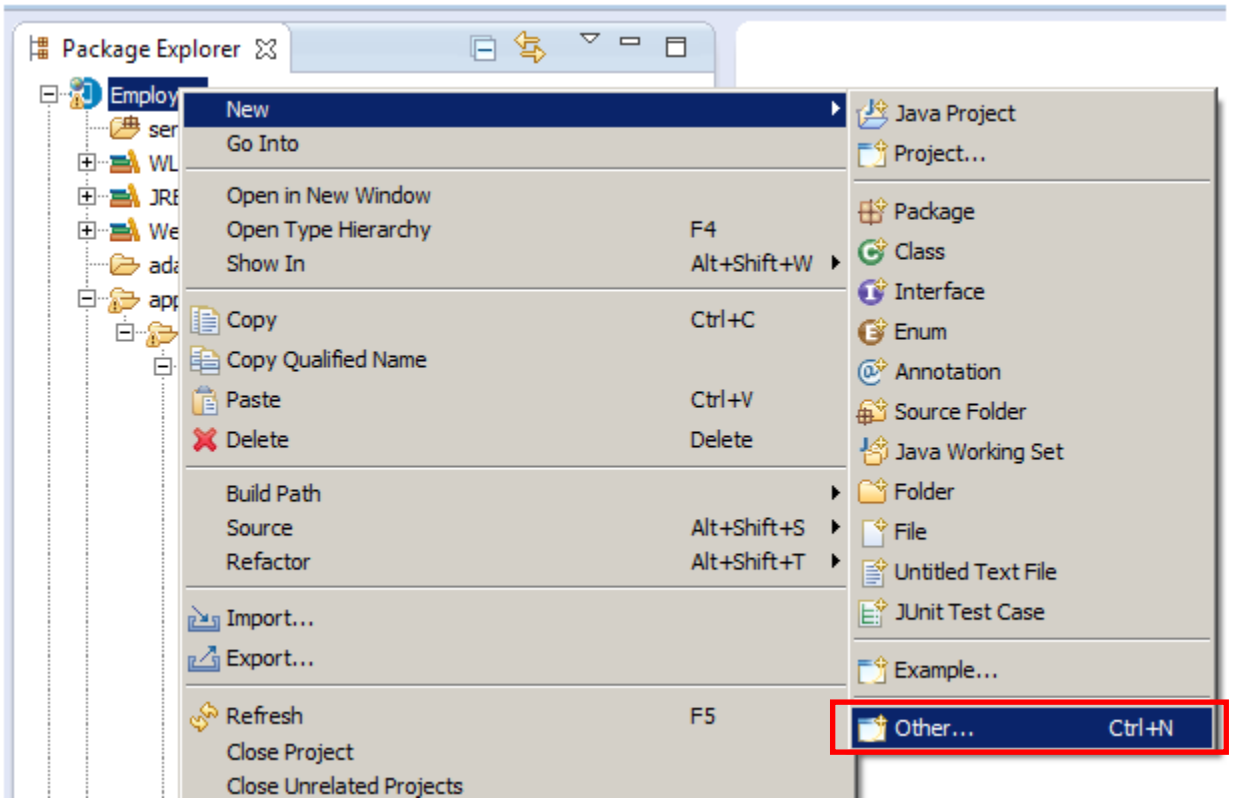
.

4. Close the map and the message flow

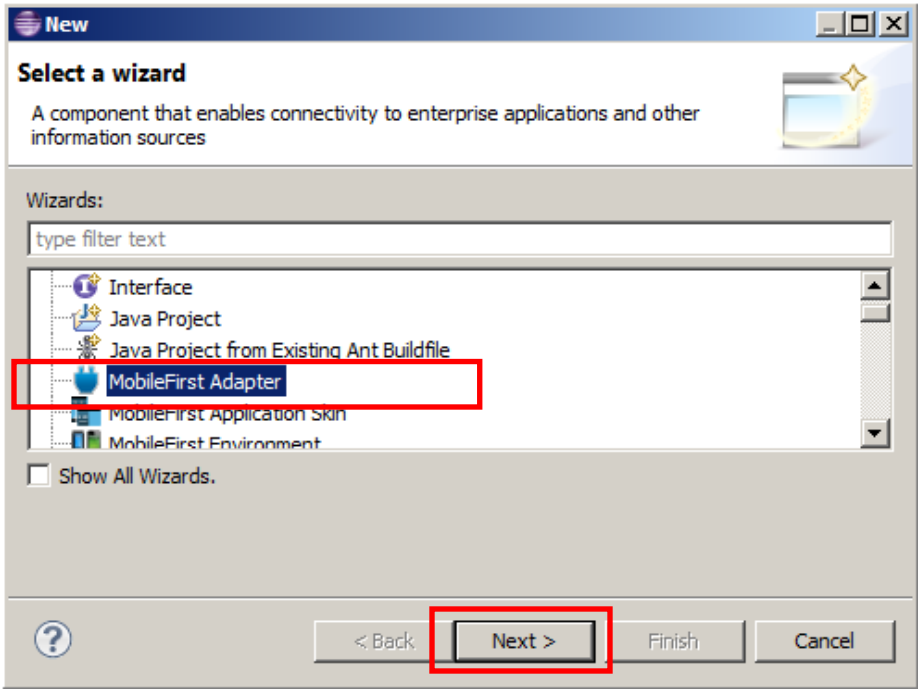
3.5 Generate the MobileFirst Adapter

In this part of the lab you will generate the adapter, which will enable the Mobile app to connect to the application, running on the Integration Node.

1. In the Mobile First Studio, right-click 'Employee' then 'New' -> 'Other'



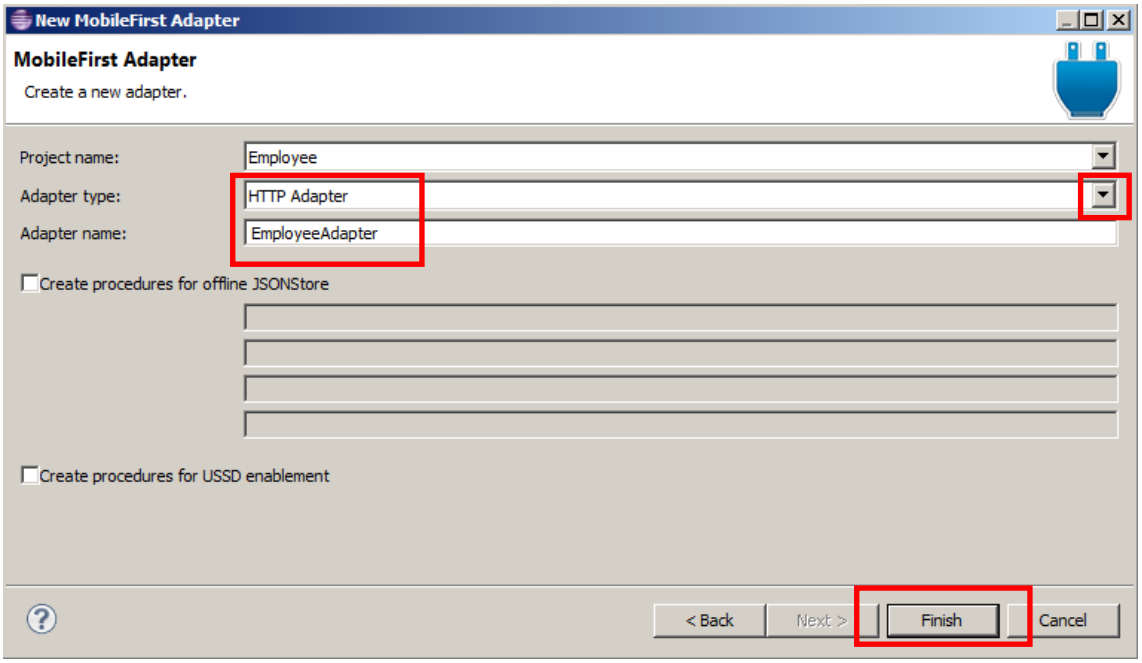
2. From the dialog, select 'MobileFirst Adapter' and click 'Next':



3. Name the adapter 'EmployeeAdapter' (case sensitive) and from the drop-down menu for the Adapter Type, select 'HTTP'.

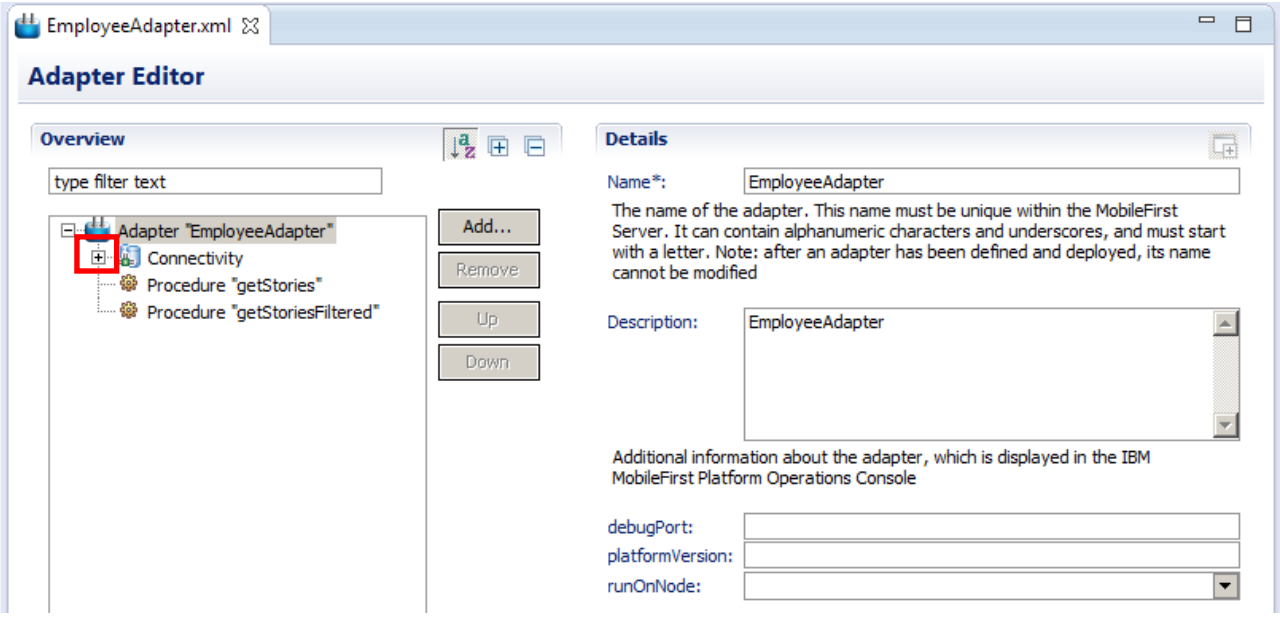
   Note: Please make sure that the name of the adapter is 'EmployeeAdapter' because this is the name used in the Mobile app for invoking the adapter.

   Click 'Finish':



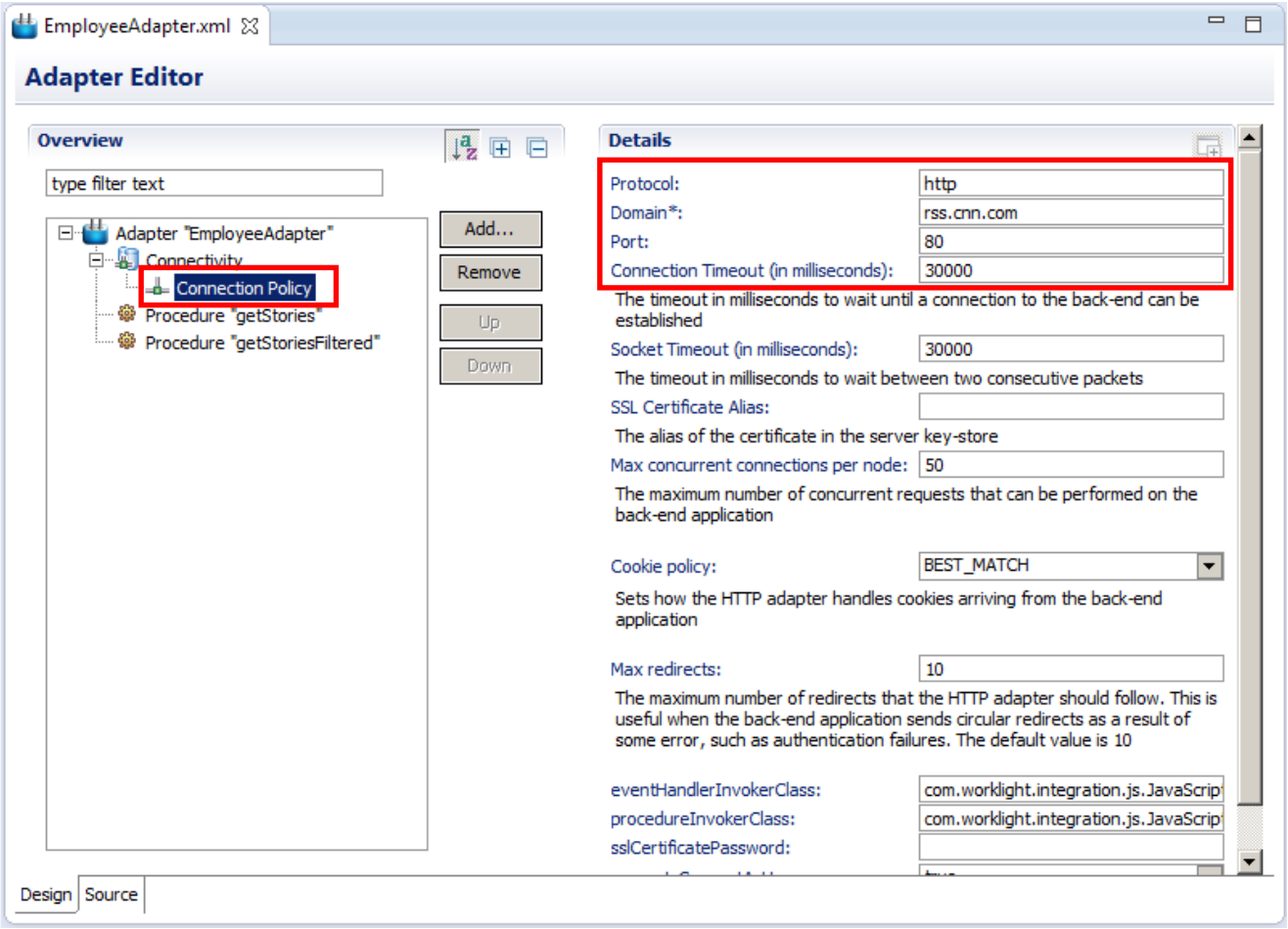
4. The ‘EmployeeAdapter.xml’ file is automatically open, wh. In the file you will make changes in order to allow connection to the REST API Service on IIB. You can also add a description if needed.

   Expand ‘Connectivity’.



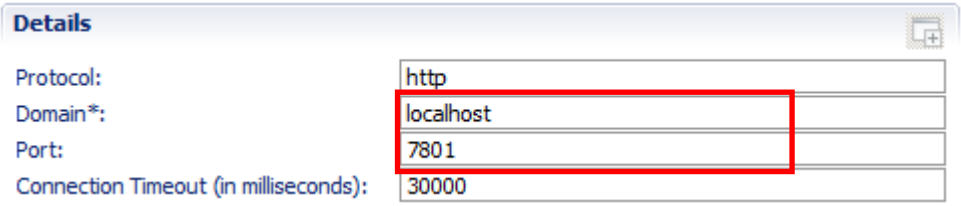
5. Expand 'Connectivity' and click 'Connection Policy'.

   In the Details pane, you will see the policy's details and parameters.

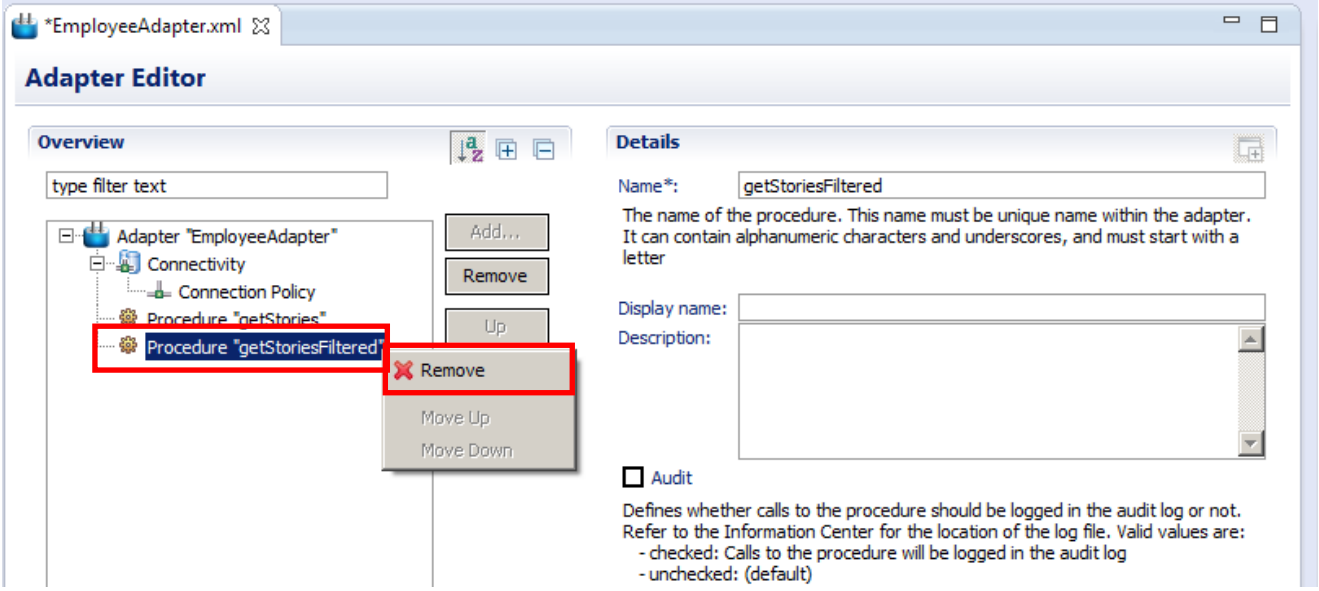


6. For this adapter, change the 'Domain' and 'Port' as follows.
   Change the following details as below:

   Domain: **localhost**
   Port: **7801 (or the one you wrote down in step 2.2.8)**



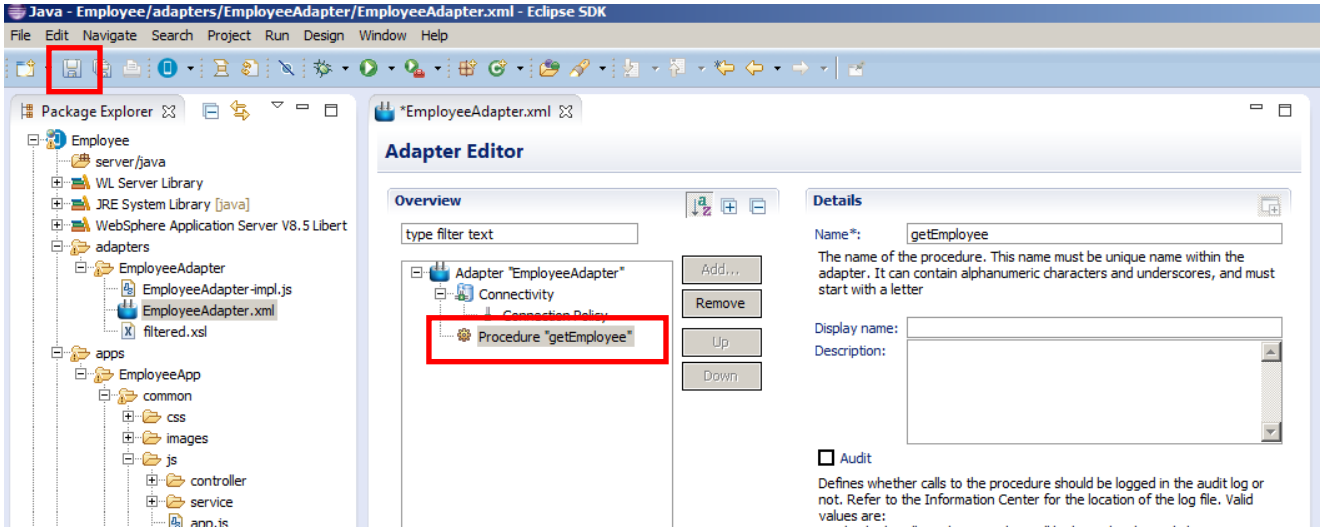
7. Right-click ‘Procedure “getStoriesFiltered”’ and then ‘Remove’. This was a procedure generated by default, which is not needed.



8. The operation that we use in our REST API Service on IIB is ‘getEmployee’. This has to be specified as a procedure name in the Adapter.

   Click ‘Procedure “getStories”’ and in the Name field change it to ‘getEmployee’.

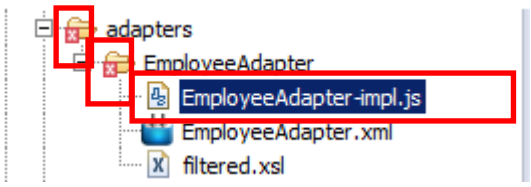
   Save the adapter with ‘Ctrl-S’ or the ‘Save’ button in the MobileFirst Studio:



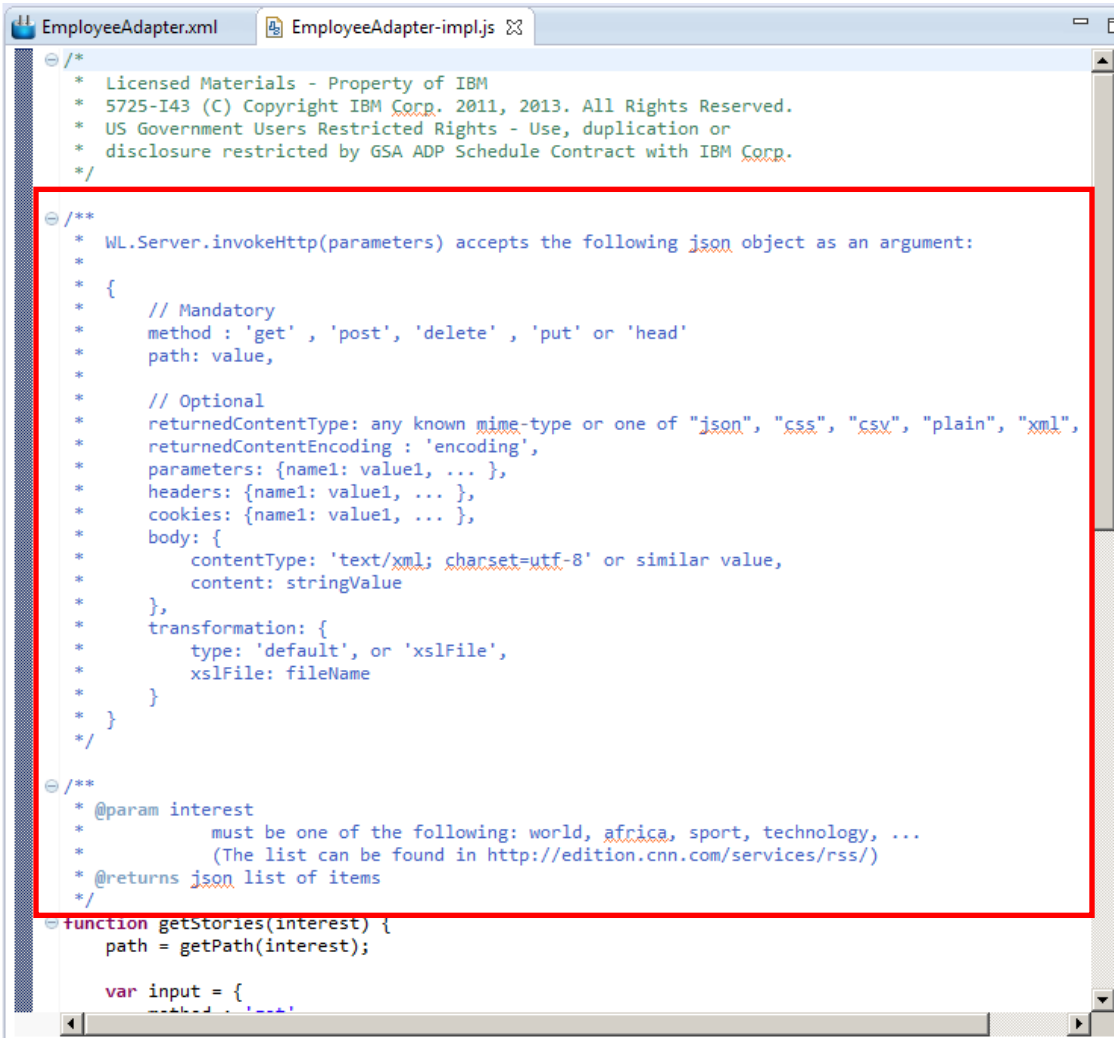
3.6 Adapter Implementation File

1. You will notice in the Package Explorer that there is an error in the 'EmployeeAdapter' section (red cross). This is because we have not made any changes in the adapter implementation JavaScript file to include the procedure 'getEmployee'.

   Double-click 'EmployeeAdapter-impl.js' file



2. In the editor, you will notice a large 'comment' section at the beginning describing how to edit the JavaScript code to access your remote applications. Following this are the functions that include the connection details (currently the ones generated by default when the adapter was created).



3. Scroll all the way to the bottom of the file (you can expand the view of the file to full screen by double clicking on its name tab).
   You will now edit the file to define the connection to the IIB REST API Service.

   First, delete the functions '**function** getStoriesFiltered(interest)' and '**function** getPath(interest)':

EmployeeAdapter.xml | EmployeeAdapter-impl.js

```
function getStories(interest) {
    path = getPath(interest);

    var input = {
        method : 'get',
        returnedContentType : 'xml',
        path : path
    };


    return WL.Server.invokeHttp(input);
}
/**
 *
 * @param interest
 *            must be one of the following: world, africa, sport, technology, ...
 *            (The list can be found in http://edition.cnn.com/services/rss/)
 * @returns json list of items
 */
function getStoriesFiltered(interest) {
    path = getPath(interest);

    var input = {
        method : 'get',
        returnedContentType : 'xml',
        path : path,
        transformation : {
            type : 'xslFile',
            xslFile : 'filtered.xsl'
        }
    };

    return WL.Server.invokeHttp(input);
}



function getPath(interest) {
    if (interest == undefined || interest == '') {
        interest = '';
    }else {
        interest = '_' + interest;
    }
    return 'rss/edition' + interest + '.rss';
}
```

Delete this part

4. On the last remaining '**function** getStories(interest)' you need to do the following changes:

To ensure that changes are made correctly, we have provided this code for you.

Copy/paste the contents of **C:\student10\mobile\MobileFirst\EmployeeAdapter_REST.txt,** replacing the function **getStories(interest)**.

`function getStories(interest) {`	`function getEmployee(empNumber) {`
`path = getPath(interest);`	`path = "TestWebApp/resources/employees/";`

`var input = {`
`method : 'get',`
`returnedContentType : 'xml',`
`path : path`
	`var input = {`
	`method : 'get',`
	`returnedContentType : 'json',`
	`path : path + empNumber,`
	`body: {`
	`contentType: 'application/json; charset=UTF-8'`
	`}`

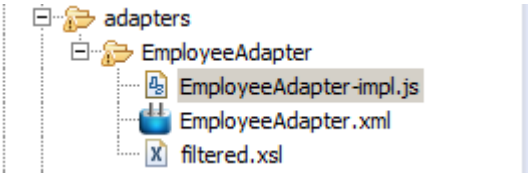
`};`	`};`

5. Once you have completed the changes your 'EmployeeAdapter-impl.js' file (excluding the 'comment' part) should look like below.

```
 */
function getEmployee(empNumber) {

    path = "TestWebApp/resources/employees/";

    var input = {
        method : 'get',
        returnedContentType : 'json',
        path : path + empNumber,
        body: {
            contentType: 'application/json; charset=UTF-8'
        }
    };


    return WL.Server.invokeHttp(input);
}
```

6. Save the changes with 'Ctrl-S' and close the file.

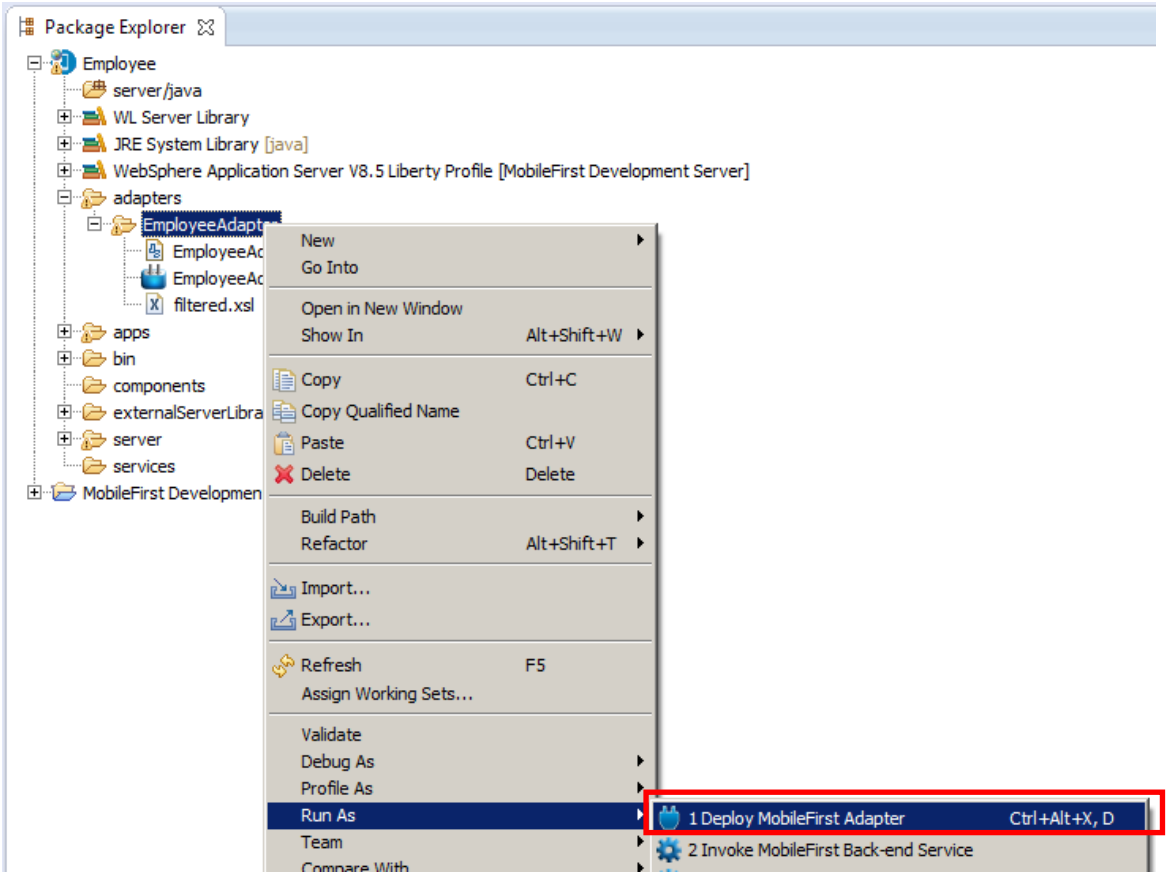
7. You will see that now the error in the 'adapters' section has disappeared:



4. Test the Mobile Application

4.1 View the MobileFirst Console

1. In Package Explorer, right-click the EmployeeAdapter and select 'Run As' --> 'Deploy MobileFirst Adapter.

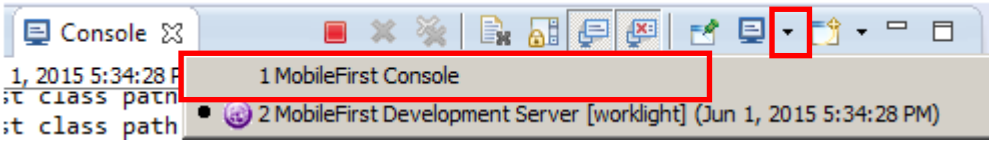


Please note that this may take a minute or two

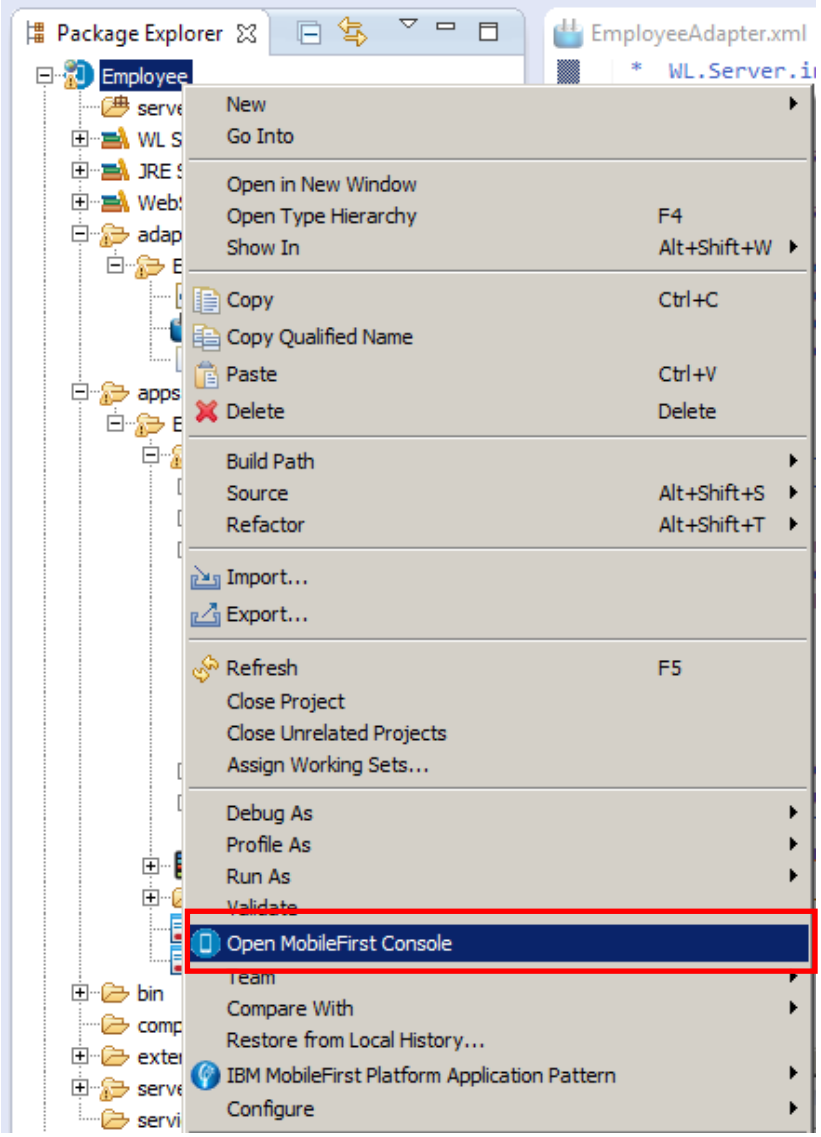
2. Once the adapter deployment has completed, you will see the confirmation in the console.

```
[2015-05-15 14:25:38]          Server port: 10080
[2015-05-15 14:25:41]          Adapter build and deploy finished.
```

Please note that the you may be shown the Mobile First Server console. To switch to Mobile First console, click on the arrow next to the little 'screen' and make a selection as shown below



3. Right-click the Employee project and from the menu click on ‘Open MobileFirst Console’.



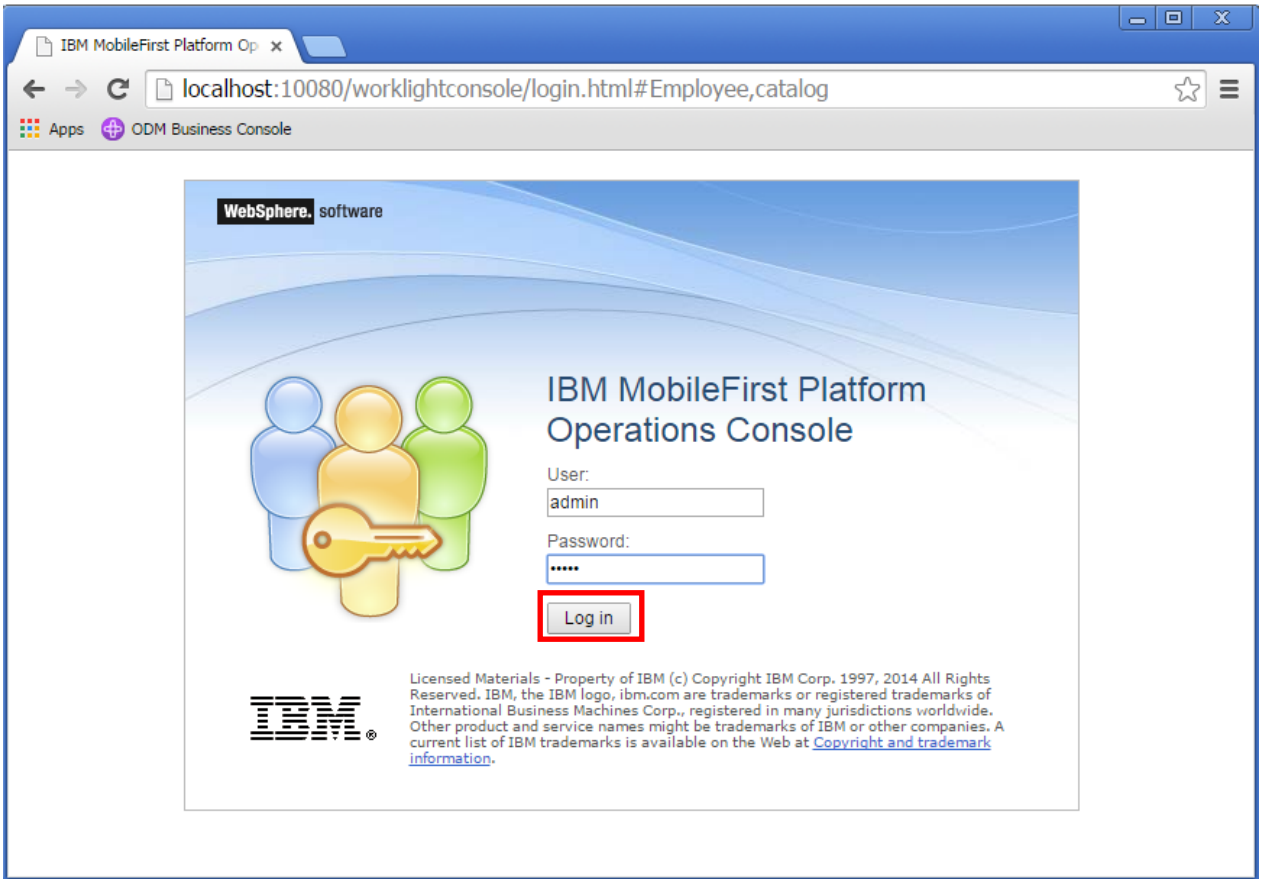
This will initiate the start of the server, compile the Employee project, and will deploy the EmployeeApp to the server.

This will take a few minutes. Do not be concerned if you see a few warnings or error messages in the Console.

4. Once deployment has completed, a Browser will open automatically with the log in page for IBM MobileFirst console.

   Please note that the Firefox Browser (the default Browser) will not be able to render the mobile pages view, so it is recommended that you use Chrome (copy the URL and open in Chrome).

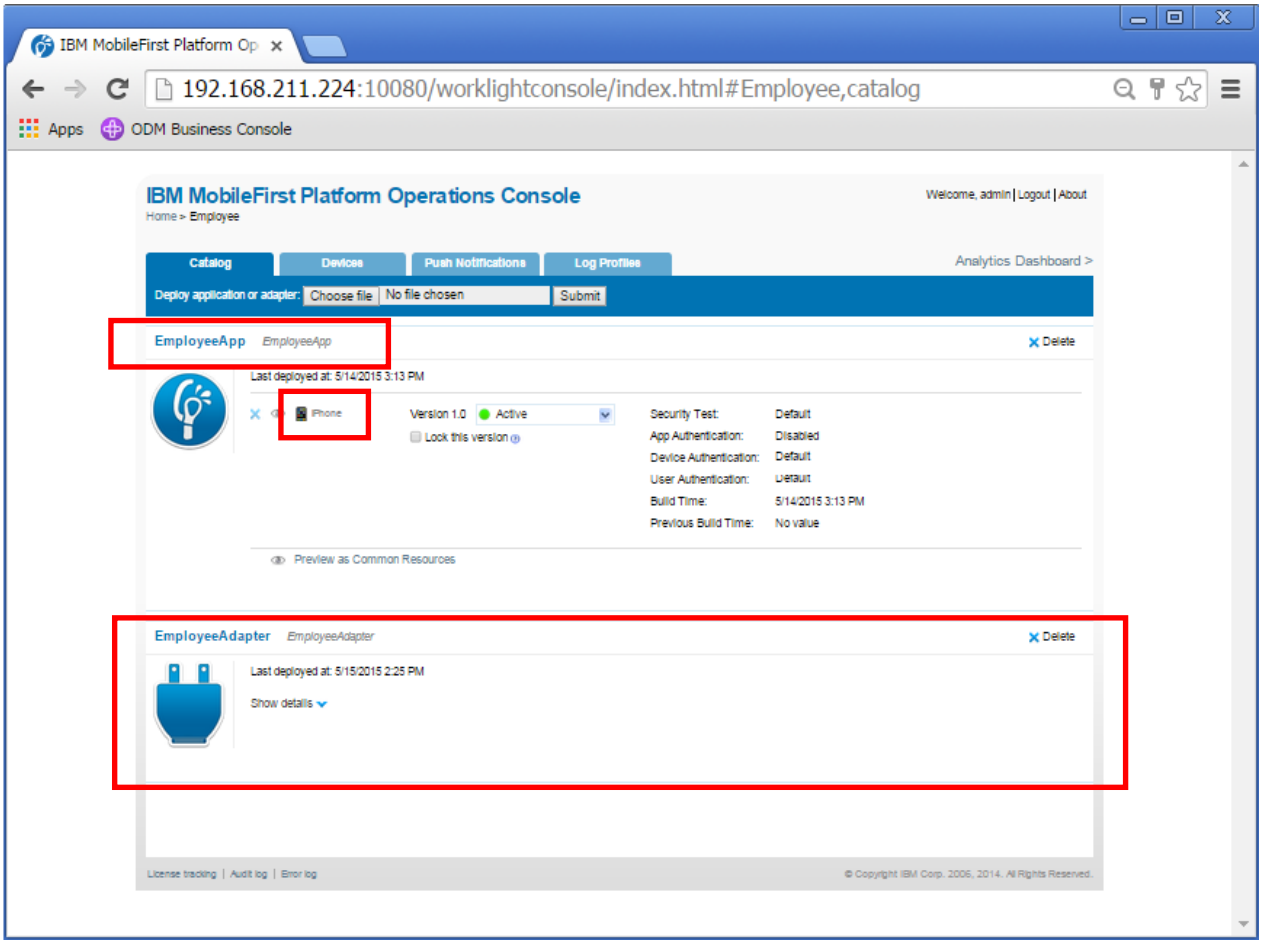
   Enter admin/admin as User/ Password.



5. In the console you will see the deployed 'EmployeeApp'. You are able to view the properties of the app and preview in a Mobile Browser Simulator.

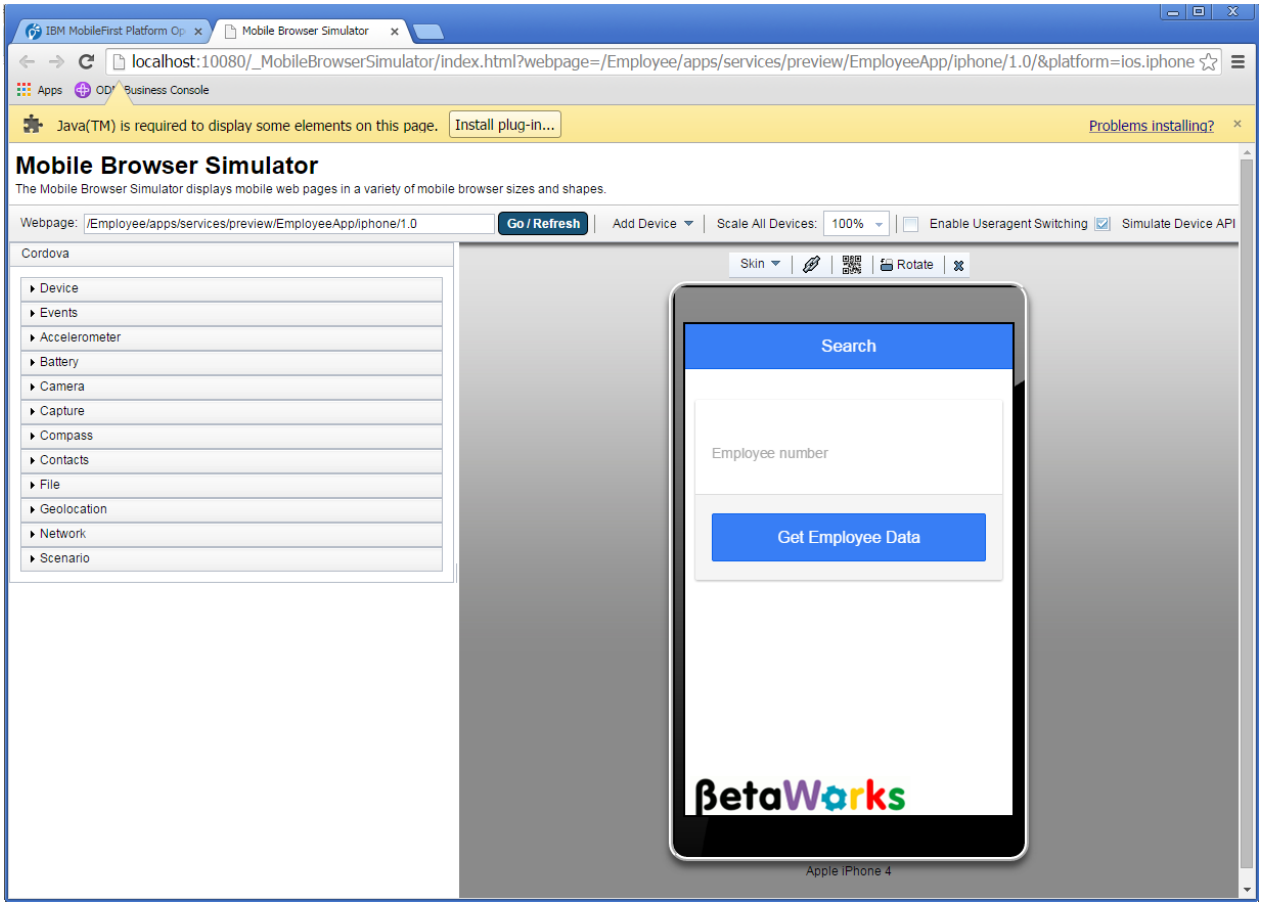
   In the bottom half of the view you will see the adapter that you created for connecting the Mobile application to IBM Integration Bus

   Click the 'iPhone' icon as shown below:



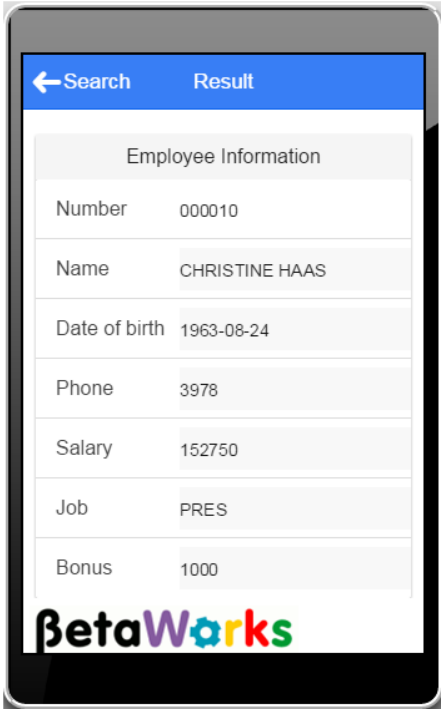
6. The Mobile Browser simulator is opened in a new tab with a preview of the Mobile application on iPhone4.

   Ignore the message for 'unsupported java plug-in-.

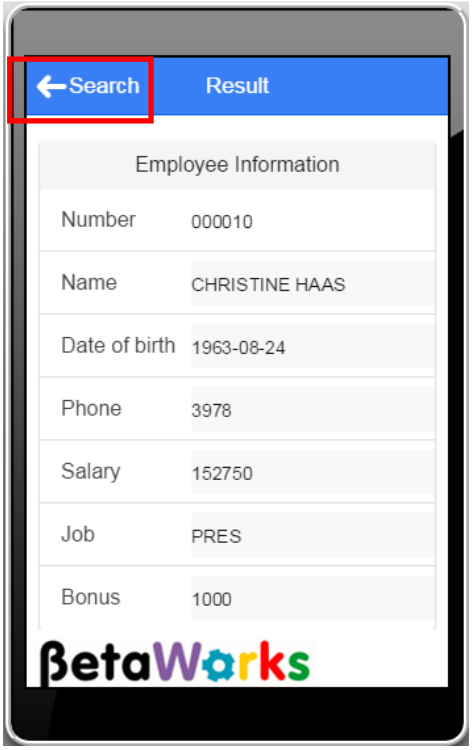


7. You are now presented with the first screen of the Mobile App and you will need to enter an 'Employee Number' in order to receive the corresponding data from the database, accessed from the REST API Service running on IIB.
   Enter 000010 as an Employee Number in the field and press (click) 'Get Employee Data':

8. The employee data has been returned from the database (probably already familiar to you):



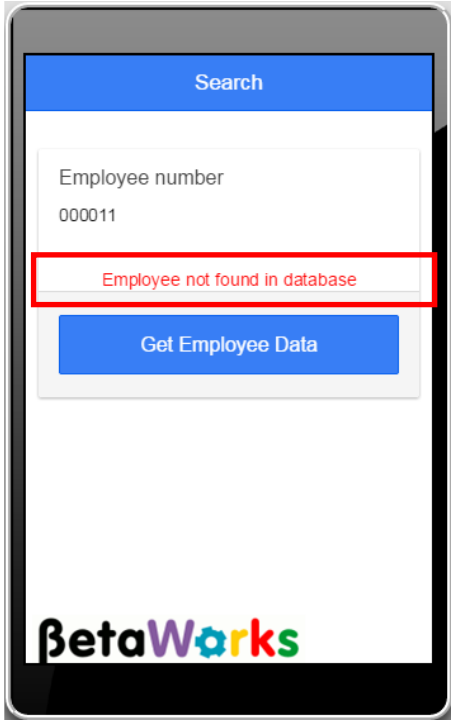
9. In the Mobile Browser Simulator, click the 'Search' button to return to the mobile view to search for another employee:



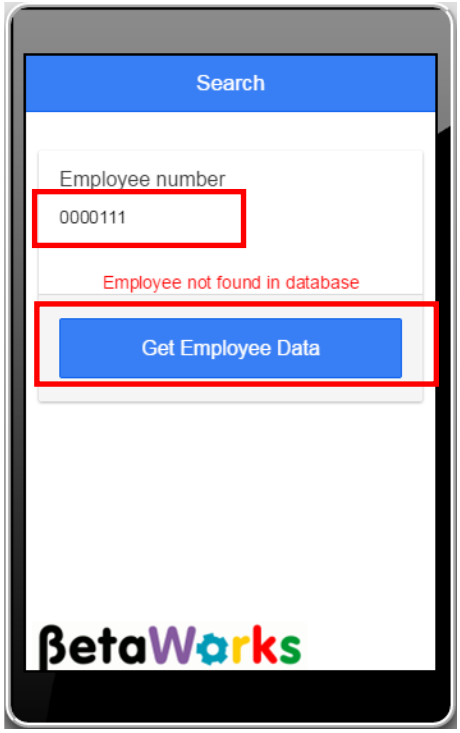
4.2 Error handling notifications

1. Enter '000011' as an Employee Number and press 'Get Employee Data' button:

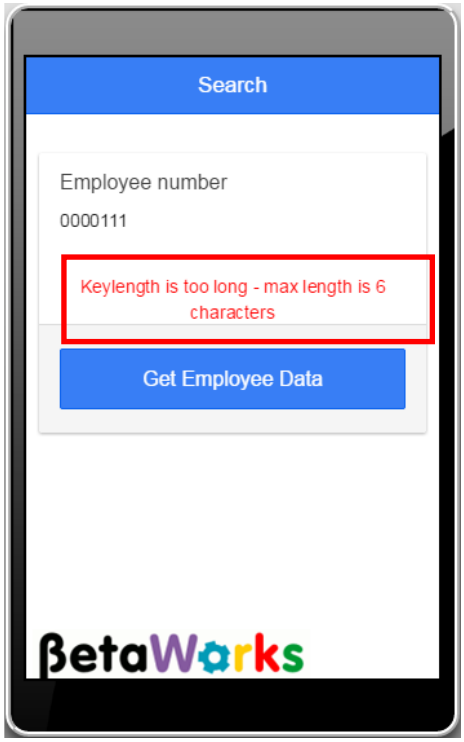
2. You will see the message returned from IIB 'Employee not found in database' in red colour font:



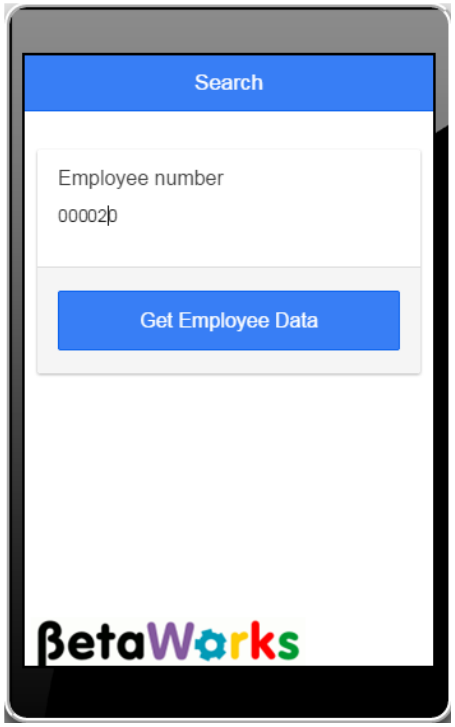
3. Enter a new seven digit number to search in the database – '0000111' and press 'Get Employee data':



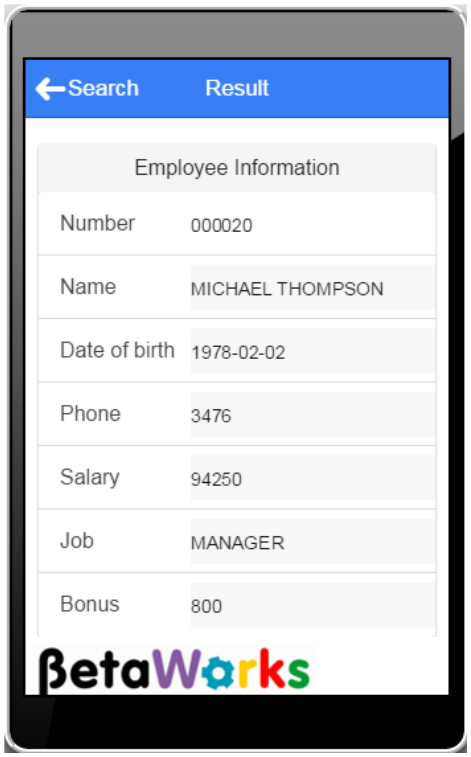
4. This time, you will see a different message, advising you that the input number should be 6 characters in length:



5. Feel free to use another test, by entering Employee number 000020 or 000050:



6. You will see again the returned data in the mobile application 'Result' page:



5. Appendix

To create the RESTNODE and configure it, we have provided a script. In an IIB Command Console, change directory to c:\student10\REST_service\install.

Run the script file:

```
Create_RESTNODE
```

Accept the default values for the IIB node name (RESTNODE).

This script will create the new node, and run two key commands:

- Enable HTTP embedded listeners. The REST support is only provided for embedded listeners, not the node-wide listener:

```
mqsichangeproperties RESTNODE
      -e default
      -o ExecutionGroup
      -n httpNodesUseEmbeddedListener -v true
```

- Enable Cross-Origin Resource Scripting for REST. This is required when testing with the SwaggerUI test tool. See http://en.wikipedia.org/wiki/Cross-origin_resource_sharing for further information.

```
mqsichangeproperties RESTNODE -e default
      -o HTTPConnector
      -n corsEnabled -v true
```

The script will also configure the JDBC parameters for connection to the SAMPLE database, which will be used in this scenario.

END OF LAB GUIDE

.