BetaWorks

IBM Integration Bus

Developing a REST API

Featuring:

The REST API tools for IIB
Testing with Swagger UI

January 2016
Hands-on lab built at product
Version 10.0.0.3

1. Introduction and Preparation

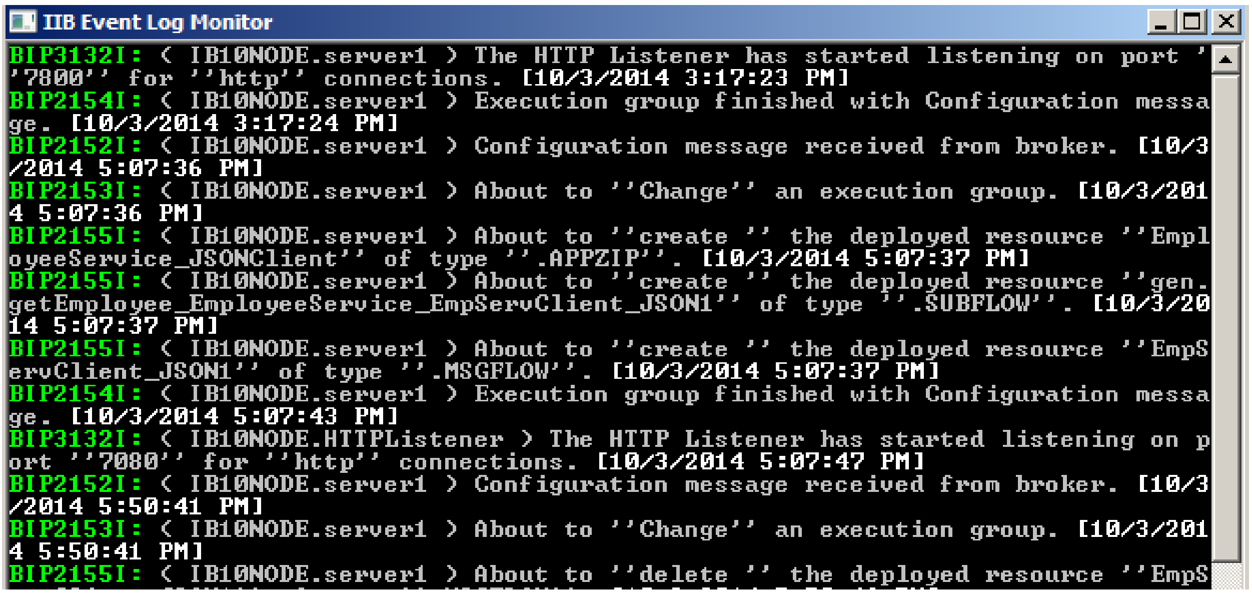
1.1 Introduction

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.



```
BIP3132I: ( IB10NODE.server1 ) The HTTP Listener has started listening on port '
'7800'' for ''http'' connections. [10/3/2014 3:17:23 PM]
BIP2154I: ( IB10NODE.server1 ) Execution group finished with Configuration messa
ge. [10/3/2014 3:17:24 PM]
BIP2152I: ( IB10NODE.server1 ) Configuration message received from broker. [10/3
/2014 5:07:36 PM]
BIP2153I: ( IB10NODE.server1 ) About to ''Change'' an execution group. [10/3/201
4 5:07:36 PM]
BIP2155I: ( IB10NODE.server1 ) About to ''create '' the deployed resource ''Empl
oyeeService_JSONClient'' of type ''.APPZIP''. [10/3/2014 5:07:37 PM]
BIP2155I: ( IB10NODE.server1 ) About to ''create '' the deployed resource ''gen.
getEmployee_EmployeeService_EmpServClient_JSON1'' of type ''.SUBFLOW''. [10/3/20
14 5:07:37 PM]
BIP2155I: ( IB10NODE.server1 ) About to ''create '' the deployed resource ''EmpS
ervClient_JSON1'' of type ''.MSGFLOW''. [10/3/2014 5:07:37 PM]
BIP2154I: ( IB10NODE.server1 ) Execution group finished with Configuration messa
ge. [10/3/2014 5:07:43 PM]
BIP3132I: ( IB10NODE.HTTPListener ) The HTTP Listener has started listening on p
ort ''7080'' for ''http'' connections. [10/3/2014 5:07:47 PM]
BIP2152I: ( IB10NODE.server1 ) Configuration message received from broker. [10/3
/2014 5:50:41 PM]
BIP2153I: ( IB10NODE.server1 ) About to ''Change'' an execution group. [10/3/201
4 5:50:41 PM]
BIP2155I: ( IB10NODE.server1 ) About to ''delete '' the deployed resource ''EmpS
```

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_50200000=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:

```
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.

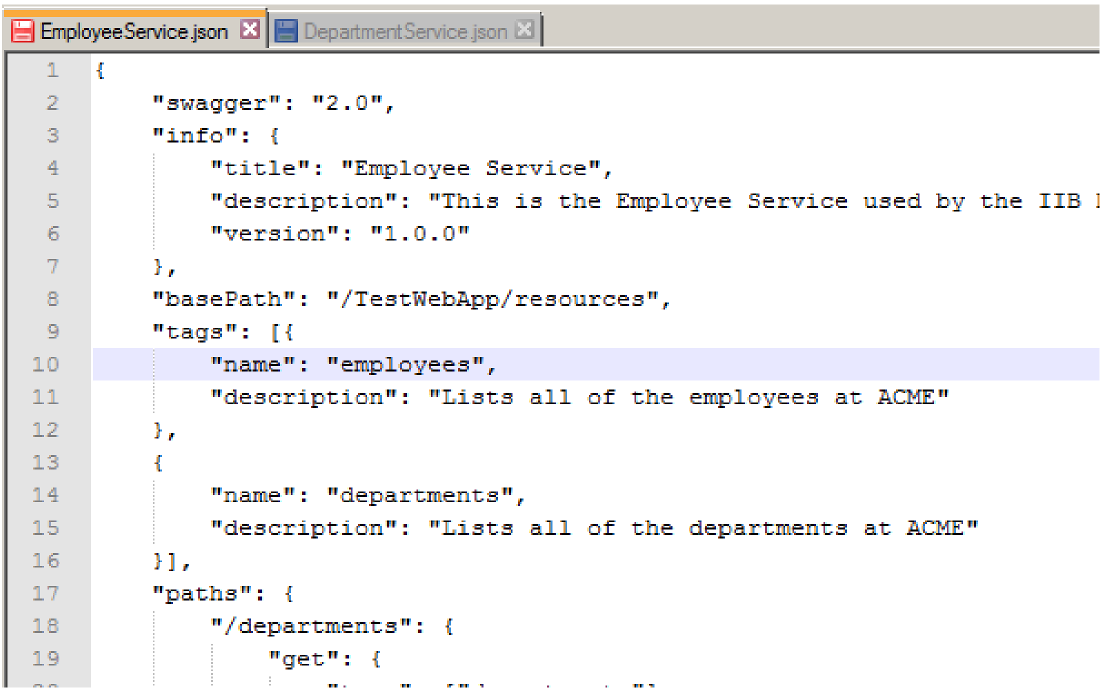
   The main section of the document is a series of operations (GET, POST, PUT, etc), associated with different types of operation (getEmployee, listEmployees, etc).

   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.



```
{
    "swagger": "2.0",
    "info": {
        "title": "Employee Service",
        "description": "This is the Employee Service used by the IIB
        "version": "1.0.0"
    },
    "basePath": "/TestWebApp/resources",
    "tags": [{
        "name": "employees",
        "description": "Lists all of the employees at ACME"
    },
    {
        "name": "departments",
        "description": "Lists all of the departments at ACME"
    }],
    "paths": {
        "/departments": {
            "get": {
```

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

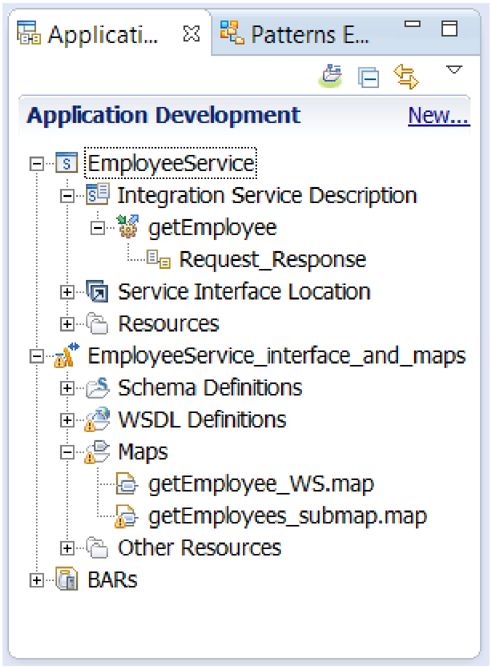
   ```
   c:\users\iibuser\IBM\IIB 10\workspace_REST
   ```

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

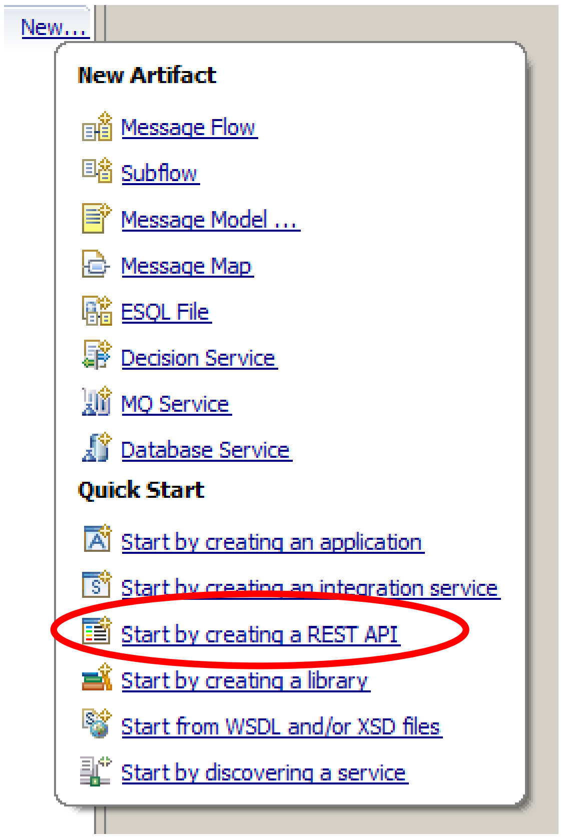
3. When imported, you will see the projects as shown here. Note the HRDB project is an included project inside the EmployeeService_interface_and_maps library.



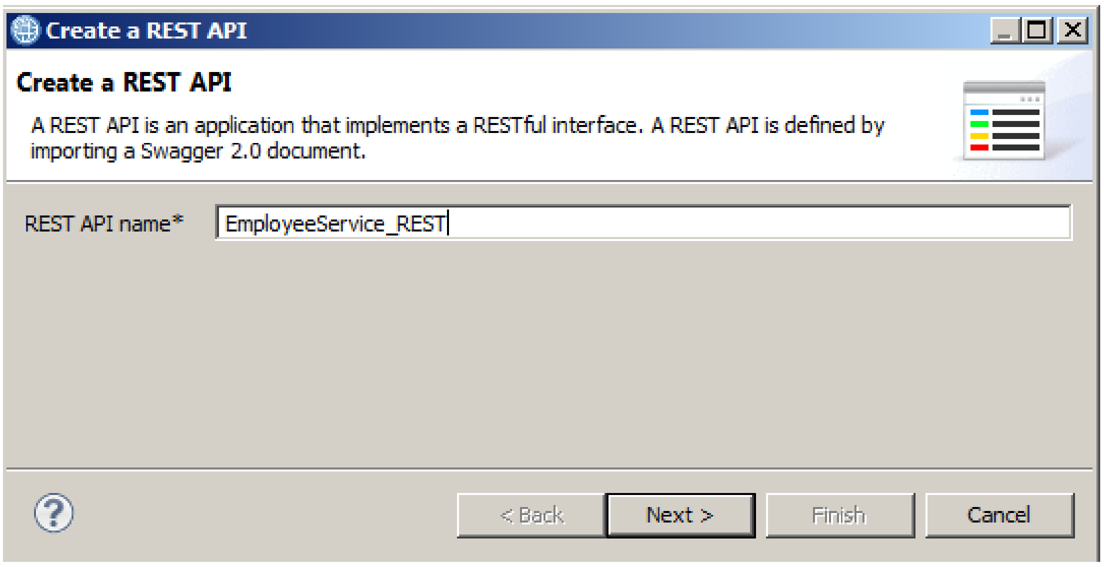
- EmployeeService
  - Integration Service Description
    - getEmployee
      - Request_Response
  - Service Interface Location
  - Resources
- EmployeeService_interface_and_maps
  - Schema Definitions
  - WSDL Definitions
  - Maps
    - getEmployee_WS.map
    - getEmployees_submap.map
  - Other Resources
- BARs

2.3 Create the new REST API

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


New…
New Artifact
Message Flow
Subflow
Message Model …
Message Map
ESQL File
Decision Service
MQ Service
Database Service
Quick Start
Start by creating an application
Start by creating an integration service
Start by creating a REST API
Start by creating a library
Start from WSDL and/or XSD files
Start by discovering a service

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


Create a REST API

Create a REST API

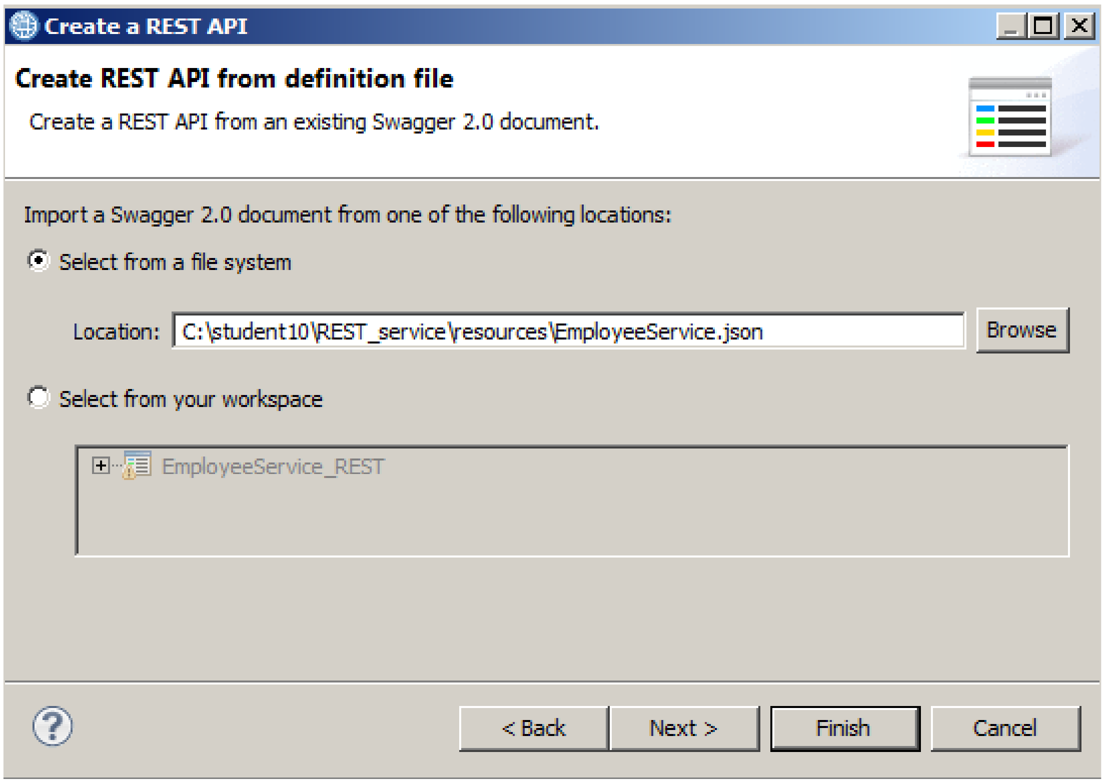
A REST API is an application that implements a RESTful interface. A REST API is defined by importing a Swagger 2.0 document.

REST API name* EmployeeService_REST

3. Using the Browse button, import the JSON document

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

   Click Next.



Create a REST API

Create REST API from definition file

Create a REST API from an existing Swagger 2.0 document.

Import a Swagger 2.0 document from one of the following locations:

(●) Select from a file system

Location: C:\student10\REST_service\resources\EmployeeService.json

( ) Select from your workspace

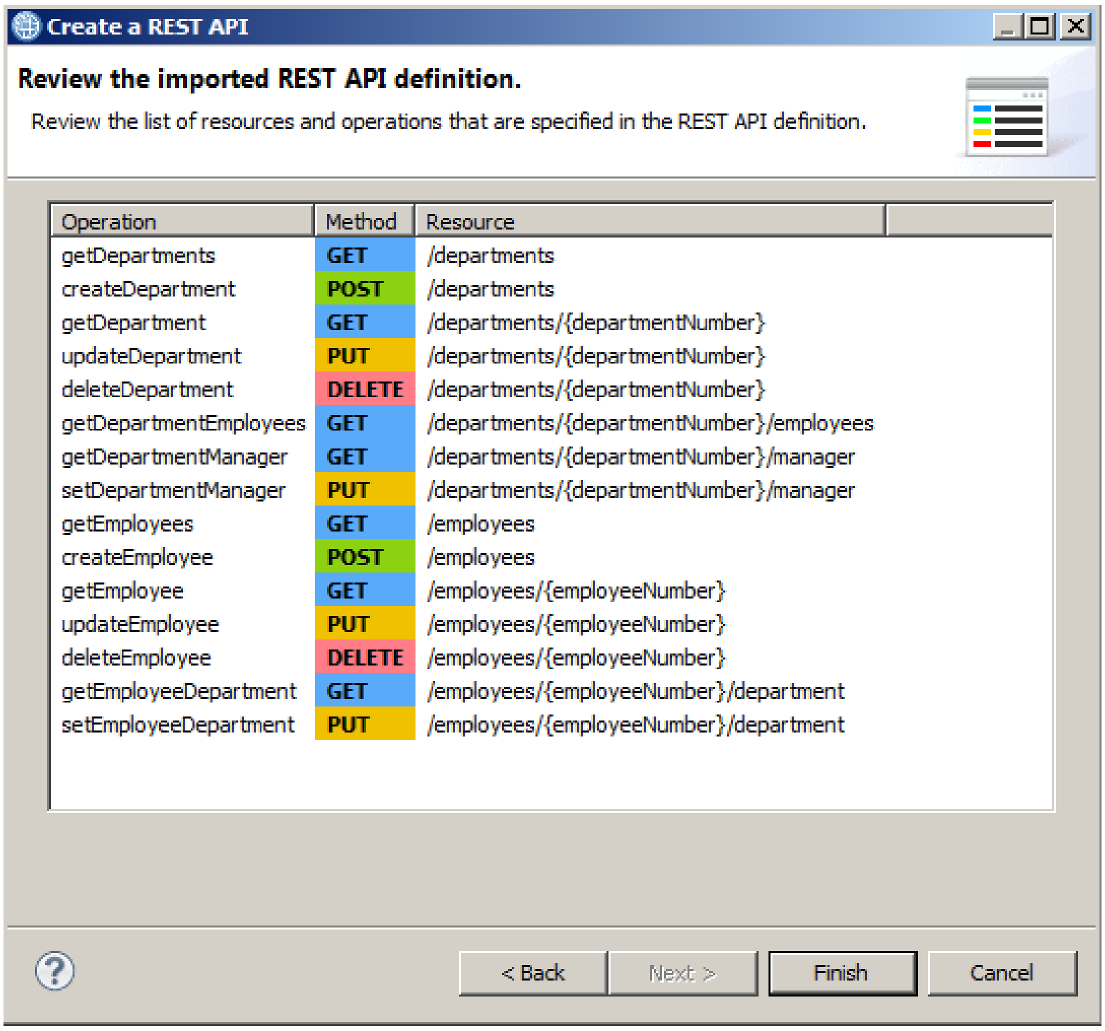
EmployeeService_REST

.

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 (ie. retrieve a list of all employees), and an operation named getEmployee. This lab will implement the **getEmployee** operation.

   Click Finish.



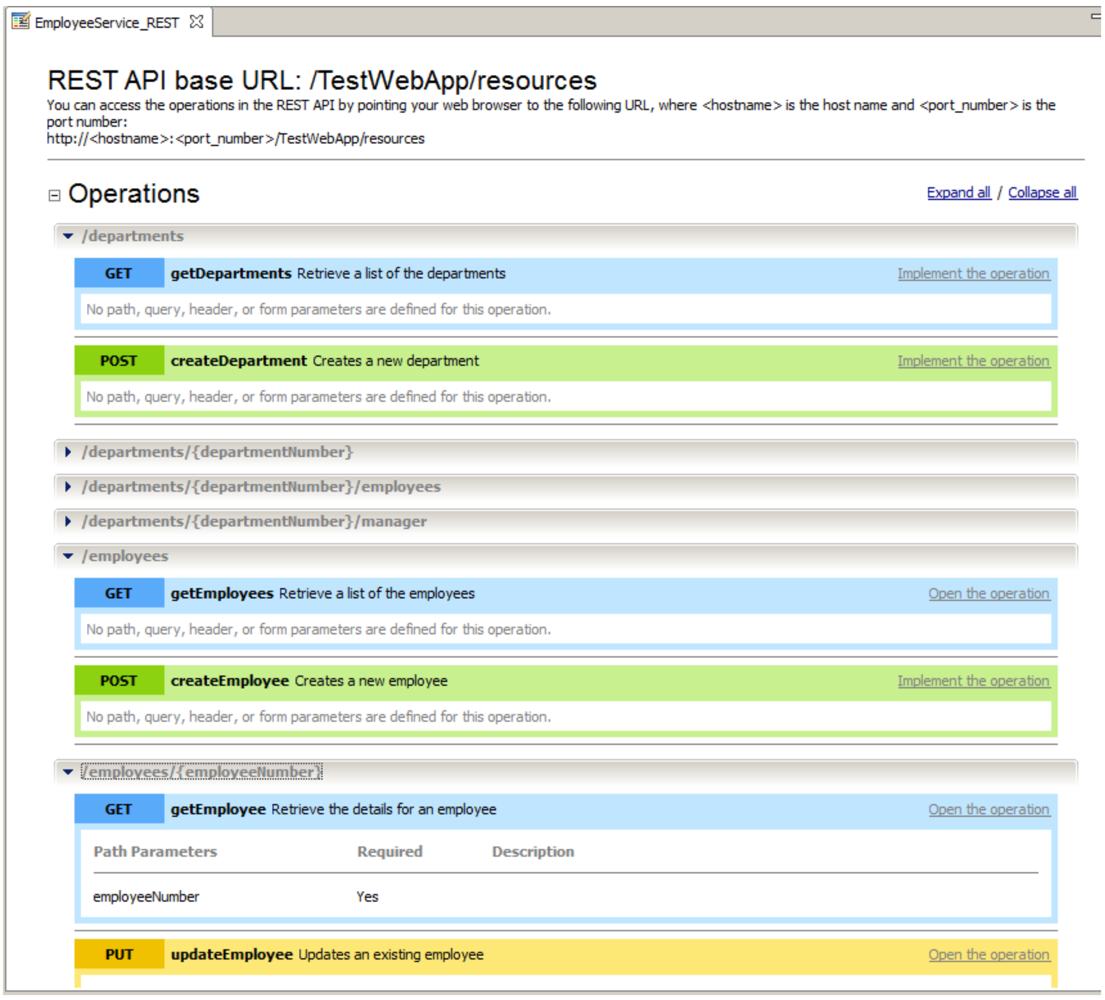
Review the imported REST API definition.

Review the list of resources and operations that are specified in the REST API definition.

Operation	Method	Resource
getDepartments	GET	/departments
createDepartment	POST	/departments
getDepartment	GET	/departments/{departmentNumber}
updateDepartment	PUT	/departments/{departmentNumber}
deleteDepartment	DELETE	/departments/{departmentNumber}
getDepartmentEmployees	GET	/departments/{departmentNumber}/employees
getDepartmentManager	GET	/departments/{departmentNumber}/manager
setDepartmentManager	PUT	/departments/{departmentNumber}/manager
getEmployees	GET	/employees
createEmployee	POST	/employees
getEmployee	GET	/employees/{employeeNumber}
updateEmployee	PUT	/employees/{employeeNumber}
deleteEmployee	DELETE	/employees/{employeeNumber}
getEmployeeDepartment	GET	/employees/{employeeNumber}/department
setEmployeeDepartment	PUT	/employees/{employeeNumber}/department

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.



EmployeeService_REST

REST API base URL: /TestWebApp/resources

You can access the operations in the REST API by pointing your web browser to the following URL, where <hostname> is the host name and <port_number> is the port number:
http://<hostname>:<port_number>/TestWebApp/resources

Operations — Expand all / Collapse all

/departments
- GET **getDepartments** Retrieve a list of the departments — Implement the operation
  No path, query, header, or form parameters are defined for this operation.
- POST **createDepartment** Creates a new department — Implement the operation
  No path, query, header, or form parameters are defined for this operation.

/departments/{departmentNumber}

/departments/{departmentNumber}/employees

/departments/{departmentNumber}/manager

/employees
- GET **getEmployees** Retrieve a list of the employees — Open the operation
  No path, query, header, or form parameters are defined for this operation.
- POST **createEmployee** Creates a new employee — Implement the operation
  No path, query, header, or form parameters are defined for this operation.

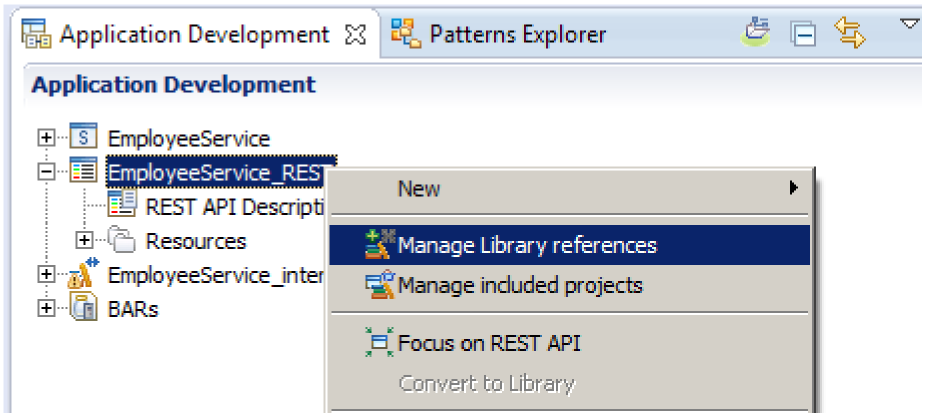
/employees/{employeeNumber}
- GET **getEmployee** Retrieve the details for an employee — Open the operation

Path Parameters	Required	Description
employeeNumber	Yes	

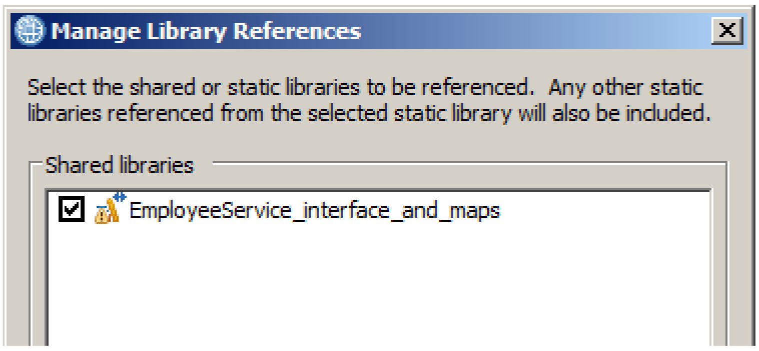
- PUT **updateEmployee** Updates an existing employee — Open the operation

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.



Select the shared or static libraries to be referenced. Any other static libraries referenced from the selected static library will also be included.

Shared libraries

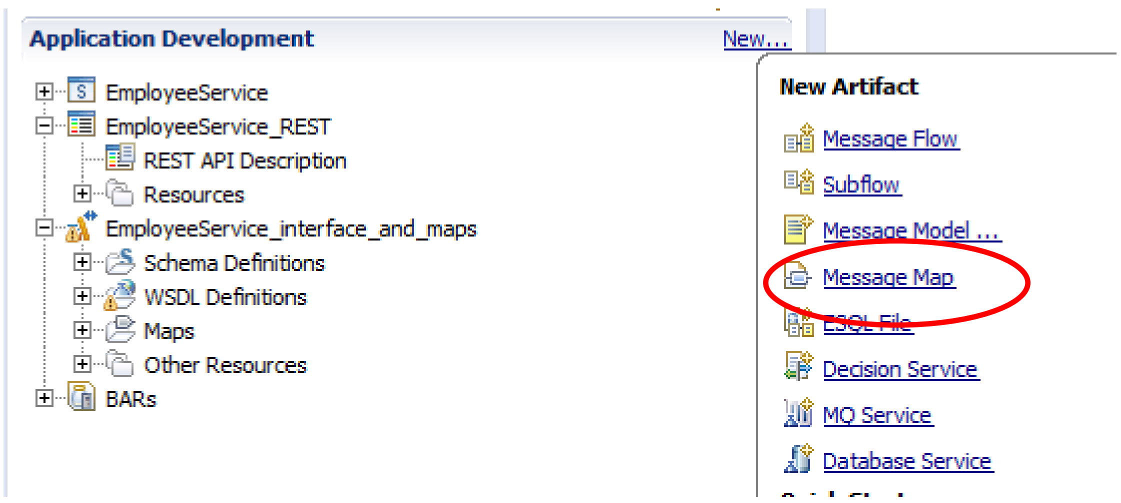
☑ EmployeeService_interface_and_maps

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 New…

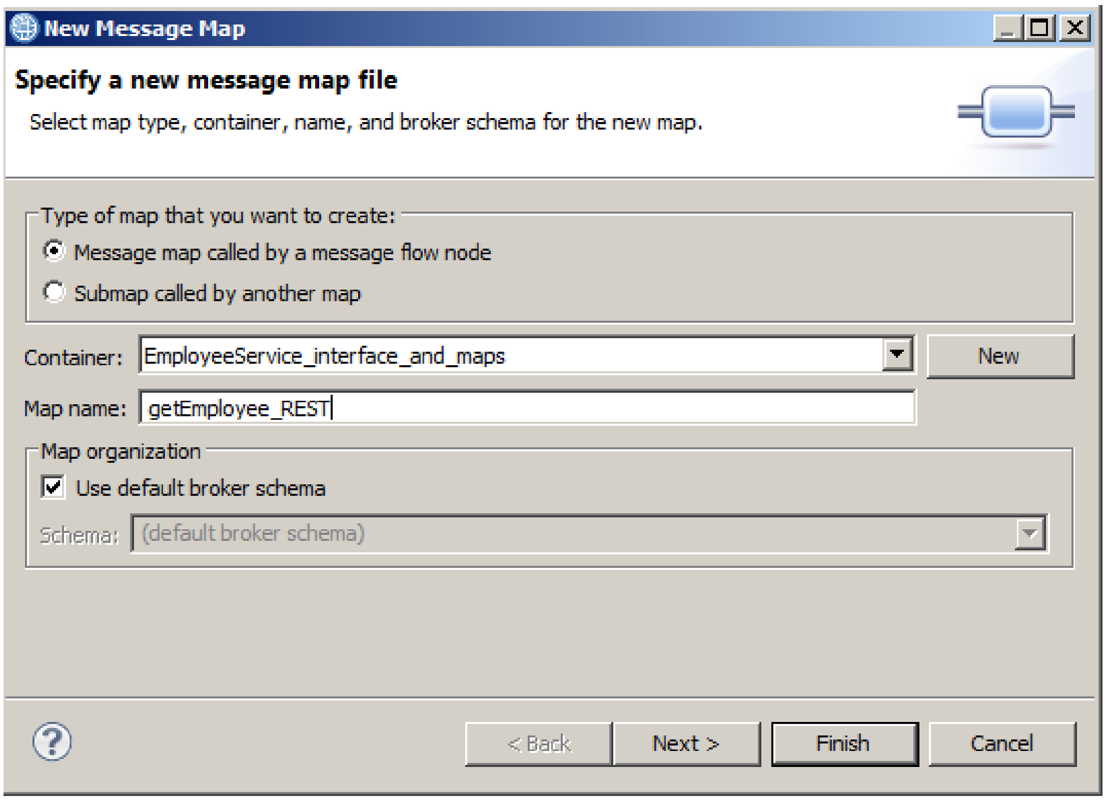
- EmployeeService
- EmployeeService_REST
  - REST API Description
  - Resources
- EmployeeService_interface_and_maps
  - Schema Definitions
  - WSDL Definitions
  - Maps
  - Other Resources
- BARs

New Artifact

- Message Flow
- Subflow
- Message Model …
- Message Map
- ESQL File
- Decision Service
- MQ Service
- Database Service

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

   Click Next.



New Message Map

Specify a new message map file

Select map type, container, name, and broker schema for the new map.

Type of map that you want to create:
- (●) Message map called by a message flow node
- ( ) Submap called by another map

Container: EmployeeService_interface_and_maps New

Map name: getEmployee_REST

Map organization
- [✓] Use default broker schema

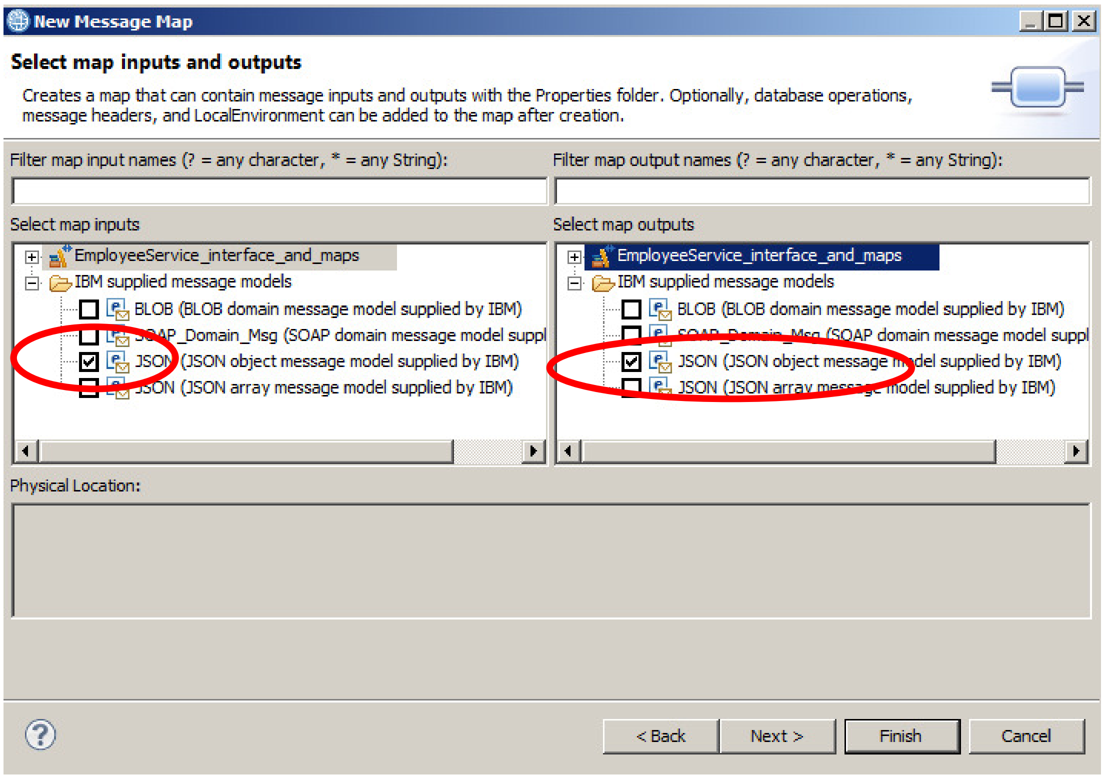
Schema: (default broker schema)

< Back | Next > | Finish | Cancel

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.



New Message Map

Select map inputs and outputs

Creates a map that can contain message inputs and outputs with the Properties folder. Optionally, database operations, message headers, and LocalEnvironment can be added to the map after creation.

Filter map input names (? = any character, * = any String):

Filter map output names (? = any character, * = any String):

Select map inputs

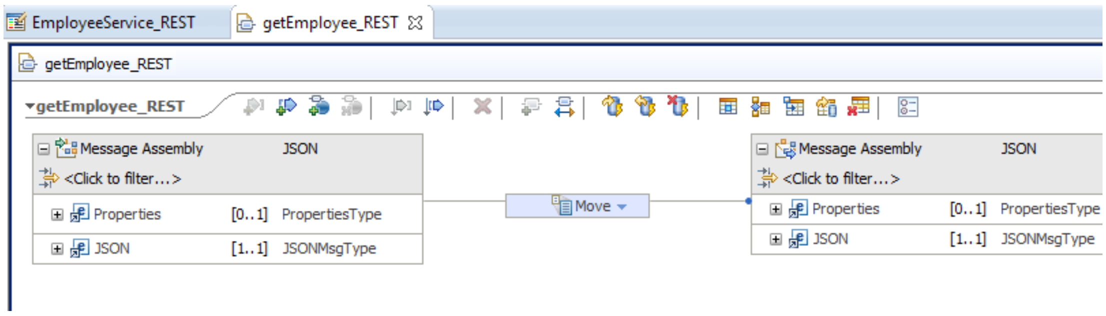
- EmployeeService_interface_and_maps
- IBM supplied message models
  - [ ] BLOB (BLOB domain message model supplied by IBM)
  - [ ] SOAP_Domain_Msg (SOAP domain message model suppl
  - [✓] JSON (JSON object message model supplied by IBM)
  - [ ] JSON (JSON array message model supplied by IBM)

Select map outputs

- EmployeeService_interface_and_maps
- IBM supplied message models
  - [ ] BLOB (BLOB domain message model supplied by IBM)
  - [ ] SOAP_Domain_Msg (SOAP domain message model suppl
  - [✓] JSON (JSON object message model supplied by IBM)
  - [ ] JSON (JSON array message model supplied by IBM)

Physical Location:

4. The basic mappings will be shown.

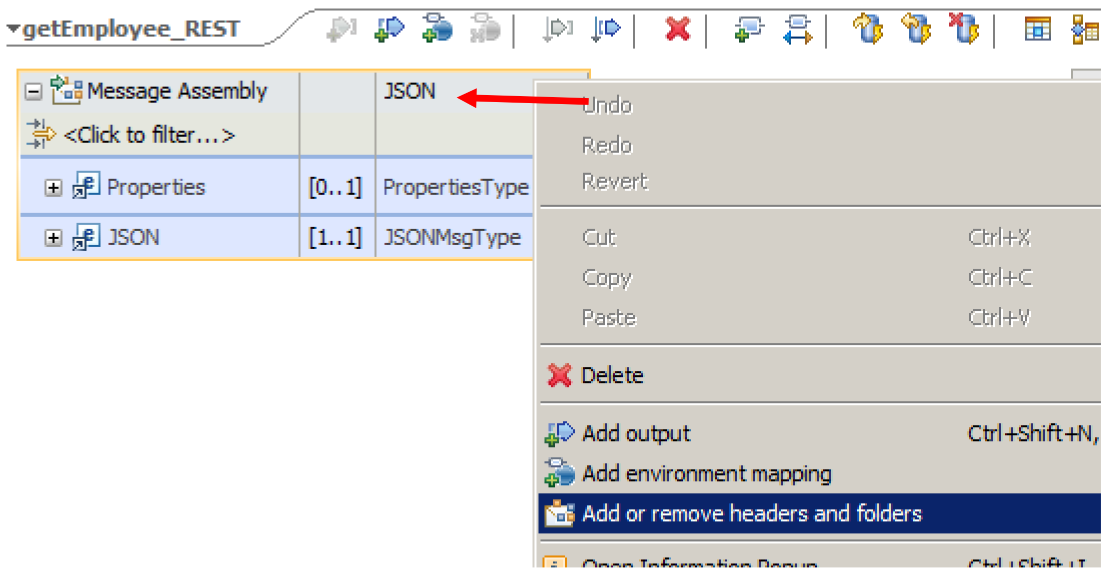


Message Assembly		JSON
<Click to filter...>		
Properties	[0..1]	PropertiesType
JSON	[1..1]	JSONMsgType

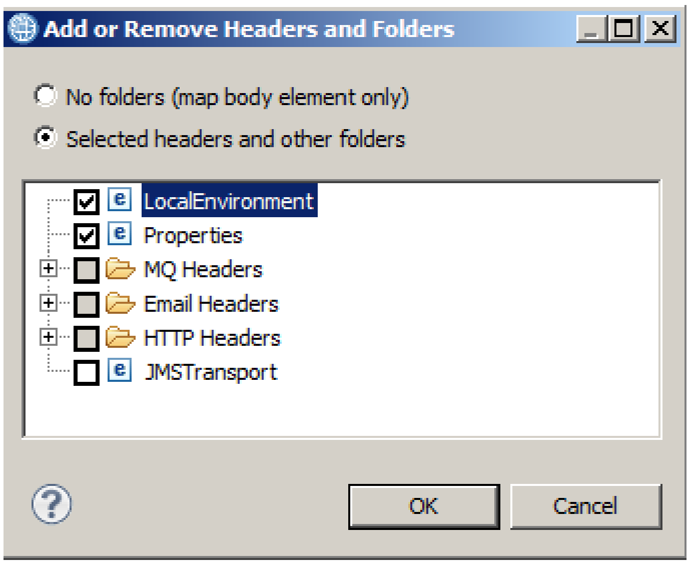
Message Assembly		JSON
<Click to filter...>		
Properties	[0..1]	PropertiesType
JSON	[1..1]	JSONMsgType

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".



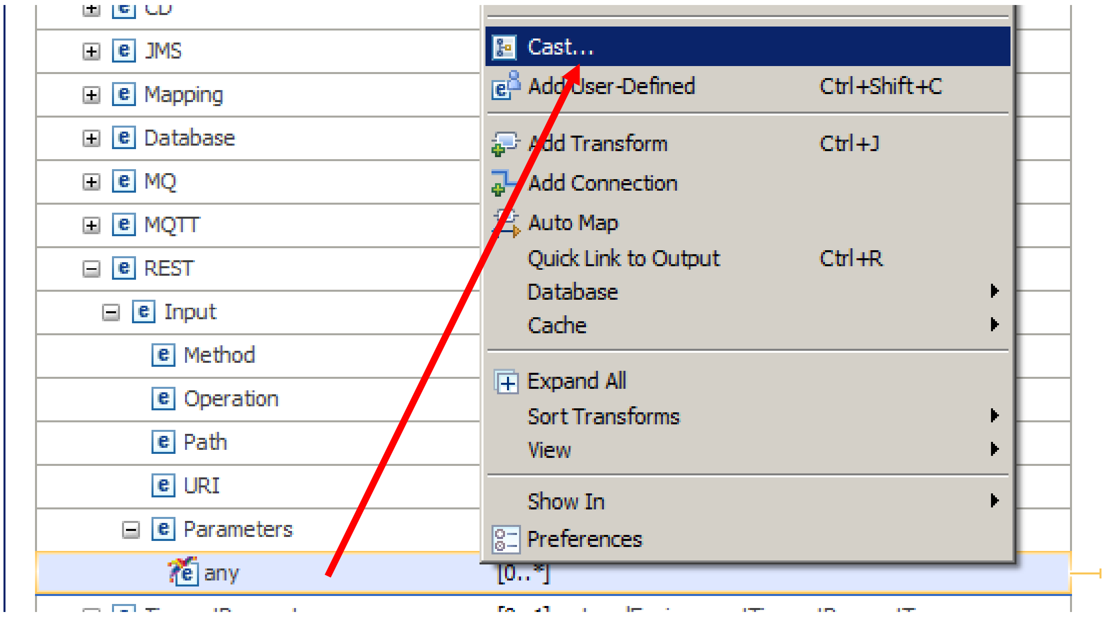
   Select the LocalEnvironment and click OK.


Add or Remove Headers and Folders
No folders (map body element only)
Selected headers and other folders
LocalEnvironment
Properties
MQ Headers
Email Headers
HTTP Headers
JMSTransport
OK
Cancel

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.

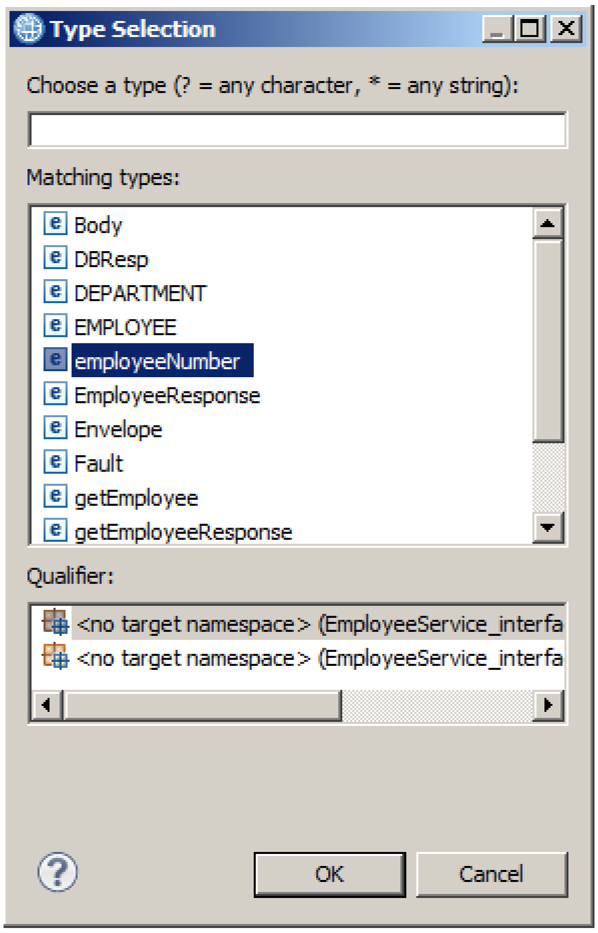
   Right-click the "any" element and select "Cast".



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.



Type Selection

Choose a type (? = any character, * = any string):

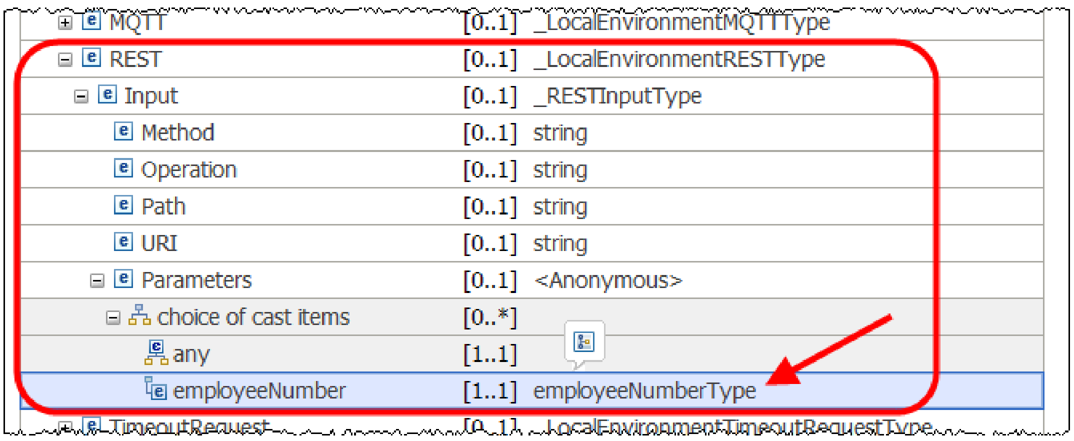
Matching types:

- Body
- DBResp
- DEPARTMENT
- EMPLOYEE
- employeeNumber
- EmployeeResponse
- Envelope
- Fault
- getEmployee
- getEmployeeResponse

Qualifier:

- <no target namespace> (EmployeeService_interfa
- <no target namespace> (EmployeeService_interfa

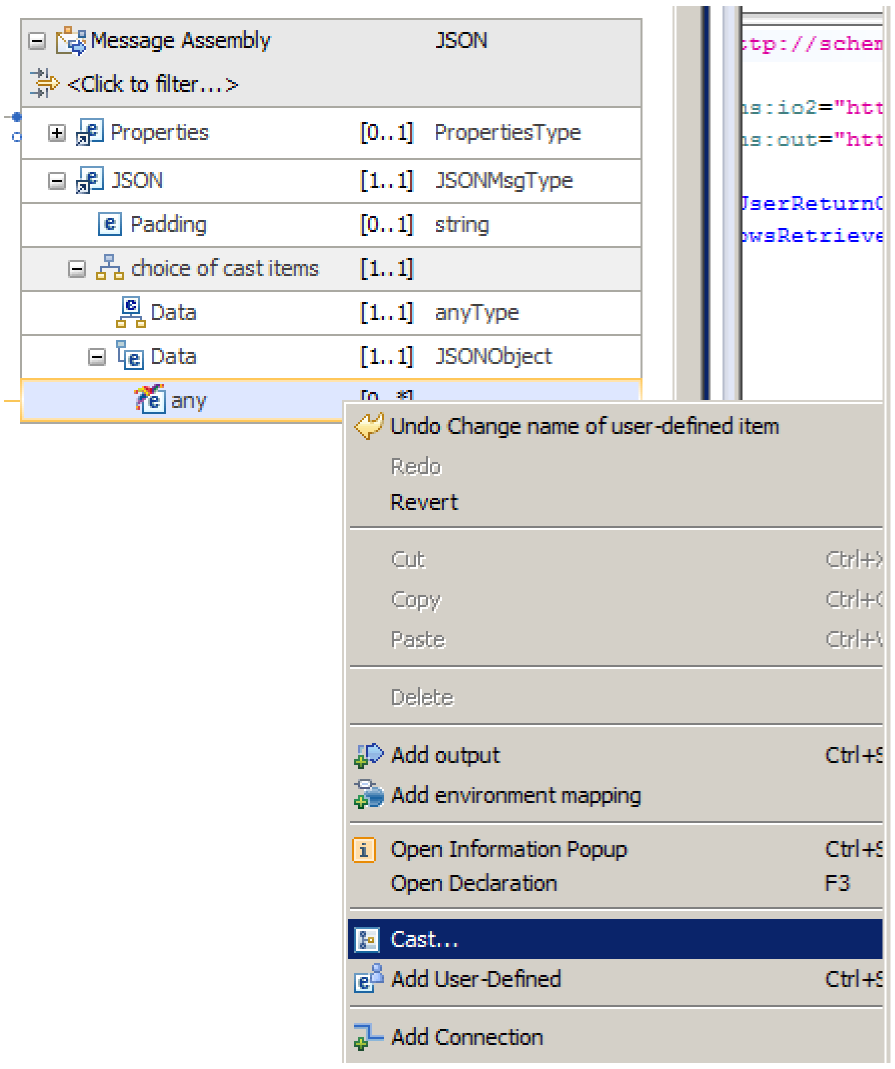
When complete, the element should look like this:



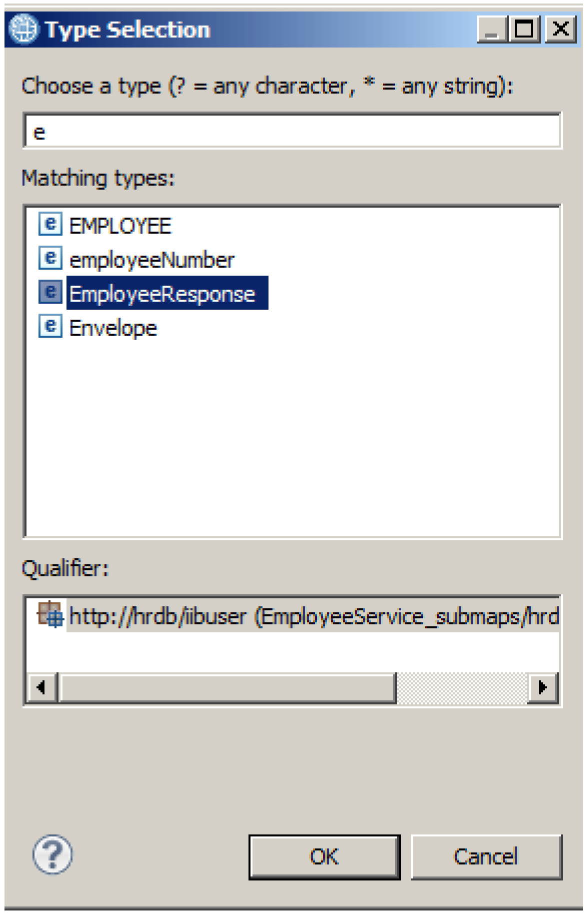
MQTT	[0..1]	_LocalEnvironmentMQTTType
REST	[0..1]	_LocalEnvironmentRESTType
Input	[0..1]	_RESTInputType
Method	[0..1]	string
Operation	[0..1]	string
Path	[0..1]	string
URI	[0..1]	string
Parameters	[0..1]	<Anonymous>
choice of cast items	[0..*]	
any	[1..1]	
employeeNumber	[1..1]	employeeNumberType
TimeoutRequest	[0..1]	_LocalEnvironmentTimeoutRequestType

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.



Type Selection

Choose a type (? = any character, * = any string):

e

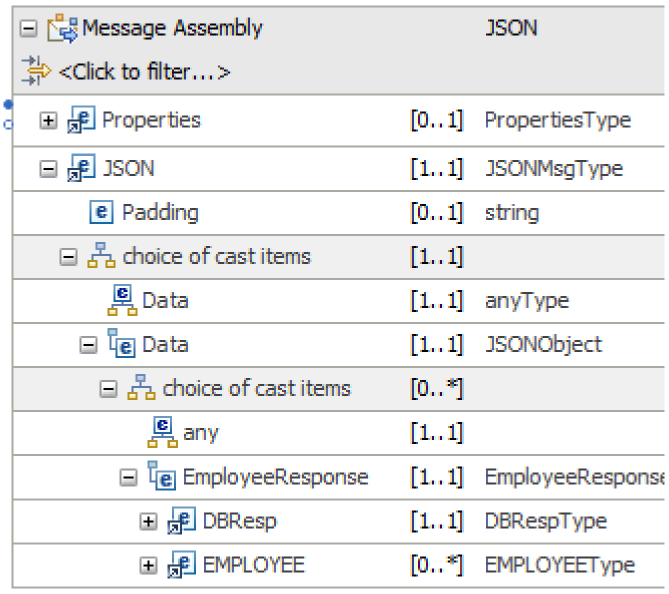
Matching types:

- EMPLOYEE
- employeeNumber
- EmployeeResponse
- Envelope

Qualifier:

http://hrdb/iibuser (EmployeeService_submaps/hrd

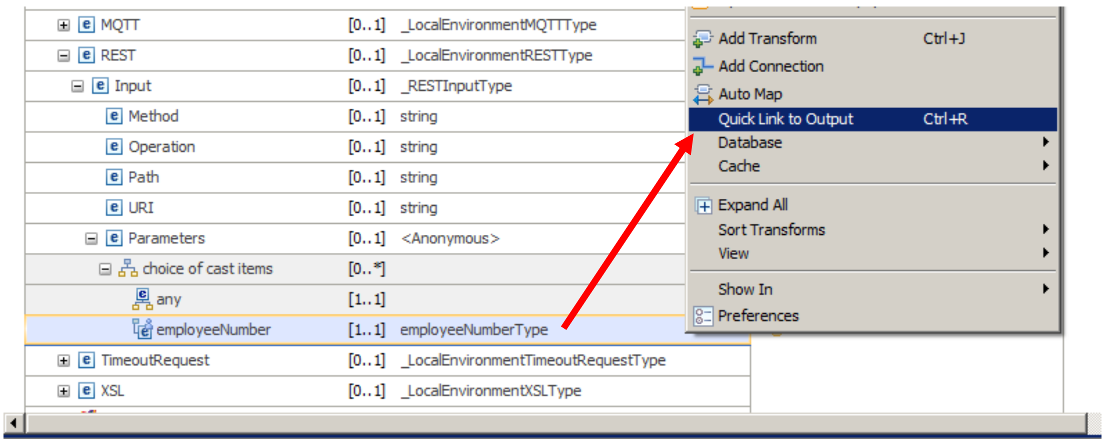
The output assembly will now look like this.



Message Assembly		JSON
<Click to filter...>		
Properties	[0..1]	PropertiesType
JSON	[1..1]	JSONMsgType
Padding	[0..1]	string
choice of cast items	[1..1]	
Data	[1..1]	anyType
Data	[1..1]	JSONObject
choice of cast items	[0..*]	
any	[1..1]	
EmployeeResponse	[1..1]	EmployeeResponse
DBResp	[1..1]	DBRespType
EMPLOYEE	[0..*]	EMPLOYEEType

11. The new element employeeNumber now needs to be mapped to the output JSON/EmployeeResponse assembly, using a submap.

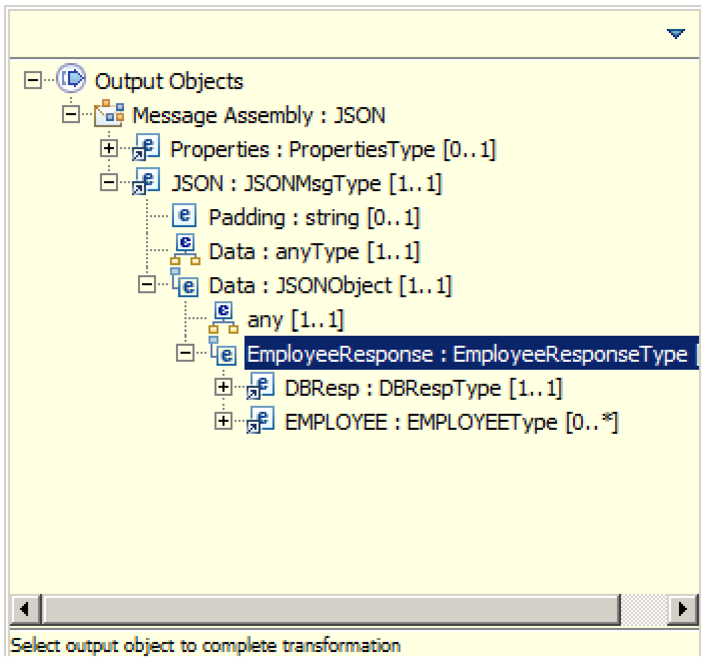
    Unfortunately, expanding the input Local Environment has probably meant that the output message has disappeared from the map display. To handle this, right-click employeeNumber, and select "Quick Link to Output".



MQTT	[0..1]	_LocalEnvironmentMQTTType
REST	[0..1]	_LocalEnvironmentRESTType
Input	[0..1]	_RESTInputType
Method	[0..1]	string
Operation	[0..1]	string
Path	[0..1]	string
URI	[0..1]	string
Parameters	[0..1]	<Anonymous>
choice of cast items	[0..*]	
any	[1..1]	
employeeNumber	[1..1]	employeeNumberType
TimeoutRequest	[0..1]	_LocalEnvironmentTimeoutRequestType
XSL	[0..1]	_LocalEnvironmentXSLType

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.

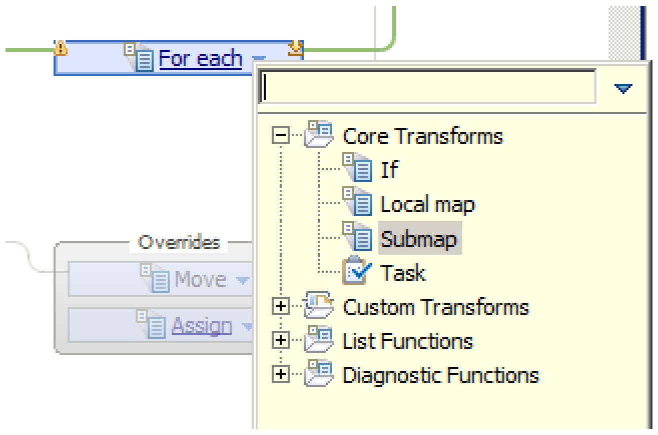


```
Output Objects
  Message Assembly : JSON
    Properties : PropertiesType [0..1]
    JSON : JSONMsgType [1..1]
      Padding : string [0..1]
      Data : anyType [1..1]
      Data : JSONObject [1..1]
        any [1..1]
        EmployeeResponse : EmployeeResponseType
          DBResp : DBRespType [1..1]
          EMPLOYEE : EMPLOYEEType [0..*]
```

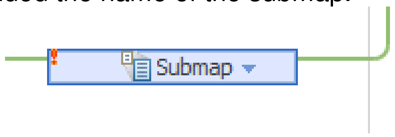
Select output object to complete transformation

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.

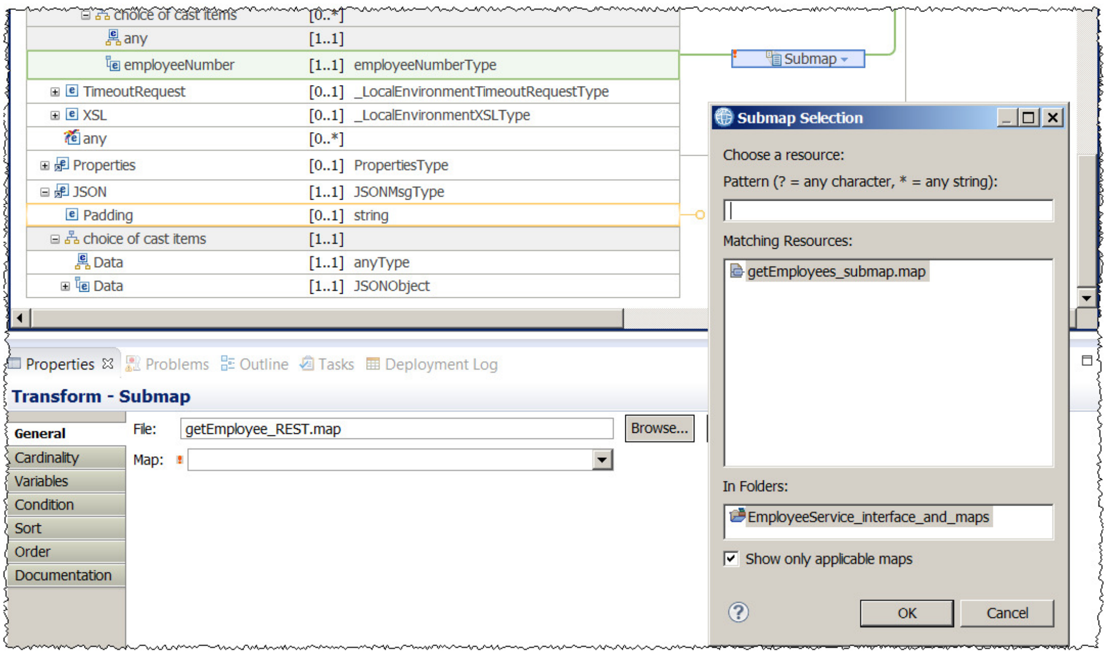


14. Set the Properties of the submap to reference the required submap.

    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).



choice of cast items	[0..*]	
any	[1..1]	
employeeNumber	[1..1]	employeeNumberType
TimeoutRequest	[0..1]	_LocalEnvironmentTimeoutRequestType
XSL	[0..1]	_LocalEnvironmentXSLType
any	[0..*]	
Properties	[0..1]	PropertiesType
JSON	[1..1]	JSONMsgType
Padding	[0..1]	string
choice of cast items	[1..1]	
Data	[1..1]	anyType
Data	[1..1]	JSONObject

Transform - Submap

General
Cardinality
Variables
Condition
Sort
Order
Documentation

File: getEmployee_REST.map Browse...
Map:

Submap Selection

Choose a resource:
Pattern (? = any character, * = any string):

Matching Resources:
getEmployees_submap.map

In Folders:
EmployeeService_interface_and_maps

☑ Show only applicable maps

OK Cancel

.

The main map is now complete, so save (Ctrl-S) and close.

2.5 Implement the getEmployee operation

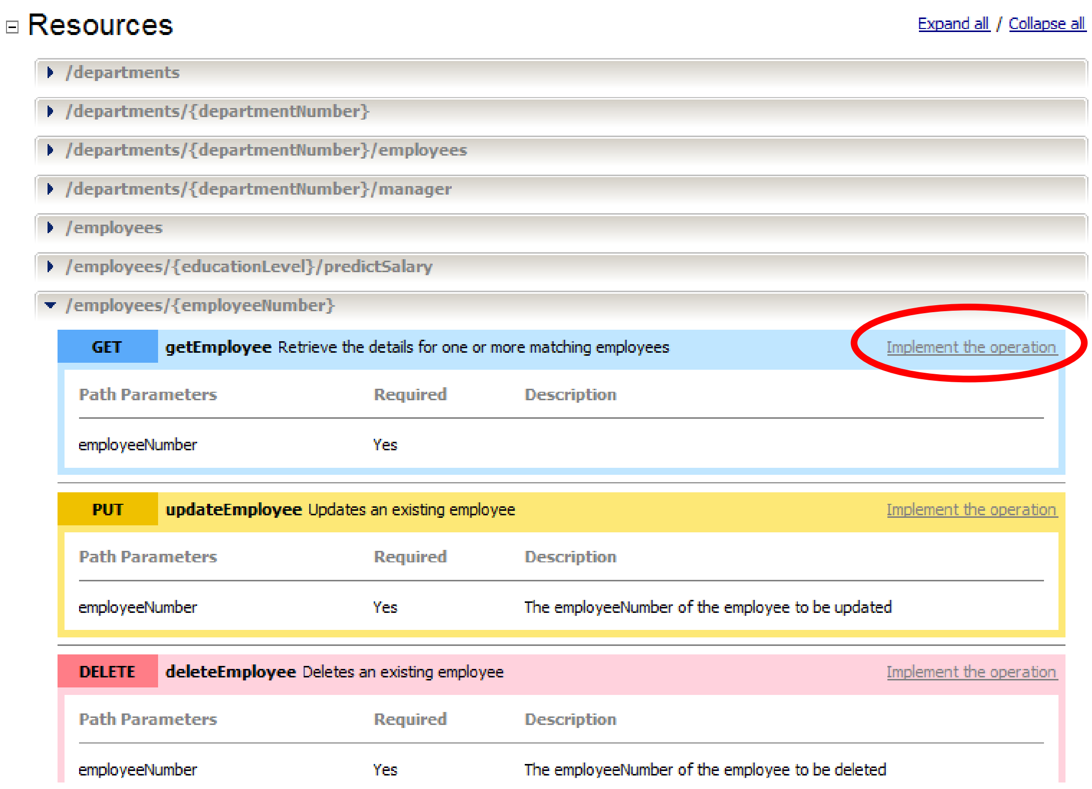
1. You will now implement the getEmployee operation of the REST API.

   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

Expand all / Collapse all

- /departments
- /departments/{departmentNumber}
- /departments/{departmentNumber}/employees
- /departments/{departmentNumber}/manager
- /employees
- /employees/{educationLevel}/predictSalary
- /employees/{employeeNumber}

GET **getEmployee** Retrieve the details for one or more matching employees — Implement the operation

Path Parameters	Required	Description
employeeNumber	Yes	

PUT **updateEmployee** Updates an existing employee — Implement the operation

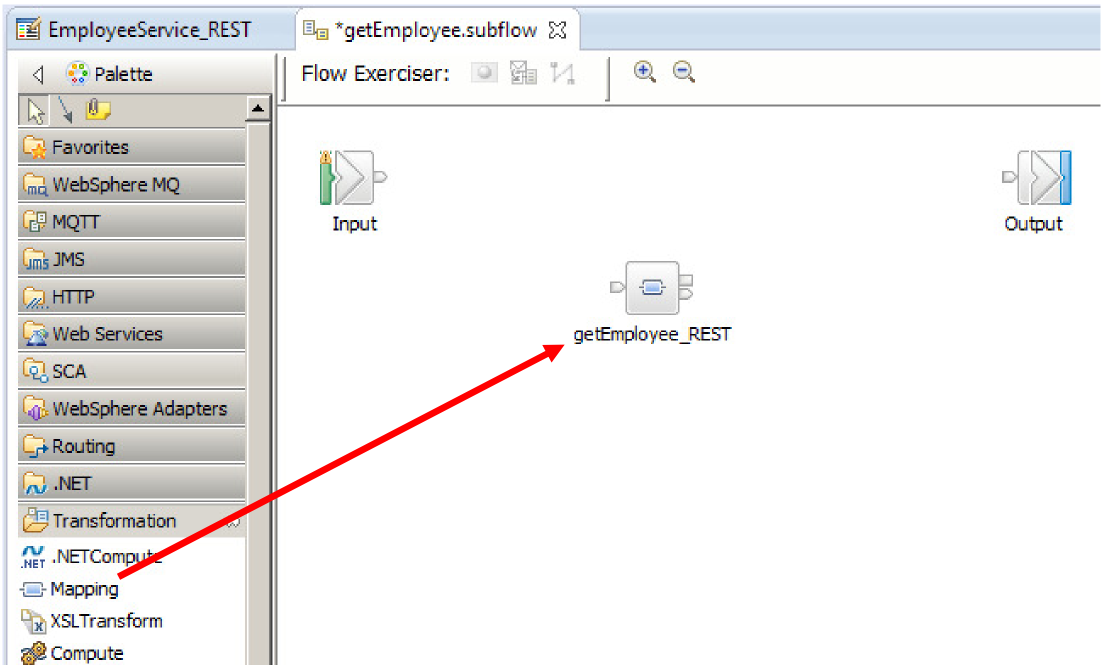
Path Parameters	Required	Description
employeeNumber	Yes	The employeeNumber of the employee to be updated

DELETE **deleteEmployee** Deletes an existing employee — Implement the operation

Path Parameters	Required	Description
employeeNumber	Yes	The employeeNumber of the employee to be deleted

2. Drop a Mapping node onto the flow editor.

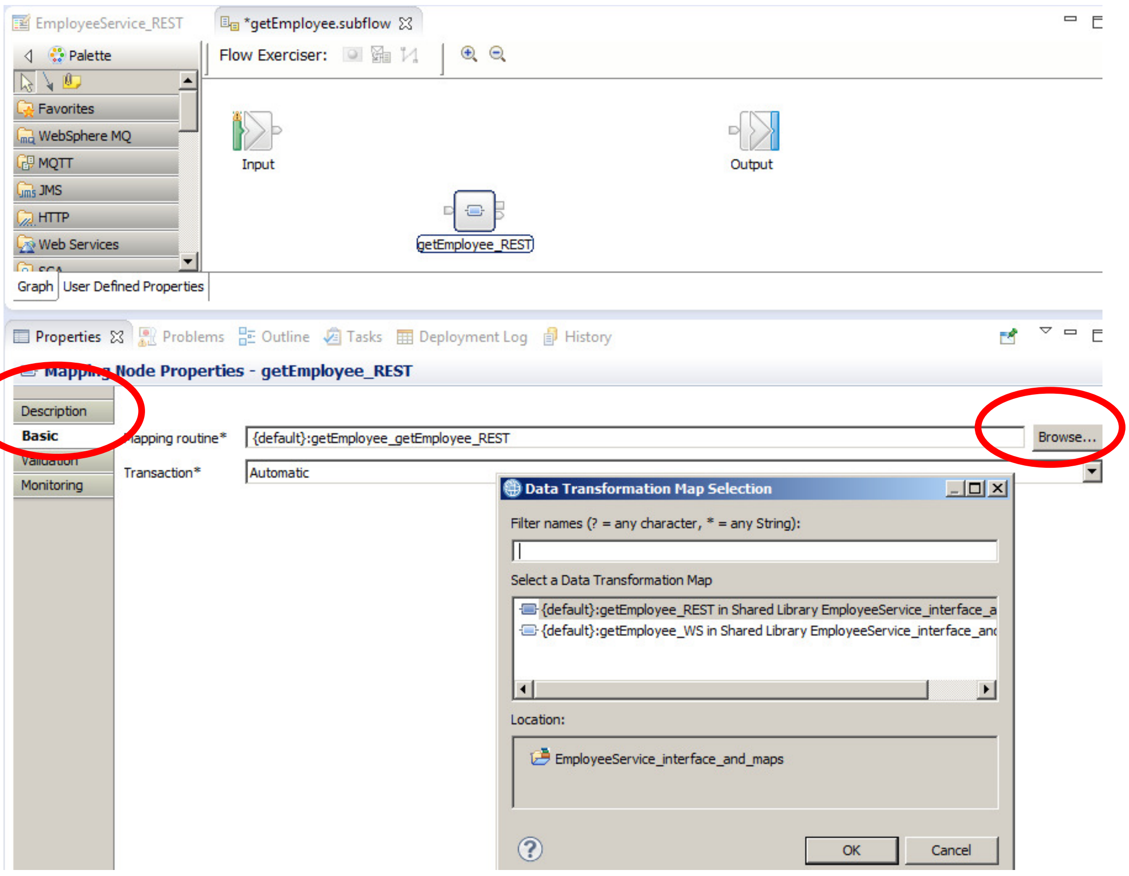
   Name the new map "getEmployee_REST".



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.



Mapping Node Properties - getEmployee_REST

Mapping routine* {default}:getEmployee_getEmployee_REST

Transaction* Automatic

Filter names (? = any character, * = any String):

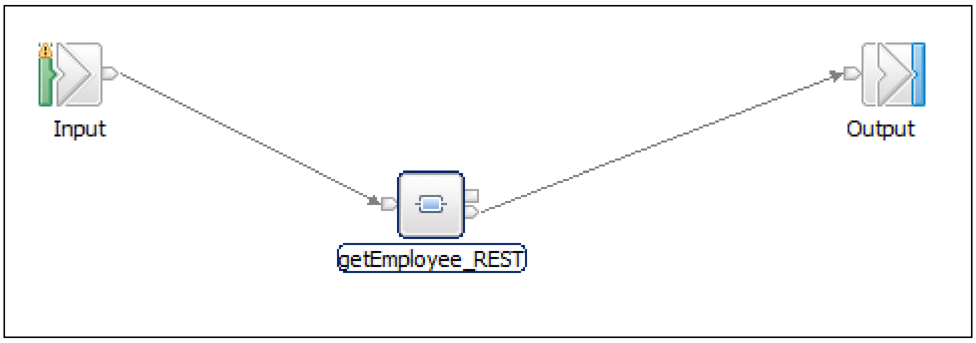
Select a Data Transformation Map

{default}:getEmployee_REST in Shared Library EmployeeService_interface_a
{default}:getEmployee_WS in Shared Library EmployeeService_interface_anc

Location:

EmployeeService_interface_and_maps

4. Connect the nodes as shown. Save and close the subflow.



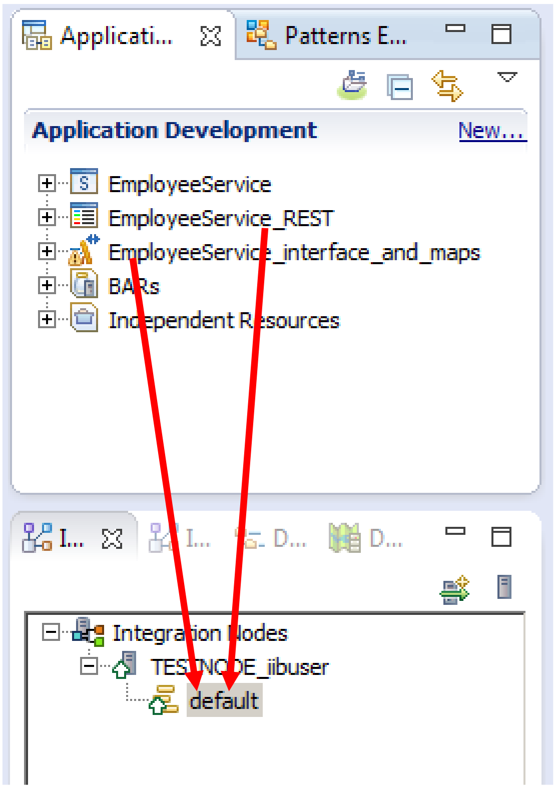
3. Test the EmployeeService REST API

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

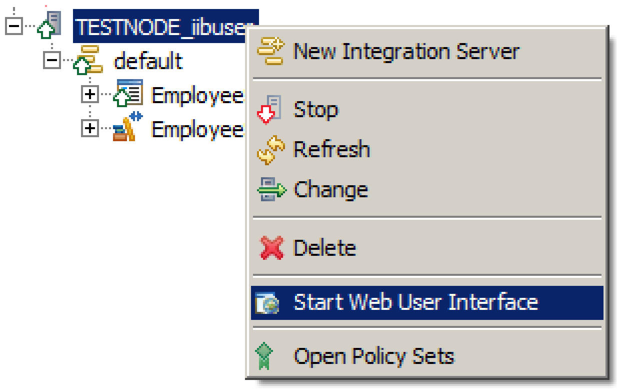
1. In the navigator, deploy the following resources, in order, to the default server.

    1. EmployeeService_interface_and_maps
    2. EmployeeService_REST



3.2 Test the service

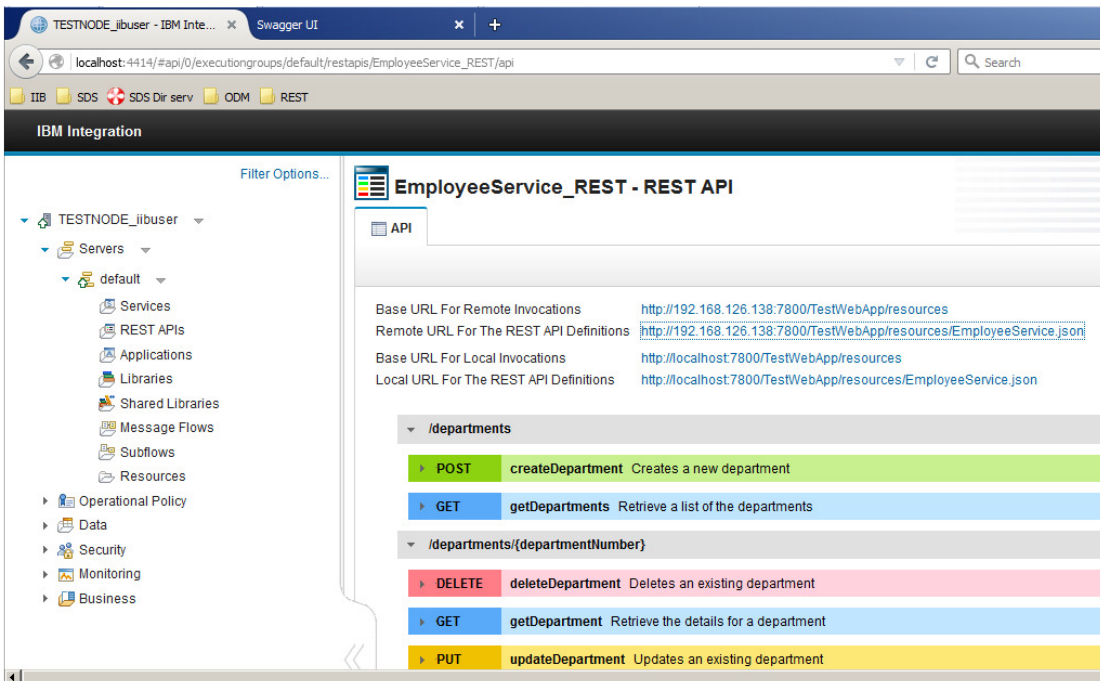
1. Open the IIB web UI by right-clicking TESTNODE_iibuser and selecting Start Web User Interface.



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).

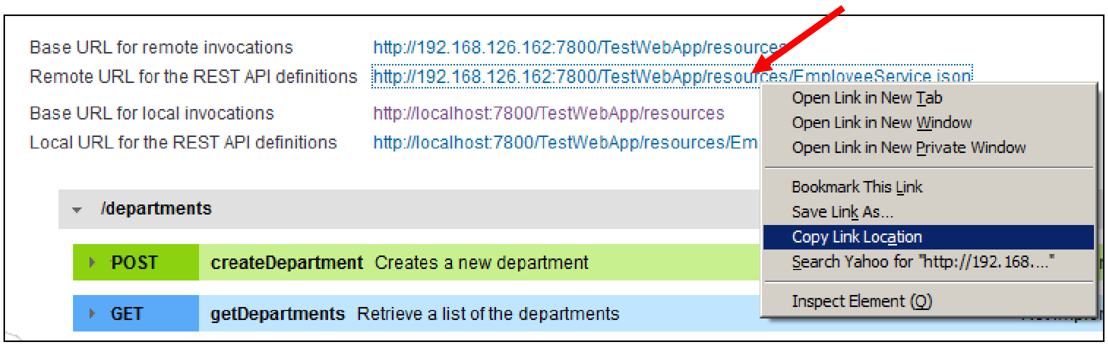


Base URL For Remote Invocations	http://192.168.126.138:7800/TestWebApp/resources
Remote URL For The REST API Definitions	http://192.168.126.138:7800/TestWebApp/resources/EmployeeService.json
Base URL For Local Invocations	http://localhost:7800/TestWebApp/resources
Local URL For The REST API Definitions	http://localhost:7800/TestWebApp/resources/EmployeeService.json

/departments
- POST createDepartment Creates a new department
- GET getDepartments Retrieve a list of the departments

/departments/{departmentNumber}
- DELETE deleteDepartment Deletes an existing department
- GET getDepartment Retrieve the details for a department
- PUT updateDepartment Updates an existing department

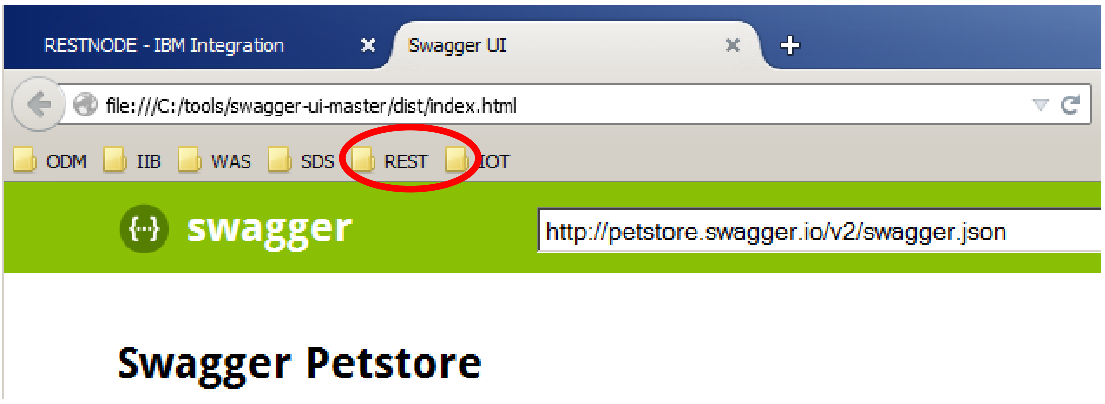
3. On the "Remote URL for REST API definitions", right-click and select "Copy Link Location".


Base URL for remote invocations http://192.168.126.162:7800/TestWebApp/resources
Remote URL for the REST API definitions http://192.168.126.162:7800/TestWebApp/resources/EmployeeService.json
Base URL for local invocations http://localhost:7800/TestWebApp/resources
Local URL for the REST API definitions http://localhost:7800/TestWebApp/resources/Em

Open Link in New Tab
Open Link in New Window
Open Link in New Private Window
Bookmark This Link
Save Link As...
Copy Link Location
Search Yahoo for "http://192.168...."
Inspect Element (Q)
/departments
POST createDepartment Creates a new department
GET getDepartments Retrieve a list of the departments

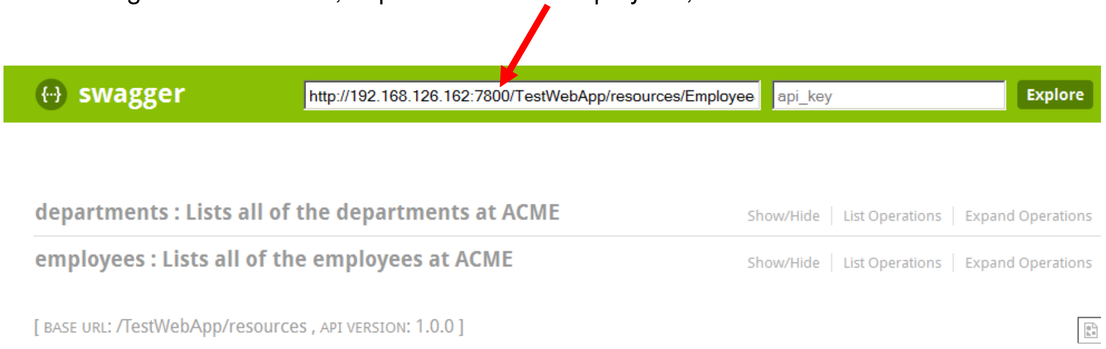
4. In Firefox, open a new tab, and open the SwaggerUI tool (using the bookmark in the REST folder).

   By default, this will open the Petstore Swagger document.


RESTNODE - IBM Integration
Swagger UI
file:///C:/tools/swagger-ui-master/dist/index.html
ODM IIB WAS SDS REST IOT
swagger
http://petstore.swagger.io/v2/swagger.json
Swagger Petstore

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

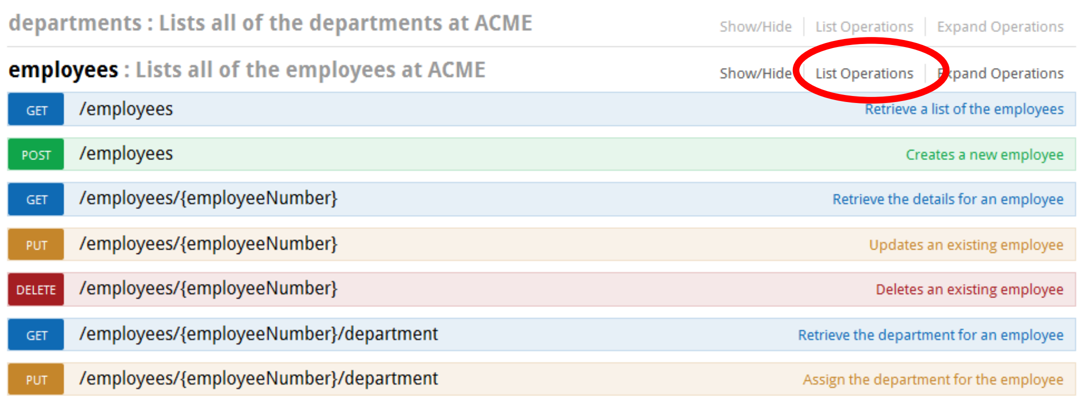
   The two high-level functions, departments and employees, will be shown.



departments : Lists all of the departments at ACME Show/Hide | List Operations | Expand Operations
employees : Lists all of the employees at ACME Show/Hide | List Operations | Expand Operations
[BASE URL: /TestWebApp/resources , API VERSION: 1.0.0]

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.



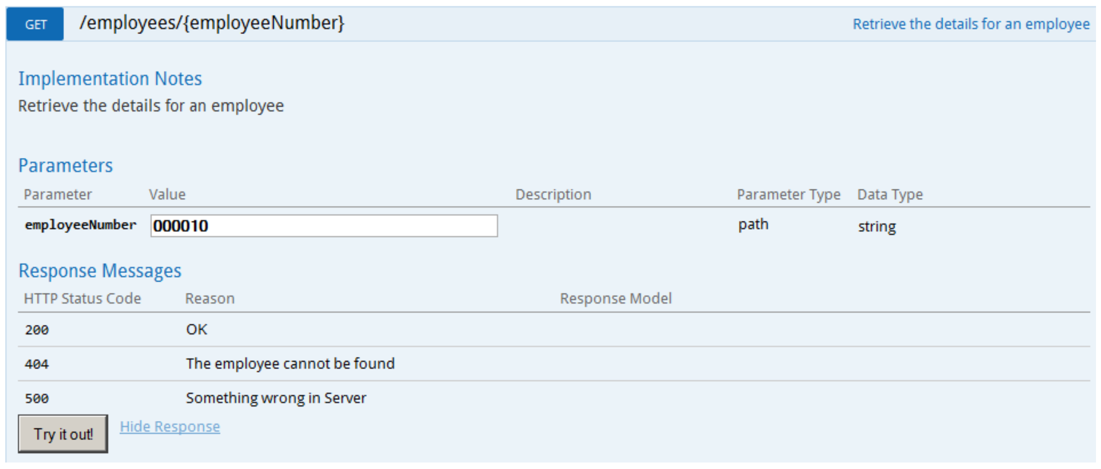
departments : Lists all of the departments at ACME — Show/Hide | List Operations | Expand Operations

employees : Lists all of the employees at ACME — Show/Hide | List Operations | Expand Operations

GET	/employees	Retrieve a list of the employees
POST	/employees	Creates a new employee
GET	/employees/{employeeNumber}	Retrieve the details for an employee
PUT	/employees/{employeeNumber}	Updates an existing employee
DELETE	/employees/{employeeNumber}	Deletes an existing employee
GET	/employees/{employeeNumber}/department	Retrieve the department for an employee
PUT	/employees/{employeeNumber}/department	Assign the department for the employee

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.



GET /employees/{employeeNumber} — Retrieve the details for an employee

Implementation Notes

Retrieve the details for an employee

Parameters

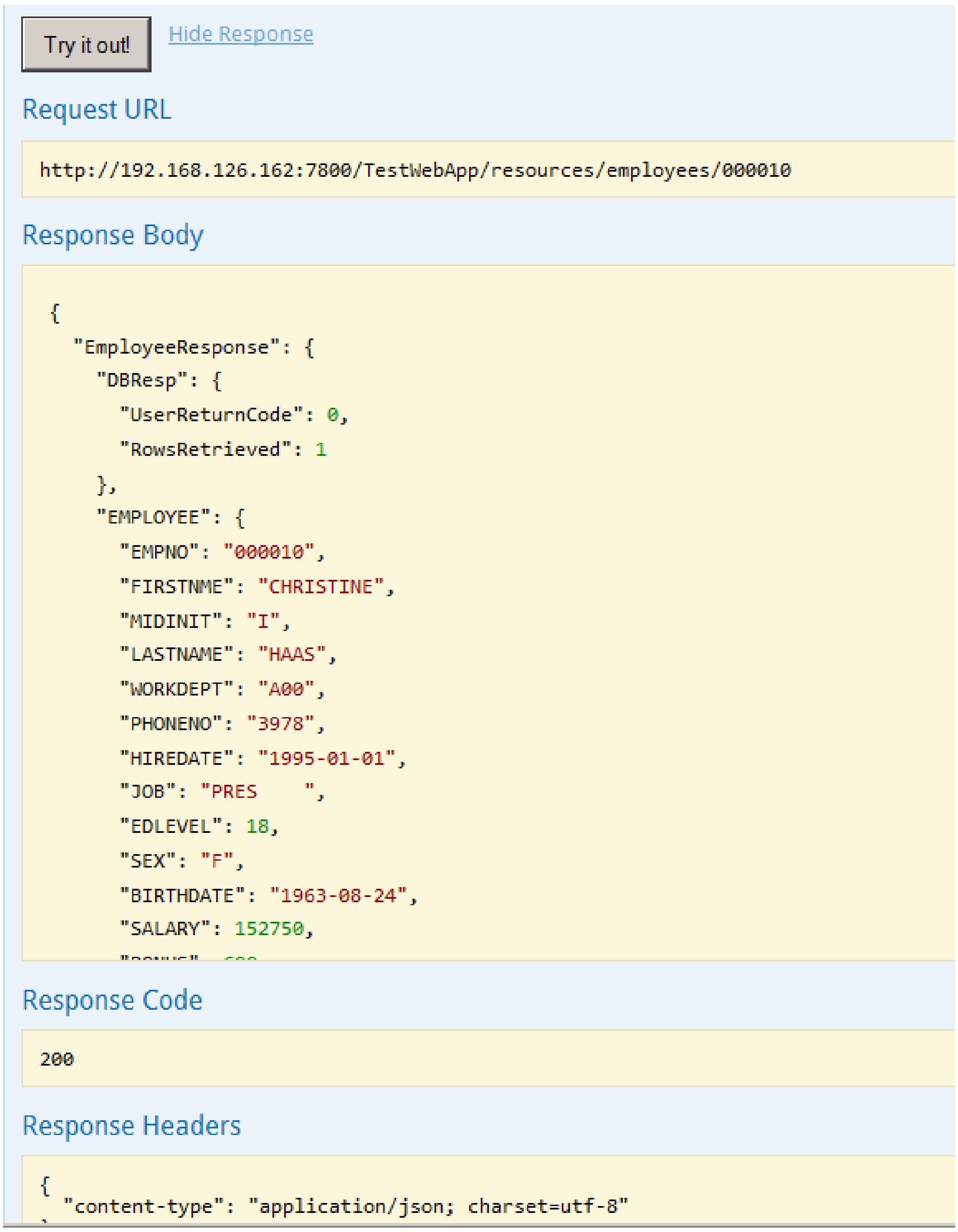
Parameter	Value	Description	Parameter Type	Data Type
employeeNumber	000010		path	string

Response Messages

HTTP Status Code	Reason	Response Model
200	OK	
404	The employee cannot be found	
500	Something wrong in Server	

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.



Request URL

```
http://192.168.126.162:7800/TestWebApp/resources/employees/000010
```

Response Body

```
{
  "EmployeeResponse": {
    "DBResp": {
      "UserReturnCode": 0,
      "RowsRetrieved": 1
    },
    "EMPLOYEE": {
      "EMPNO": "000010",
      "FIRSTNME": "CHRISTINE",
      "MIDINIT": "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

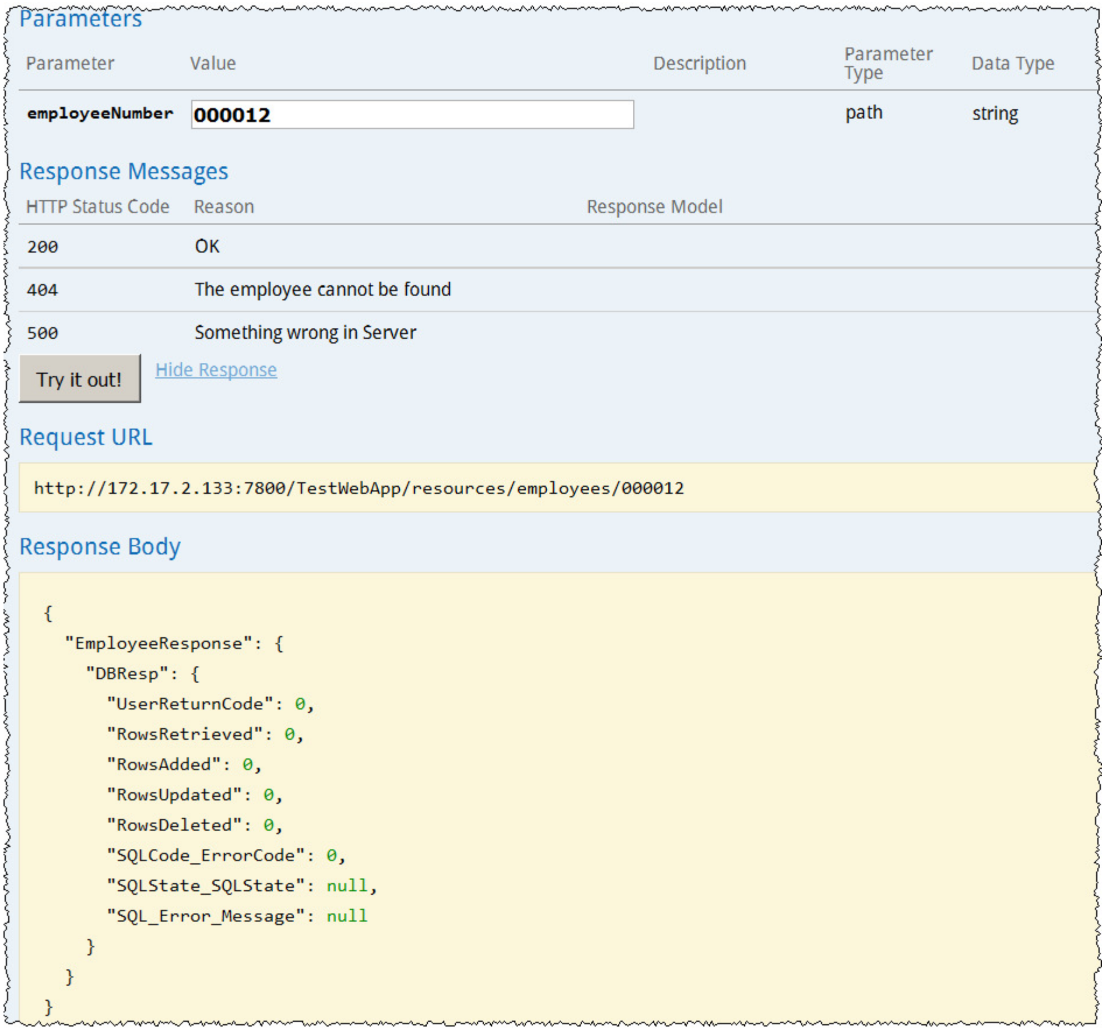
```
200
```

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).



Parameters

Parameter	Value	Description	Parameter Type	Data Type
employeeNumber	000012		path	string

Response Messages

HTTP Status Code	Reason	Response Model
200	OK	
404	The employee cannot be found	
500	Something wrong in Server	

Request URL

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

Response Body

```
{
  "EmployeeResponse": {
    "DBResp": {
      "UserReturnCode": 0,
      "RowsRetrieved": 0,
      "RowsAdded": 0,
      "RowsUpdated": 0,
      "RowsDeleted": 0,
      "SQLCode_ErrorCode": 0,
      "SQLState_SQLState": null,
      "SQL_Error_Message": null
    }
  }
}
```

END OF LAB GUIDE