IBM Integration Bus

Developing a REST API

Featuring:

The REST API tools for IIB
Testing with Swagger UI
1. INTRODUCTION AND PREPARATION........................................................................................................................... 3
  1.1 INTRODUCTION....................................................................................................................................................... 3
  1.2 OPEN THE WINDOWS LOG MONITOR FOR IIB........................................................................................................ 3
  1.3 CONFIGURE TESTNODE_IIBUSER FOR REST APIs............................................................................................ 5
  1.4 CONFIGURE INTEGRATION BUS NODE TO WORK WITH DB2.................................................................................. 6

2. CREATE THE REST API.................................................................................................................................................... 7
  2.1 EXAMINE THE EMPLOYEESERVICE JSON DOCUMENT......................................................................................... 7
  2.2 IMPORT THE SUBMAP AND MAIN MAP SHARED LIBRARY.................................................................................... 8
  2.3 CREATE THE NEW REST API..................................................................................................................................... 9
  2.4 IMPLEMENT THE MAIN MAP..................................................................................................................................... 14
  2.5 IMPLEMENT THE GETEMPLOYEE OPERATION..................................................................................................... 23

3. TEST THE EMPLOYEESERVICE REST API............................................................................................................. 26
  3.1 DEPLOY THE SERVICE.................................................................................................................................................. 26
  3.2 TEST THE SERVICE...................................................................................................................................................... 27

END OF LAB GUIDE ...................................................................................................................................................... 31
1. Introduction and Preparation

1.1 Introduction
In this lab you will create a new REST API, based on a provided Swagger document. The document is based on the EMPLOYEE scenario, and describes a number of REST operations. This guide will implement the Get Employee operation.

1.2 Open the Windows Log Monitor for IIB

A useful tool for IIB development on Windows is the IIB Log Viewer. This tool continuously monitors the Windows Event Log, and all messages from the log are displayed immediately.

From the Start menu, click IIB Event Log Monitor. The Monitor will open; it is useful to have this always open in the background.

This tool is not shipped as part of the IIB product; please contact us directly if you would like a copy.
**Important note**

This lab, version 10.0.0.3, has been updated significantly from earlier versions. The following changes have been made:

You should use the Windows user "iibuser". This user is a member of mqbrkrs and mqm, but is not a member of Administrators. The user "iibuser" can create new IIB nodes and do all required IIB development work. However, installation of the IIB product requires Administrator privileges (not required in this lab).

The database has been changed from the DB2 SAMPLE database to the DB2 HRDB database. HRDB contains two tables, EMPLOYEE and DEPARTMENT. These tables have been populated with data required for this lab. (The DDL for the HRDB is available in the student10 folder; we intend to provide corresponding DDL for Microsoft SQL/Server and Oracle over time).

The map node now retrieves multiple rows from the database, using an SQL "LIKE" function. Additionally, the map has been refactored to use a main map and a submap. Both the main map and submap are located in a shared library.

Input to the integration service and the REST service is now a simple schema containing just one element, the required employee number.

As a consequence, this version of the lab, and the associated solution, can only be used with the corresponding changes in other labs. Use version 10.0.0.3 of all labs in this series of lab guides.
1.3 Configure TESTNODE_iibuser for REST APIs

Login to Windows as the user "iibuser", password = "passw0rd". (You may already be logged in).

Start the IIB Toolkit from the Start menu.

The IIB support for the REST API requires some special configuration for the IIB node and server.

1. Ensure that TESTNODE_iibuser is started.

Enable Cross-Origin Resource Scripting for REST. This is required when testing with the SwaggerUI test tool. See http://www.w3.org/TR/cors/?cm_mc_uid=09173639950214518562833&cm_mc_sid_502000_00=1452177651 for further information.

(Helpful hint - the VM keyboard is set to UK English. If you cannot find the "\" with your keyboard, use "cd .." to move the a higher-level folder in a DOS window), or change the keyboard settings to reflect your locale.)

In an IIB Command Console (shortcut on the Start menu), run the command:

```bash
mqsichangeproperties TESTNODE_iibuser
   -e default
   -o HTTPConnector
   -n corsEnabled -v true
```
1.4 Configure Integration Bus node to work with DB2

If you have already done Lab 1 in this series (create an Integration Service), you can skip straight to Create the REST Service on the next page.

To run this lab, the Integration Bus node must be enabled to allow a JDBC connection to the HRDB database.

1. Open an IIB Command Console (from the Start menu), and navigate to
c:`\student10\Create_HR_database`

2. Run the command
   3_Create_JDBC_for_HRDB

   Accept the defaults presented in the script. This will create the required JDBC configurable service for the HRDB database.

3. Run the command
   4_Create_HRDB_SecurityID

4. Stop and restart the node to enable the above definitions to be activated

   mqsistop TESTNODE_iibuser

   mqsistart TESTNODE_iibuser

This will create the necessary security credentials enabling TESTNODE_iibuser to connect to the database.

Recreating the HRDB database and tables
The HRDB database, and the EMPLOYEE and DEPARTMENT tables have already been created on the supplied VMWare image. If you wish to recreate your own instance of this database, the command 1_Create_HRDB_database.cmd and
2_Create_HRDB_Tables.cmd are provided for this. If used in conjunction with the VM image, these commands must be run under the user "iibadmin". Appropriate database permissions are included in the scripts to GRANT access to the user iibuser.
2. Create the REST API

In this section you will create a new REST API. This scenario will be based on the EmployeeService example that you may have used in other labs in this series.

2.1 Examine the EmployeeService JSON document

1. In Windows Explorer, locate the file
   
   c:\student10\REST_service\resources\EmployeeService.json

   Open the file with the Notepad++ editor (right-click, select Edit with Notepad++).

   We have installed a JSON document plugin into Notepad++, so this JSON document will be formatted for easy reading.

   The JSON document has been constructed to define interfaces for the EMPLOYEE and DEPARTMENT tables.

   The main section of the document is a series of operations (GET, POST, PUT, etc), associated with different types of operation (getEmployee, listEmployees, etc).

   At the very bottom of the document are two definitions for Employee and Department, which defines the precise structure of these elements.

   This document will be used as the basis of the REST applications that you will build in IIB.

   Look for the variable "employeeNumber", which is referenced in several places in the document. This variable will be used by the IIB REST application.

   Close the editor without making any changes to the document.
2.2 Import the submap and main map Shared Library

1. The REST API that you will develop will use the EMPLOYEE tables from the HRDB database, and the submap that was developed in the Integration Service lab. However, you will use a pre-built version of these artefacts, to ensure the REST API is developed successfully.

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

   c:\users\iibuser\IBM\IIB 10\workspace_REST

2. In the new workspace, import the Project Interchange file:

   C:\student10\integration_service\solution\EmployeeService_10.0.0.3.zip

Import all the projects from this PI file:
- HRDB
- EmployeeService (will not be used in this lab)
- EmployeeService_interface_and_maps

3. When imported, you will see the projects as shown here. Note the HRDB project is an included project inside the EmployeeService_interface_and_maps library.
2.3 Create the new REST API

1. In the workspace, create a new REST API.

![New Artifact dialog]

2. Name the new service EmployeeService_REST, and click Next.

![Create a REST API dialog]
3. Using the Browse button, import the JSON document

    c:\student10\REST_service\resources\EmployeeService.json

Click Next.
4. The summary window will show you all of the REST operations that were defined in the JSON document. These operations were constructed to match the EMPLOYEE and DEPARTMENT tables in the HRDB database.

Note there is an operation named getEmployees (i.e. retrieve a list of all employees), and an operation named getEmployee. This lab will implement the `getEmployee` operation.

Click Finish.
5. The operations have now been imported into the IIB Toolkit. The import process has also created a base REST application and a message flow that implements the REST API.

Expand each operation. You will see that you can implement the operation by clicking the "Implement the operation" link.
6. Before proceeding with the implementation, the REST API has to reference the required Shared Library. This is because the main map (that you will build shortly) will be stored in the library.

In the navigator, right-click the EmployeeService_REST API and select Manage Library references.

Tick EmployeeService_interface_and_maps and click OK.
2.4 Implement the main map

1. A new map is required because for REST APIs, the input JSON parameter is placed in the IIB Local Environment. This new map is required to extract this parameter, and pass it to the existing submap.

The main map for the REST API getEmployee operation will be located in the EmployeeService_interface_and_maps shared library.

Highlight this library, and click New, Message Map.

![Application Development](image)

2. Select the shared library container, and name the new map getEmployee_REST.

Click Next.

![New Message Map](image)
3. For the map inputs and outputs, make the following selections:
   - Input
     - IBM supplied message models - JSON object
   - Output
     - IBM supplied message models - JSON object
   
   Click Finish.

4. The basic mappings will be shown.
5. For a REST GET operation, the input parameter (employeeNumber in this case) will be available in the LocalEnvironment. For the map to access the Local Environment, you must explicitly add this header to the Message Assembly.

On the input Message Assembly, right-click and select "Add or remove headers and folders".

Select the LocalEnvironment and click OK.
6. Expand the Local Environment and the REST section (located near the bottom of the Local Environment).

Incoming REST parameters will appear under the REST/Input/Parameters element, so the definition of this element needs to be added here.

Right-click the "any" element and select "Cast".

7. Recheck the above step to make sure that you have put this new element under the REST/Input/Parameters folder of LocalEnvironment, and NOT directly under the LocalEnvironment folder at the bottom of the list.
8. From the Type Selection popup, select employeeNumber, and click OK.

Note that the target namespace for employeeNumber is `<no target namespace>`. It is necessary to use an element with no namespace, because the REST input parameters are included in the REST folder without namespace.

When complete, the element should look like this:
9. For the output assembly, the JSON message needs to be Cast as EmployeeResponse.

   Fully expand the output assembly, and in the final "any" element, right-click and select Cast.
10. In the Type Selection (with an "e" filter), select EmployeeResponse, and click OK.
11. The new element employeeNumber now needs to be mapped to the output JSON/EmployeeResponse assembly, using a submap.

Unfortunately, expanding the input Local Environment has probably meant that the output message has disappeared from the map display. To handle this, right-click employeeNumber, and select “Quick Link to Output”.

12. In the pop-up window, scroll down to the bottom and select the JSON/Data/EmployeeResponse element.

The generated "For each" transform will be connected to the required output assembly.
13. Now change the "For Each" to a submap.

Click the blue drop-down arrow, and select "submap" from the list.

The transform will be changed to a submap, but note that an error is showing. This is because you haven't yet provided the name of the submap.

14. Set the Properties of the submap to reference the required submap.

Highlight the submap node, and in the Properties of the submap file, click Browse.

Select `getEmployees_submap`, and click OK.

(Note - if you uncheck "Show only applicable maps", you will see additional maps and submaps. These are not shown initially, mainly because the inputs and outputs for these maps would not match the requirements of the current map transform).

The main map is now complete, so save (Ctrl-S) and close.
2.5 Implement the getEmployee operation

1. You will now implement the getEmployee operation of the REST API.

   Return to the EmployeeService_REST API. In the Operations list, locate getEmployee and click “Implement the operation”.

   This will open the subflow editor. Each operation is implemented with a separate subflow.

---

**NOTE** - Select the getEmployee operation in the /employees/{employeeNumber} section.

Do not confuse this with the /getEmployees operation in the /employees section, just above.

---

![Resources Diagram]

- GET getEmployee: Retrieve the details for one or more matching employees
- PUT updateEmployee: Updates an existing employee
- DELETE deleteEmployee: Deletes an existing employee
2. Drop a Mapping node onto the flow editor.

Name the new map "getEmployee_REST".
3. This mapping node will actually be the getEmployee_REST map that you just created, so you must change the properties of the node to reference this map.

Highlight the map node, and select the Properties of the map node. Click Browse to select the getEmployee_REST map from the shared library.

Click OK.

4. Connect the nodes as shown. Save and close the subflow.
3. Test the EmployeeService REST API

This chapter will show you how to use the SwaggerUI tool to send a REST request into the REST API that you have just created.

3.1 Deploy the service

1. In the navigator, deploy the following resources, in order, to the default server.
   1. EmployeeService_interface_and_maps
   2. EmployeeService_REST
3.2 Test the service

1. Open the IIB web UI by right-clicking TESTNODE_iibuser and selecting Start Web User Interface.

2. You will be switched to the default browser. Fully expand TESTNODE_iibuser, down to the EmployeeService_REST, as shown below.

Under EmployeeService_REST, click “API”, which will show you the available operations in the REST API, and whether they have been implemented. Check that you have implemented the correct operation.

It will also show you the URLs for local and remote invocations, and the REST API definitions (the .json file).
3. On the “Remote URL for REST API definitions”, right-click and select "Copy Link Location".

4. In Firefox, open a new tab, and open the SwaggerUI tool (using the bookmark in the REST folder).

   By default, this will open the Petstore Swagger document.

5. In the entry field (not the browser address field), paste the contents of the clipboard and click Explore.

   The two high-level functions, departments and employees, will be shown.
6. We are concerned with the getEmployee operation so click "List Operations" to show the operations related to employees.

Note that SwaggerUI does not have any knowledge at this point of whether the operation has been implemented.

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

The employeeNumber will show the expression (required). Replace this with a suitable value, say 000010.
8. When you have provided an employeeNumber, click Try it out!

If successful, the returned data will look something like this. Note the database response information (user return code, number of rows returned), as well as the user data.

```
{
    "EmployeeResponse": {
        "DBResp": {
            "UserReturnCode": 0,
            "RowsRetrieved": 1
        },
        "EMPLOYEE": {
            "EMPNO": "0000010",
            "FIRSTNAME": "CHRISTINE",
            "MIDDLEINIT": "I",
            "LASTNAME": "HAAS",
            "WORKDEPT": "A00",
            "PHONENO": "3978",
            "HIREDATE": "1995-01-01",
            "JOB": "PRES",
            "EDLEVEL": 18,
            "SEX": "F",
            "BIRTHDATE": "1963-08-24",
            "SALARY": 152750,
        }
    }
}
```

Response Code

200

Response Headers

```
{
    "content-type": "application/json; charset=utf-8"
}
```
9. Provide an employeeNumber that does not exist, for example 000012 (but make sure you use an employeeNumber that has 6 characters).

You will see the service has worked (UserReturnCode = 0), but no data has been found (RowsRetrieved = 0).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Description</th>
<th>Parameter Type</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>employeeNumber</td>
<td>000012</td>
<td></td>
<td>path</td>
<td>string</td>
</tr>
</tbody>
</table>

**Response Messages**

<table>
<thead>
<tr>
<th>HTTP Status Code</th>
<th>Reason</th>
<th>Response Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>404</td>
<td>The employee cannot be found</td>
<td></td>
</tr>
<tr>
<td>500</td>
<td>Something wrong in Server</td>
<td></td>
</tr>
</tbody>
</table>

**Request URL**

http://172.17.2.133:7800/TestWebApp/resources/employees/000012

**Response Body**

```json
c {
  "EmployeeResponse": {
    "DBResp": {
      "UserReturnCode": 0,
      "RowsRetrieved": 0,
      "RowsAdded": 0,
      "RowsUpdated": 0,
      "RowsDeleted": 0,
      "SQLCode_ErrorCode": 0,
      "SQLState_SQLState": null,
      "SQL_Failed Message": null
    }
  }
}
```