L02

BetaWorks

IBM Integration Bus

Developing a REST API service

Featuring:

The REST API tools for IIB
Testing with Swagger UI

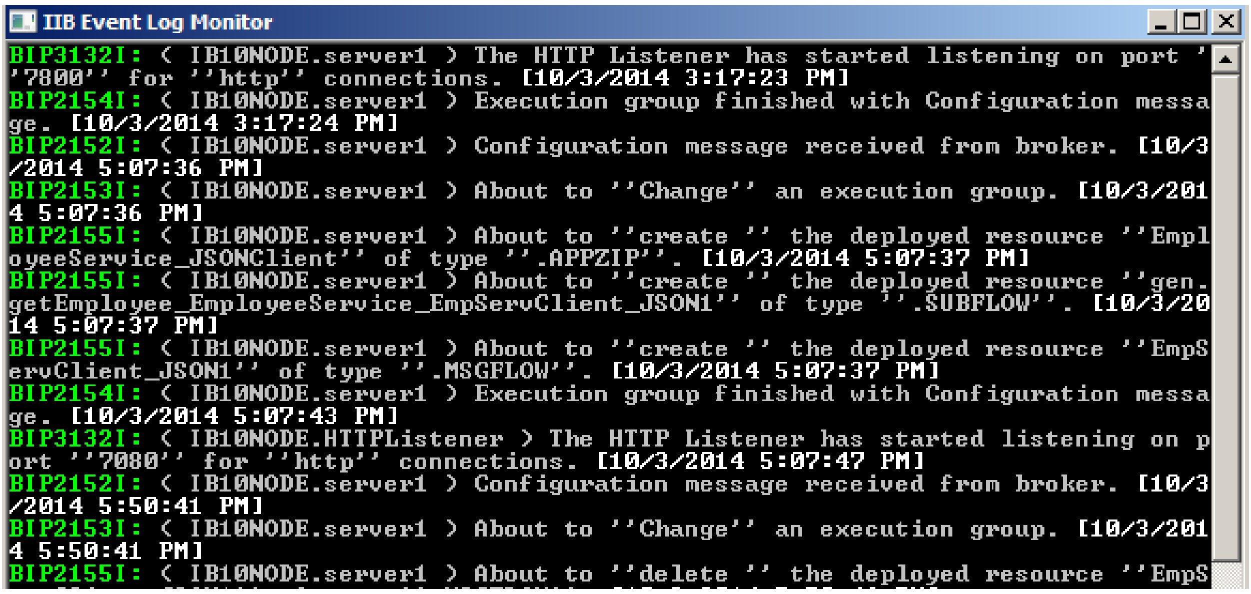
October 2015
Hands-on lab built at product
Version 10.0.0.2

1. Prepare the IIB Node

1.1 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.

1.2 Create and configure a new node for REST

The IIB support for the REST API requires some special configuration for the IIB node and server. To ensure this scenario does not interfere with other scenarios, you will create and use a separate IIB node.

1. Stop IB10NODE (right-click in Toolkit, select Stop).

2. Support for REST requires the use of the embedded HTTP listeners, rather than the node-wide listener.

   Additionally, the IIB node has to be configured for Cross-Origin Resource Sharing (CORS). For details on this, please read: http://www.w3.org/TR/cors/.

   (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).

   To configure these items, 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.

   When the node is fully restarted, you will see the following messages on the IIB Event Log viewer:

   ```
   BIP3132I: ( RESTNODE ) The HTTP Listener has started listening on port
   ''4418'' for ''WebAdmin http'' connections. [08/04/2015 09:16:21]
   BIP2152I: ( RESTNODE.default ) Configuration message received from
   integration node. [08/04/2015 09:16:22]
   BIP2153I: ( RESTNODE.default ) About to ''Start'' an integration server.
   [08/04/2015 09:16:22]
   BIP2154I: ( RESTNODE.default ) Integration server finished with
   Configuration message. [08/04/2015 09:16:22]
   ```

2. Create the REST Service

In this section you will create a new REST API service. 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.

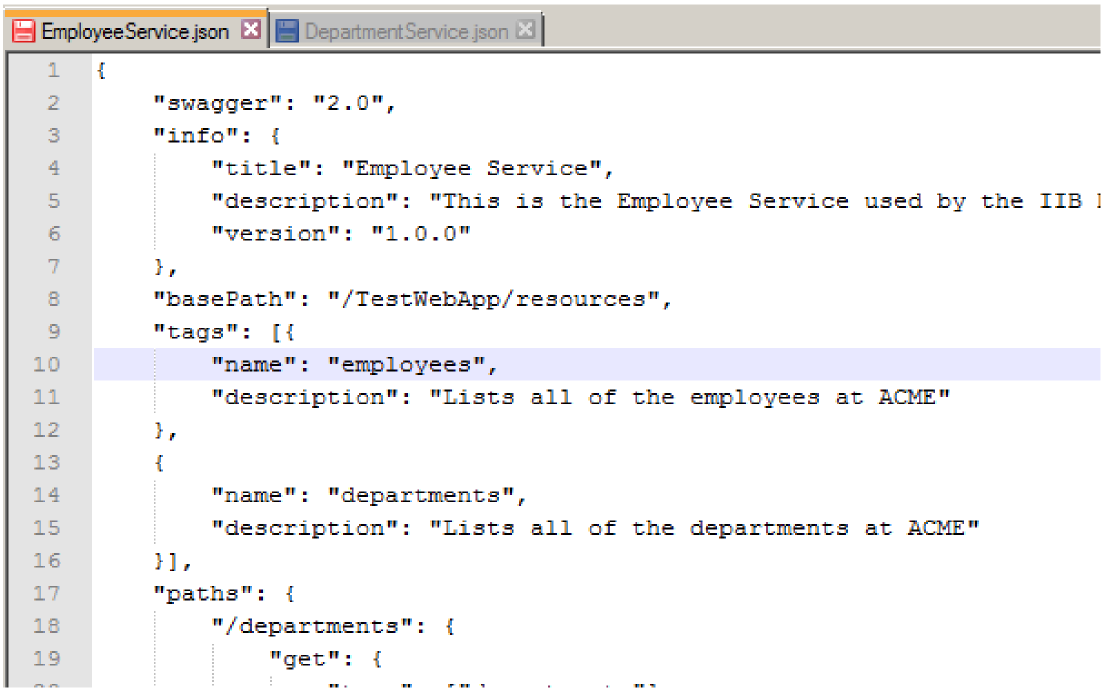
The main section of the document is a series of operations (GET, POST, PUT, etc), associated with different types of operation (getEmployee, listEmployees, etc).

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 EmployeeServiceInterface Shared Library

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

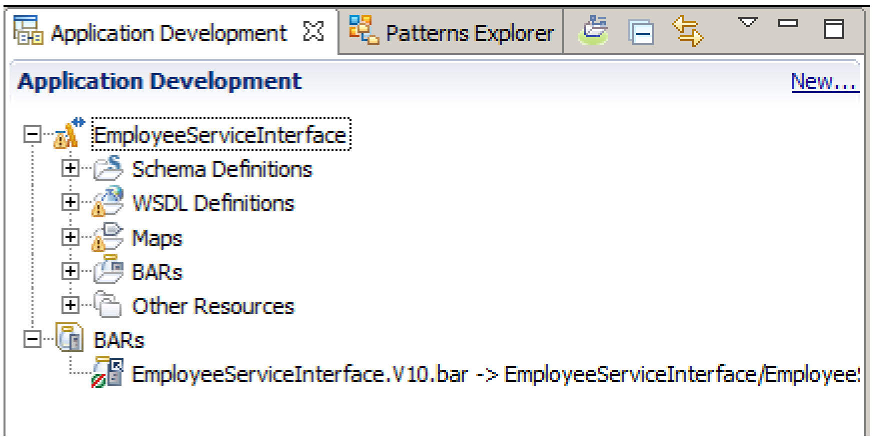
   Create a new workspace named c:\workspaces\IIB_REST (use File, Switch Workspace to show the dialogue to set the new workspace name).

   In the workspace, import the Project Interchange file:

   `C:\student10\integration_service\solution\EmployeeServiceInterface.V10.zip`

   Select both the **SAMPLE** and **EmployeeServiceInterface** projects to import.

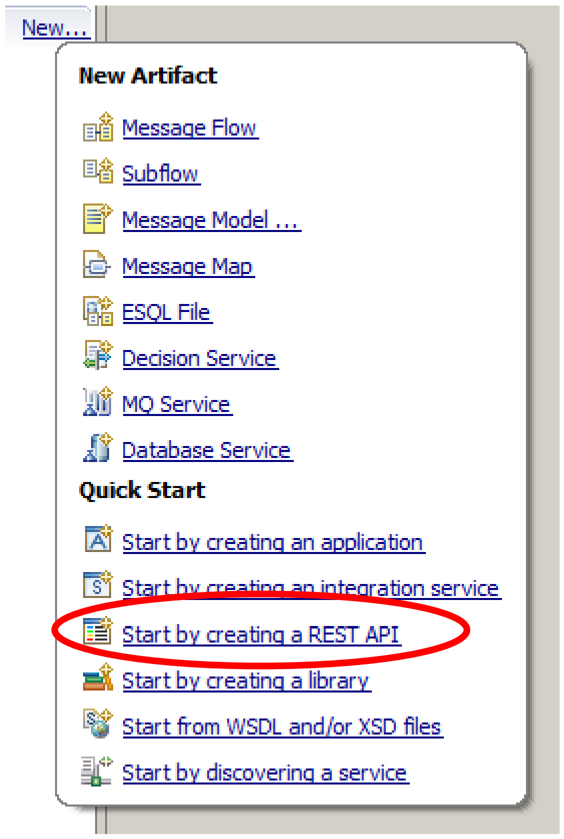
   When imported, you will see the Shared Library like this:



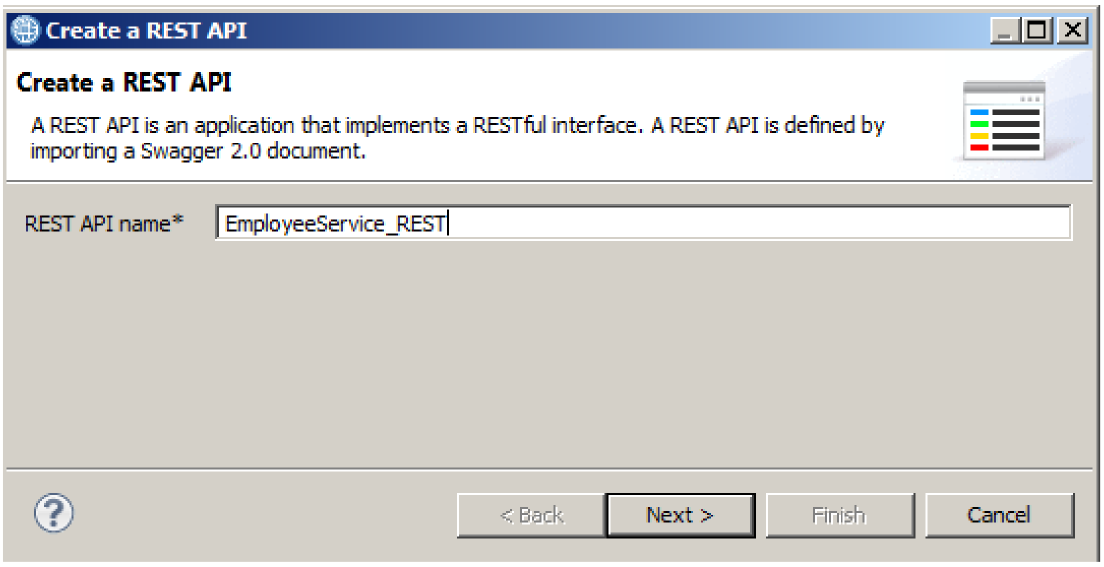
```
EmployeeServiceInterface
    Schema Definitions
    WSDL Definitions
    Maps
    BARs
    Other Resources
BARs
    EmployeeServiceInterface.V10.bar -> EmployeeServiceInterface/Employee!
```

2.3 Develop the new REST service

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


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

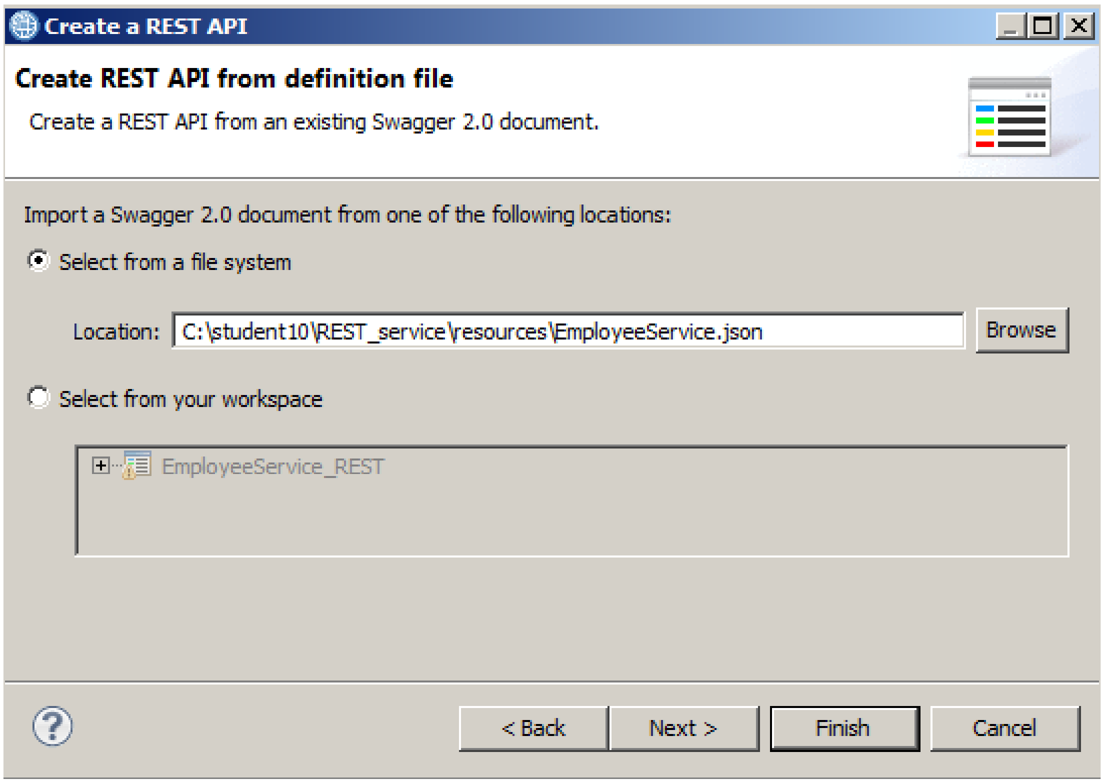
A REST API is an application that implements a RESTful interface. A REST API is defined by importing a Swagger 2.0 document.

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

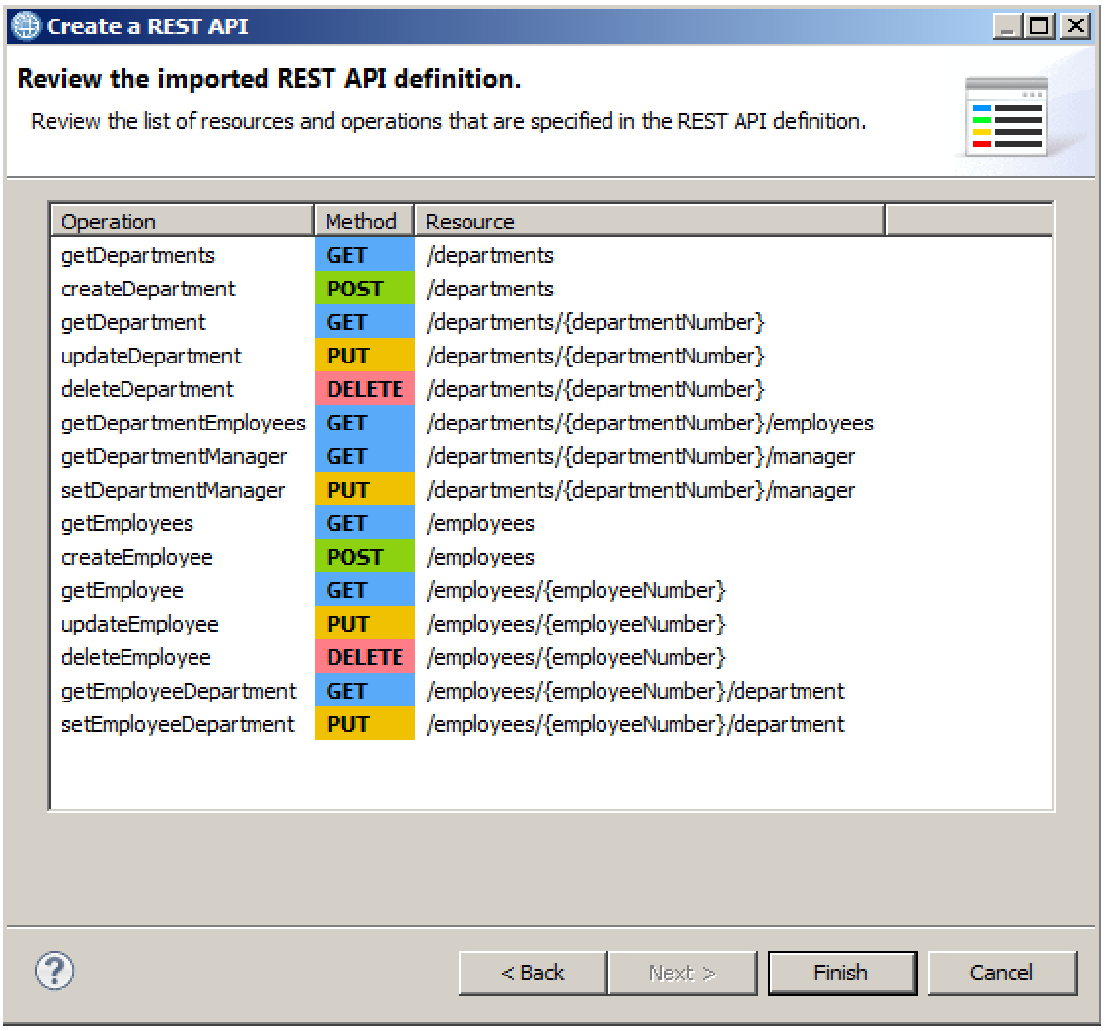
EmployeeService_REST

.

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 SAMPLE 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.



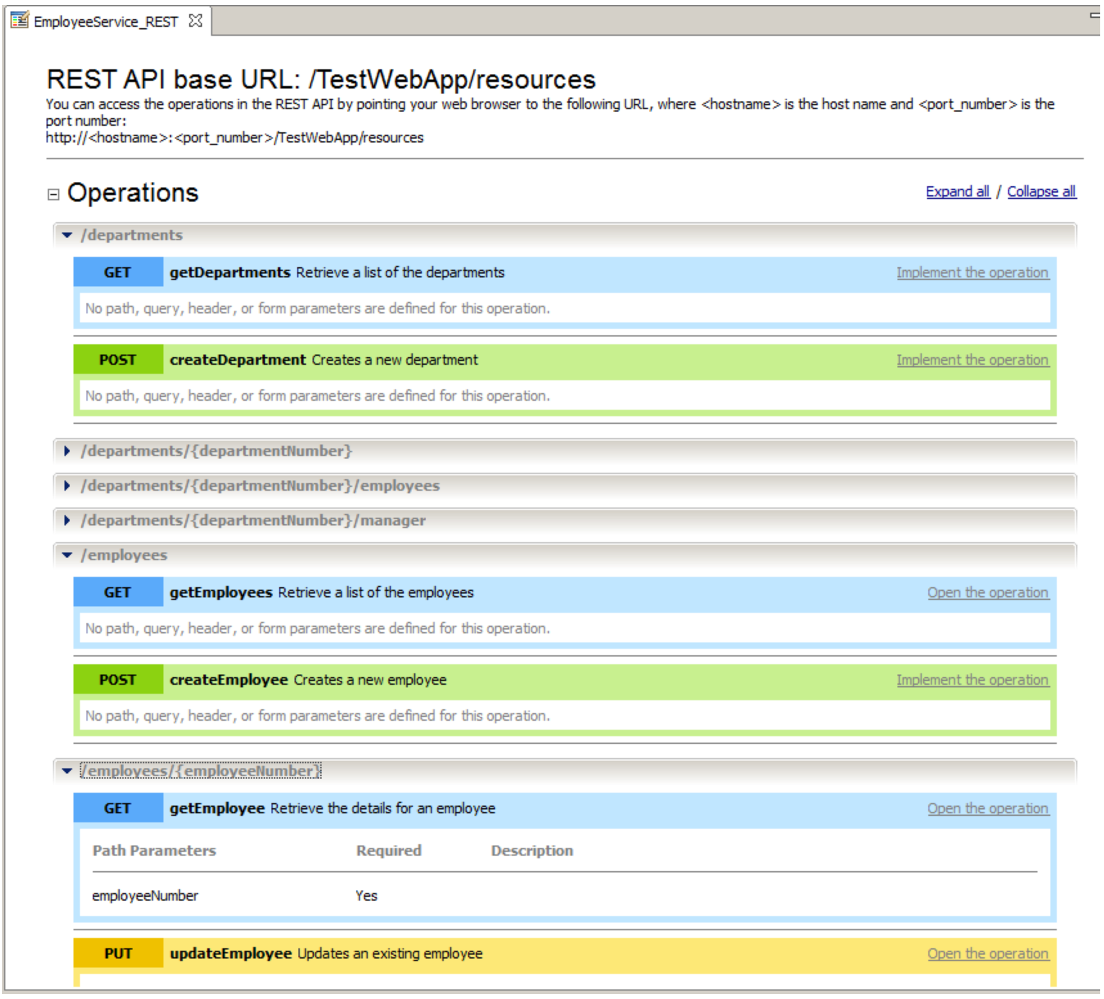
Review the imported REST API definition.

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 service.

   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}

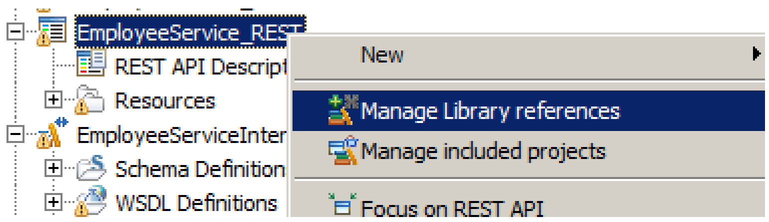
GET getEmployee Retrieve the details for an employee — Open the operation

Path Parameters	Required	Description
employeeNumber	Yes	

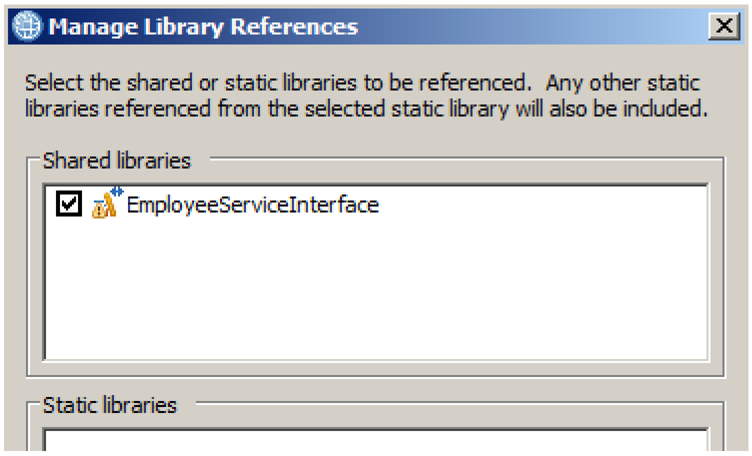
PUT updateEmployee Updates an existing employee — Open the operation

6. Before proceeding with the implementation, the REST service has to reference the required Shared Library. This is because it needs access to both the schema definitions and the IIB getEmp map that are stored in the library.

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



   Tick EmployeeServiceInterface 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

[✓] EmployeeServiceInterface

Static libraries

2.4 Implement the getEmployee operation

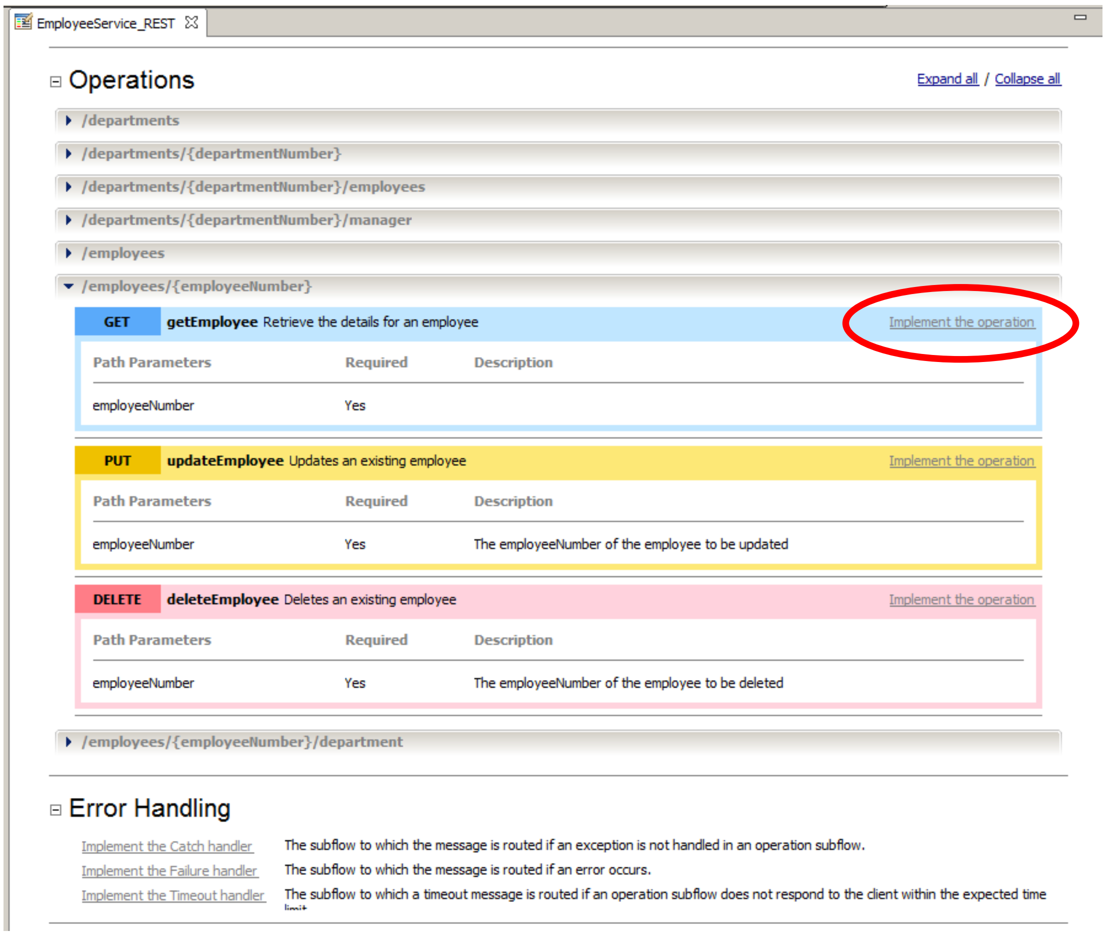
1. You will now implement the getEmployee operation.

   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.


EmployeeService_REST

Operations

Expand all / Collapse all

/departments

/departments/{departmentNumber}

/departments/{departmentNumber}/employees

/departments/{departmentNumber}/manager

/employees

/employees/{employeeNumber}

GET **getEmployee** Retrieve the details for an employee — 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

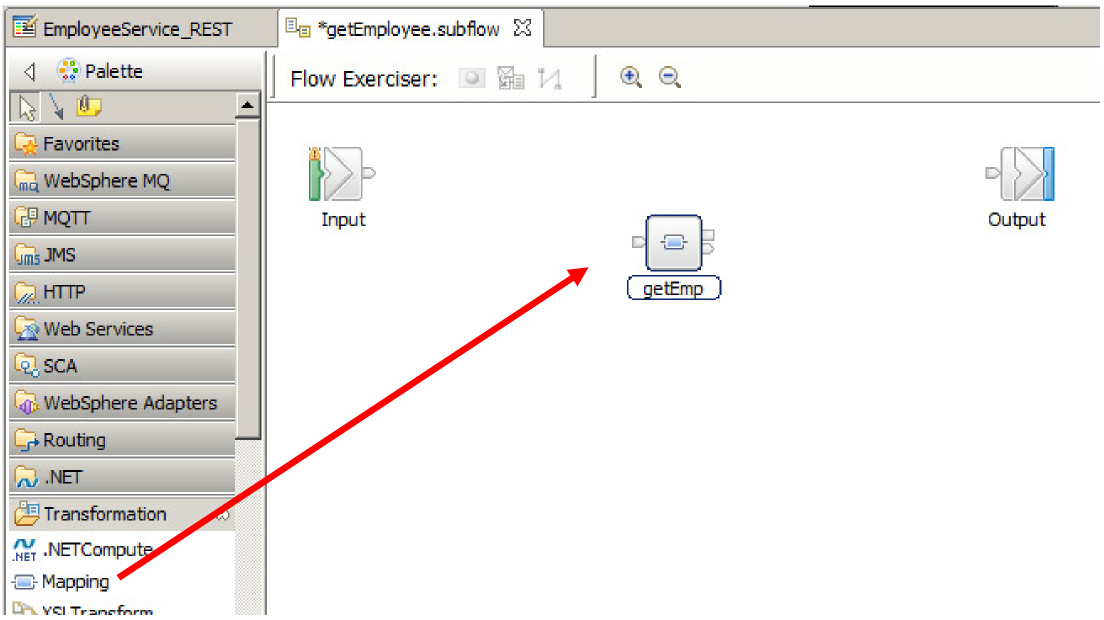
/employees/{employeeNumber}/department

Error Handling

Implement the Catch handler	The subflow to which the message is routed if an exception is not handled in an operation subflow.
Implement the Failure handler	The subflow to which the message is routed if an error occurs.
Implement the Timeout handler	The subflow to which a timeout message is routed if an operation subflow does not respond to the client within the expected time limit

2. Drop a Mapping node onto the flow editor.

   Name the new map "getEmp".

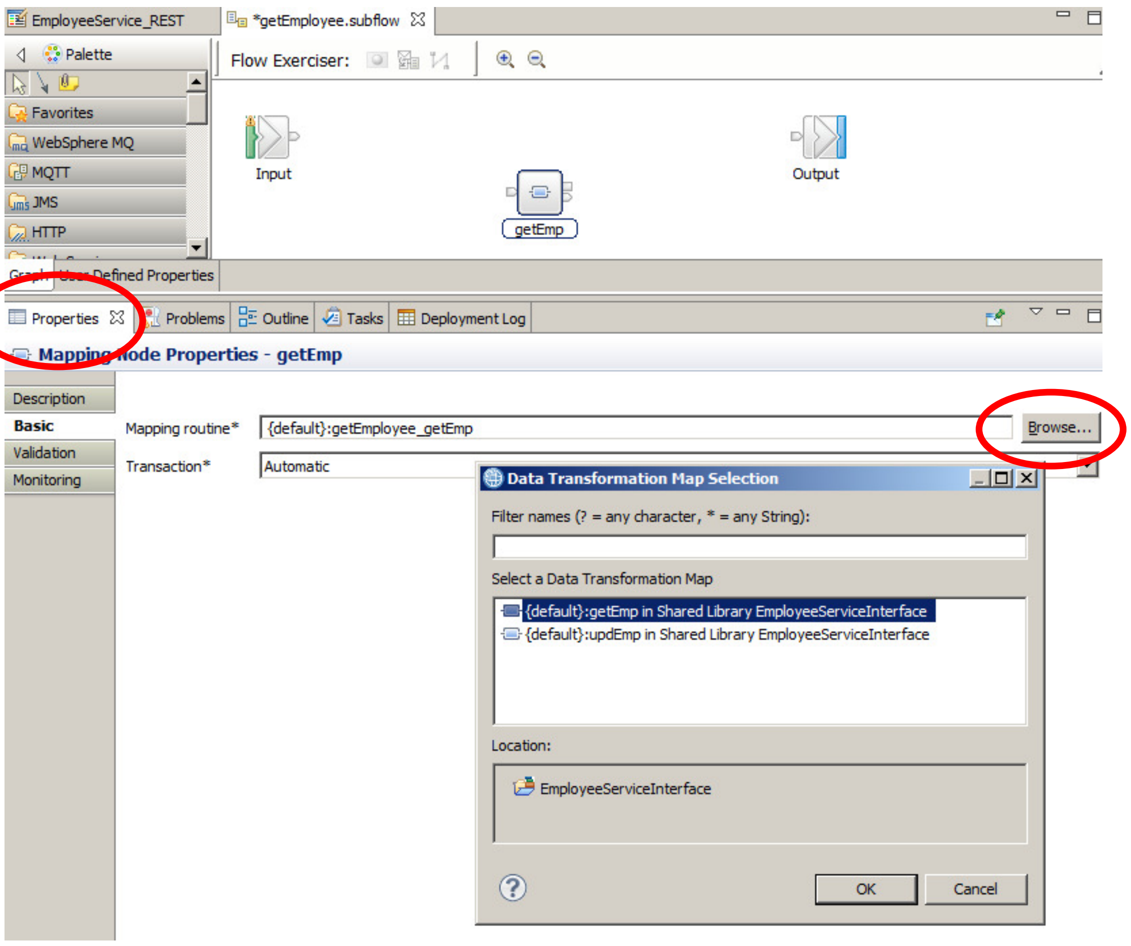


3. This mapping node will actually be the getEmp map that is already built and available in the EmployeeServiceInterface, so you must change the properties of the node to reference this map.

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

   If you are using the prebuilt version of the Shared Library, you will also see a map called updEmp.

   Click OK.



Mapping Node Properties - getEmp

Mapping routine* {default}:getEmployee_getEmp

Transaction* Automatic

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

Select a Data Transformation Map

{default}:getEmp in Shared Library EmployeeServiceInterface
{default}:updEmp in Shared Library EmployeeServiceInterface

Location:

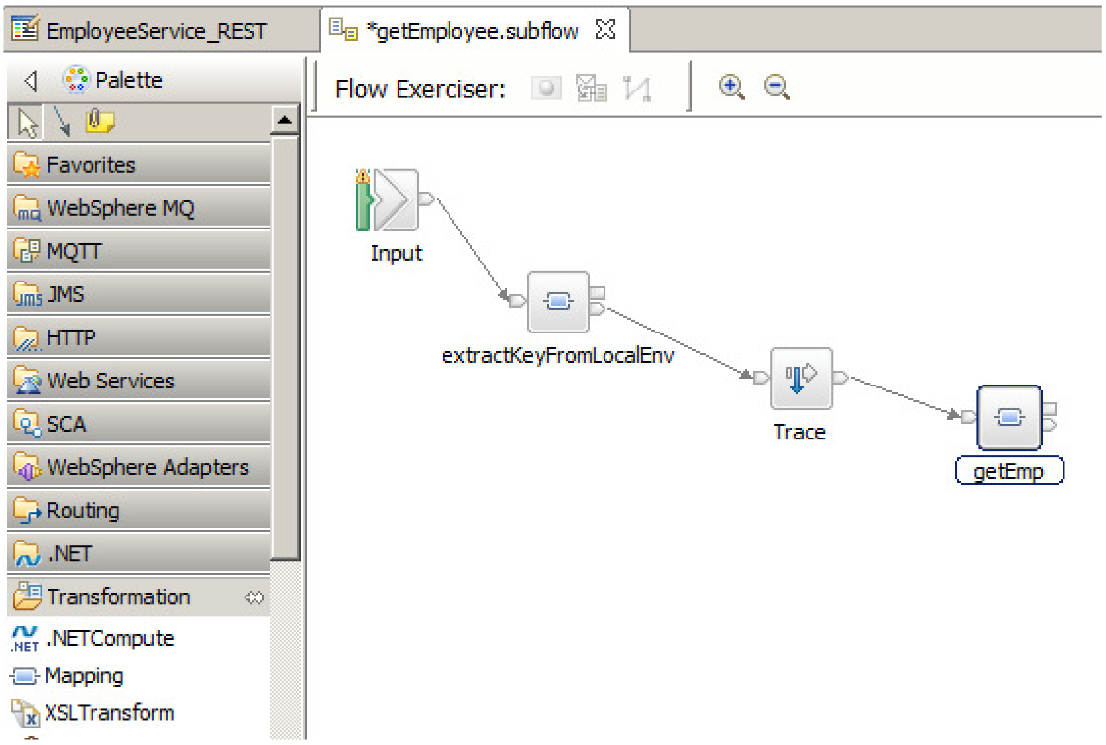
EmployeeServiceInterface

4. REST GET operations will be contained within the HTTP header part of the incoming message. When IIB receives such an incoming message, it extracts the REST parameters and places them into the Local Environment. IIB v10 has introduced a new part of the LocalEnv called REST.

   The getEmp map requires its input in the form defined by the EMPLOYEE schema. To construct the message in this format, you will develop a new mapping node to obtain the incoming parameter for the getEmployee operation. The mapping node will use this information to construct a message in the format required by getEmp.

   Drop two further nodes onto the flow and connect as shown:

   - Mapping node - name this extractKeyFromLocalEnv
   - Trace node (it's in the Construction folder)

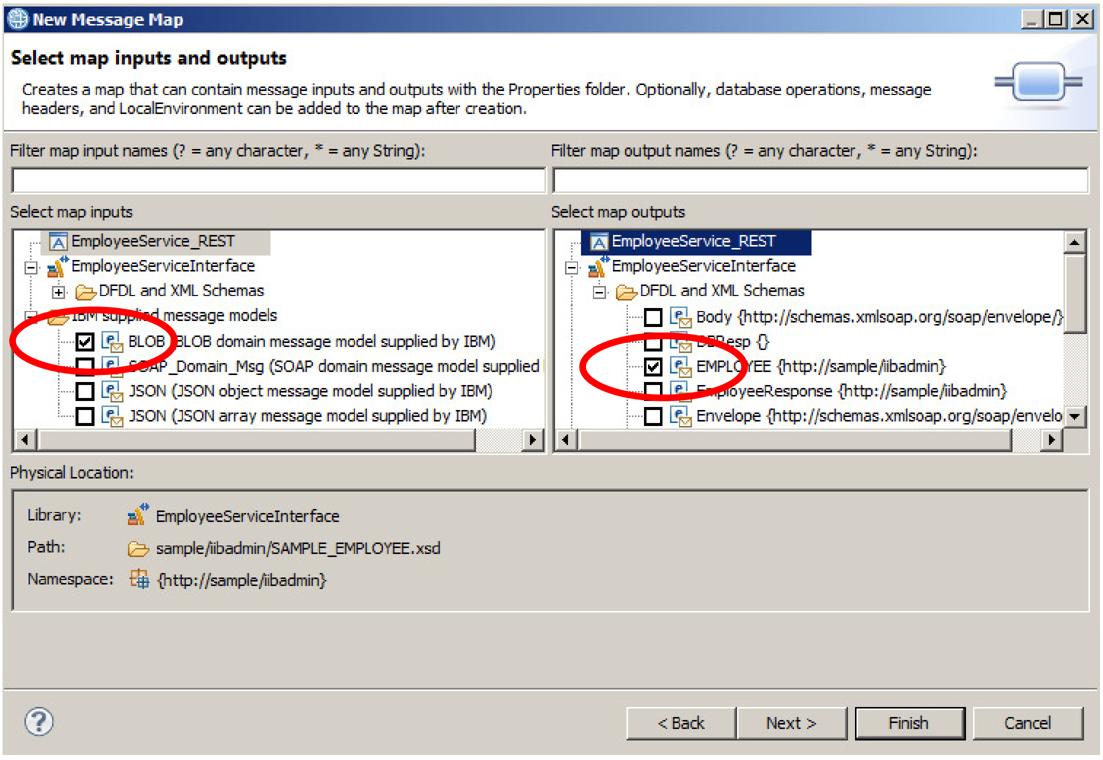


5. Open the extractKeyFromLocalEnv map, and click Next at the first window.

   For the map inputs and outputs, make the following selections:

   - Input
     - IBM supplied message models - BLOB (the subflow will obtain its information from the LocalEnvironment, and there is no need to parse the message payload)
   - Output
     - EmployeeServiceInterface, DFDL/XML schemas - EMPLOYEE

   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):

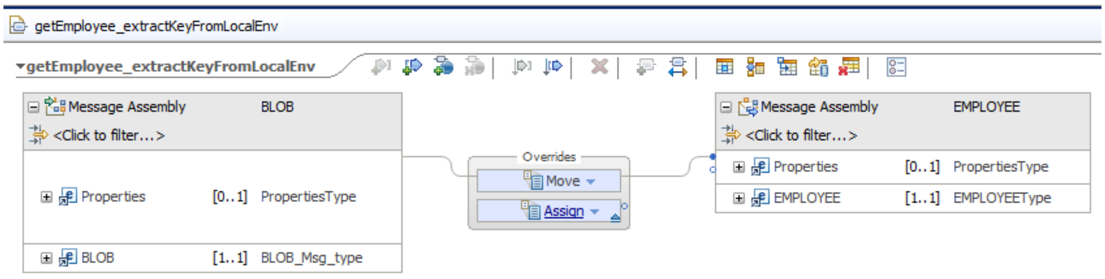
Select map inputs
- EmployeeService_REST
- EmployeeServiceInterface
  - DFDL and XML Schemas
  - IBM supplied message models
    - [✓] BLOB (BLOB domain message model supplied by IBM)
    - [ ] SOAP_Domain_Msg (SOAP domain message model supplied
    - [ ] JSON (JSON object message model supplied by IBM)
    - [ ] JSON (JSON array message model supplied by IBM)

Select map outputs
- EmployeeService_REST
- EmployeeServiceInterface
  - DFDL and XML Schemas
    - [ ] Body {http://schemas.xmlsoap.org/soap/envelope/}
    - [ ] DBResp {}
    - [✓] EMPLOYEE {http://sample/iibadmin}
    - [ ] EmployeeResponse {http://sample/iibadmin}
    - [ ] Envelope {http://schemas.xmlsoap.org/soap/envelo

Physical Location:

Library: EmployeeServiceInterface
Path: sample/iibadmin/SAMPLE_EMPLOYEE.xsd
Namespace: {http://sample/iibadmin}

6. The basic mappings will be shown.

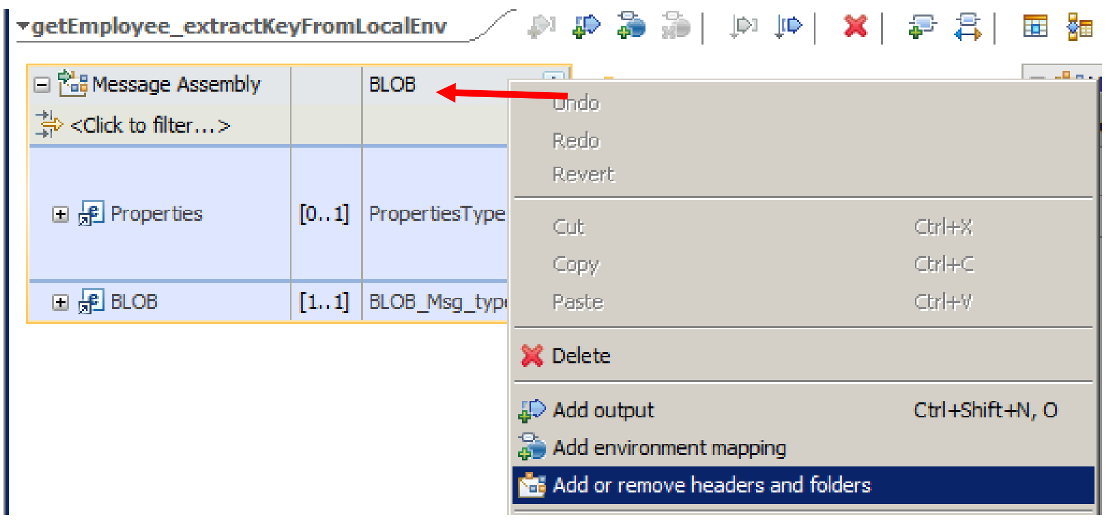

getEmployee_extractKeyFromLocalEnv

Message Assembly		BLOB
<Click to filter...>		
Properties	[0..1]	PropertiesType
BLOB	[1..1]	BLOB_Msg_type

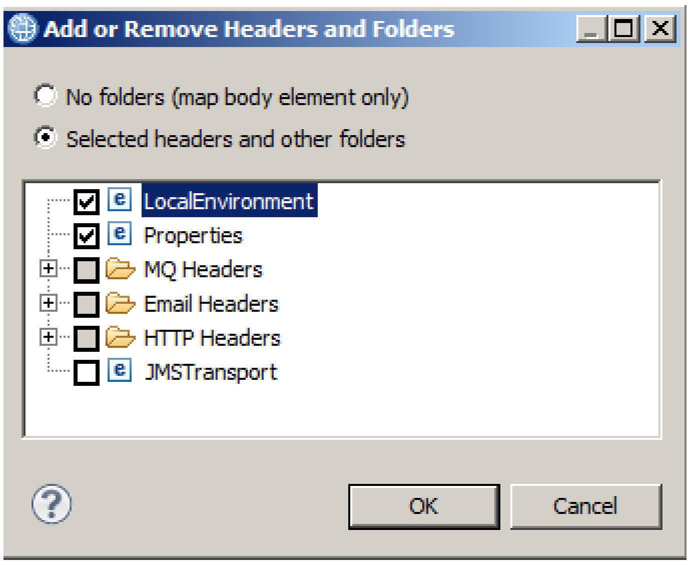
Message Assembly		EMPLOYEE
<Click to filter...>		
Properties	[0..1]	PropertiesType
EMPLOYEE	[1..1]	EMPLOYEEType

7. For a REST GET operation, the employee key 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".



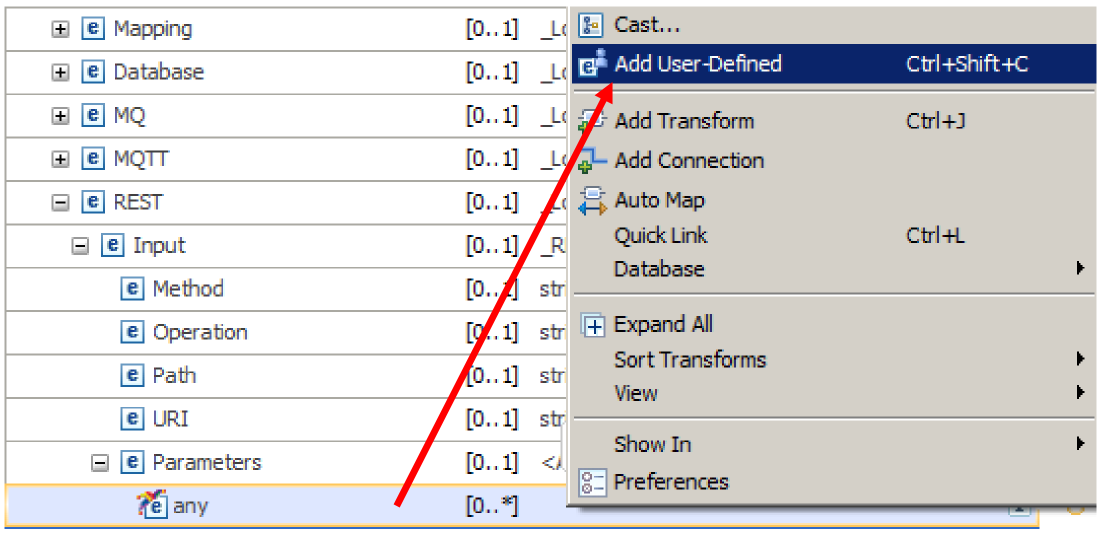
   Select the LocalEnvironment and click OK.


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

8. 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.

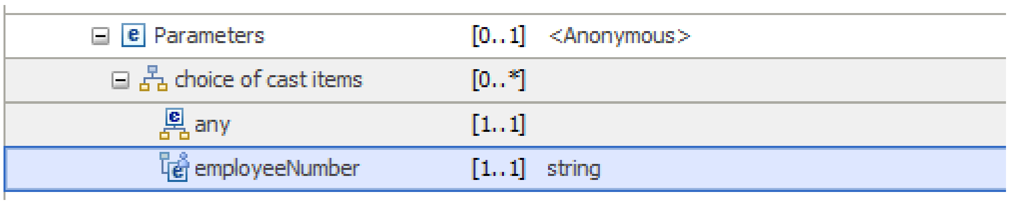
   Right-click the "any" element and select "Add User-Defined".



```
Mapping      [0..1]  _L
Database     [0..1]  _L
MQ           [0..1]  _L
MQTT         [0..1]  _L
REST         [0..1]  _L
  Input      [0..1]  _R
    Method     [0..1]  str
    Operation  [0..1]  str
    Path       [0..1]  str
    URI        [0..1]  str
    Parameters [0..1]  <A
      any      [0..*]
```

9. **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.**

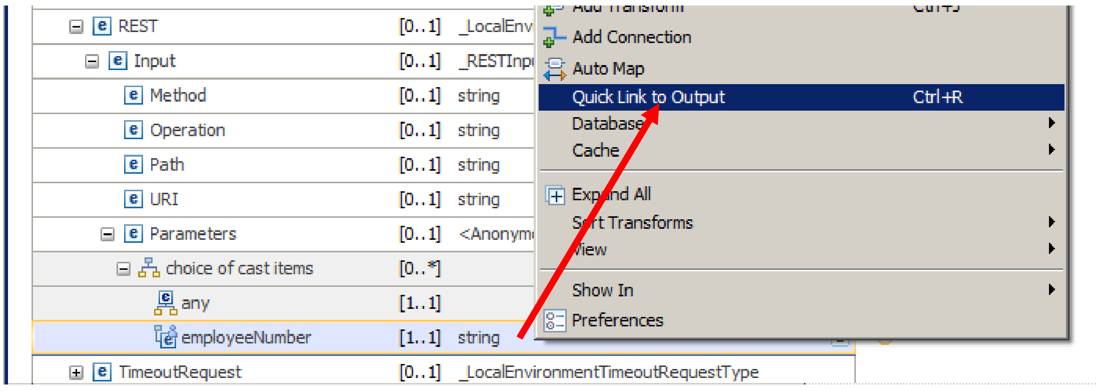
10. Name the new element employeeNumber. Note, this has to match the precise name and case of the JSON employeeNumber element that you saw in the JSON document.



```
Parameters                [0..1]  <Anonymous>
  choice of cast items    [0..*]
    any                   [1..1]
    employeeNumber        [1..1]  string
```

11. The new element needs to be mapped to the output EMPLOYEE/EMPNO message.

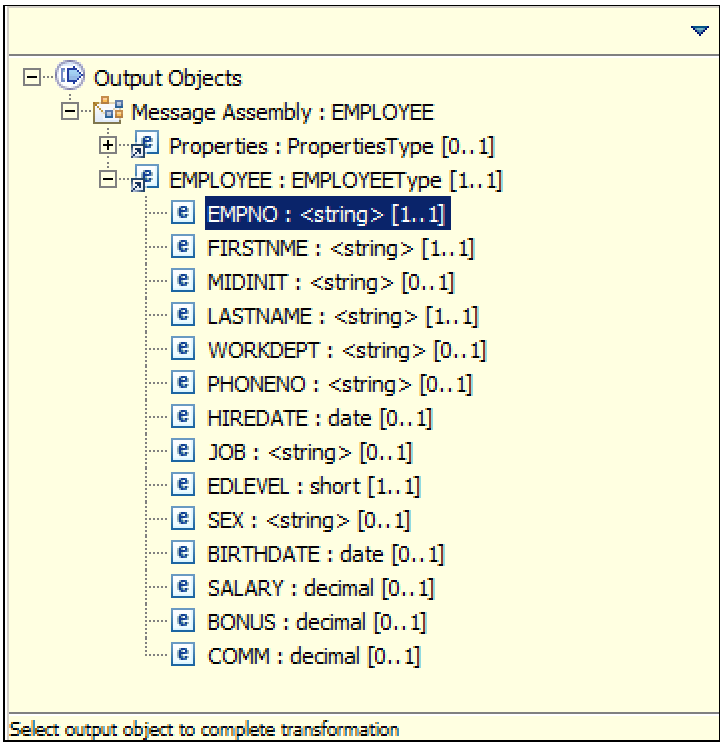
    Unfortunately, expanding the Local Environment has probably meant that the output message has disappeared from the map display. To handle this, perform a "Quick Link". Right-click employeeNumber, and select "Quick Link to Output".



```
REST                        [0..1]  _LocalEnv
  Input                     [0..1]  _RESTInp
    Method                  [0..1]  string
    Operation               [0..1]  string
    Path                    [0..1]  string
    URI                     [0..1]  string
    Parameters              [0..1]  <Anonym
      choice of cast items  [0..*]
        any                 [1..1]
        employeeNumber      [1..1]  string
TimeoutRequest              [0..1]  _LocalEnvironmentTimeoutRequestType
```

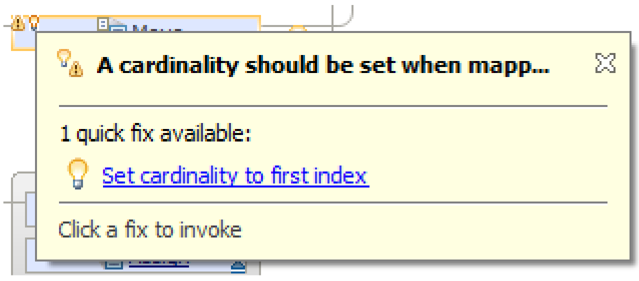
12. A pop-up window will show the available output message.

    Collapse the Properties element, and in the EMPLOYEE element, select EMPNO.



Output Objects
Message Assembly : EMPLOYEE
Properties : PropertiesType [0..1]
EMPLOYEE : EMPLOYEEType [1..1]
EMPNO : <string> [1..1]
FIRSTNME : <string> [1..1]
MIDINIT : <string> [0..1]
LASTNAME : <string> [1..1]
WORKDEPT : <string> [0..1]
PHONENO : <string> [0..1]
HIREDATE : date [0..1]
JOB : <string> [0..1]
EDLEVEL : short [1..1]
SEX : <string> [0..1]
BIRTHDATE : date [0..1]
SALARY : decimal [0..1]
BONUS : decimal [0..1]
COMM : decimal [0..1]
Select output object to complete transformation

13. A Move transform will be generated. If you wish, you can use the QuickFix to correct the cardinality warning message (hover over the small light on the Move transform, then select "Set cardinality to first index").

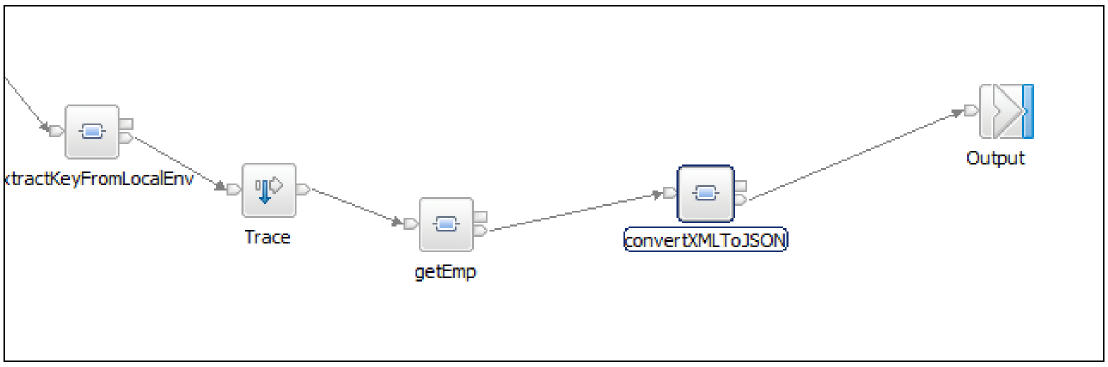


A cardinality should be set when mapp...
1 quick fix available:
Set cardinality to first index
Click a fix to invoke

    Save and close the map.

14. One final node is required. In the flow editor, drop a third map node onto the flow and name it convertXMLToJSON. Connect as shown.

    The map is required because the output needs to be sent back to the originating REST request in JSON format. (The getEmp map produces an output message in XML format).

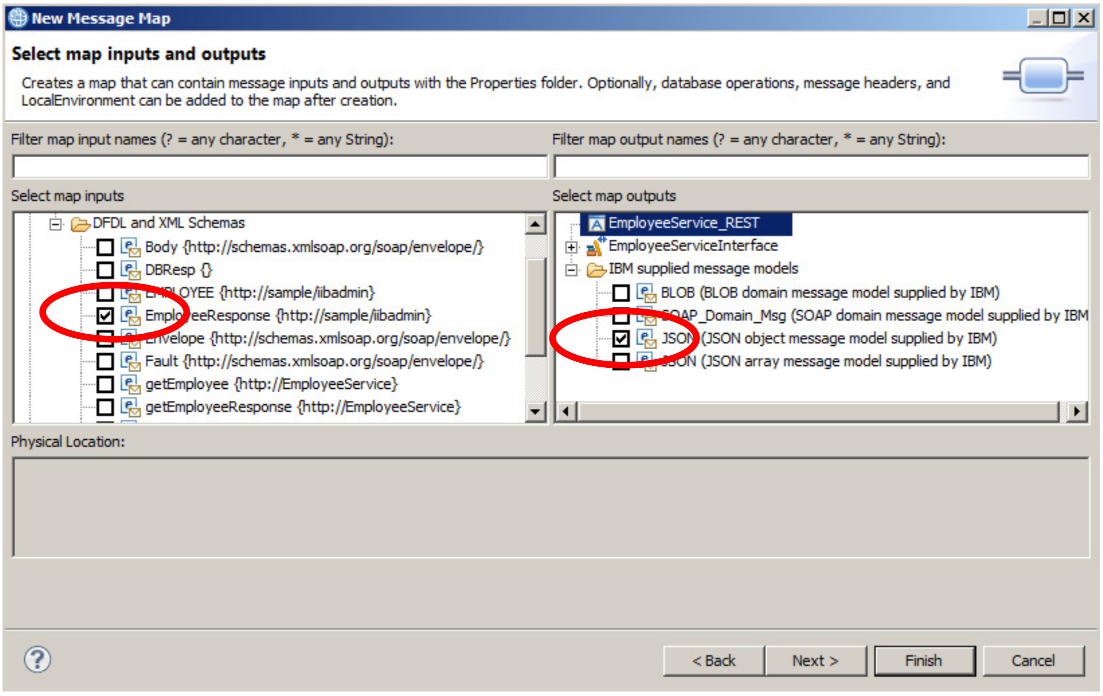


15. Open the new map.

    Click Next at the first window, and for the inputs and outputs, make the following selections:

    - Input
        - EmployeeResponse
    - 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):

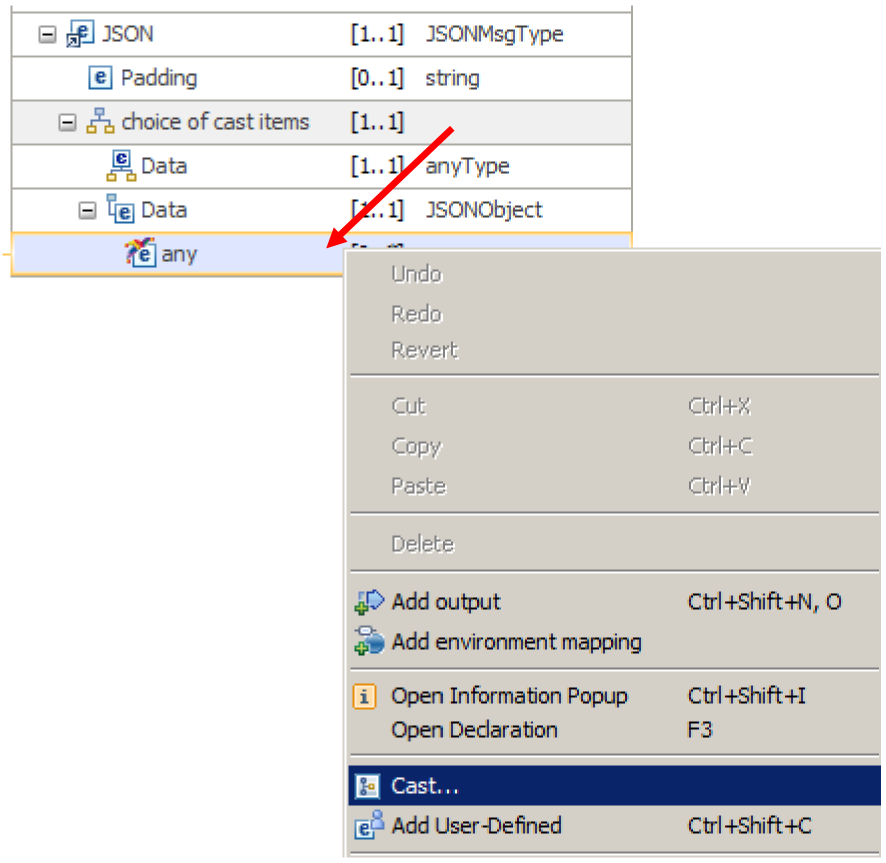
Select map inputs
- DFDL and XML Schemas
  - ☐ Body {http://schemas.xmlsoap.org/soap/envelope/}
  - ☐ DBResp {}
  - ☐ EMPLOYEE {http://sample/iibadmin}
  - ☑ EmployeeResponse {http://sample/iibadmin}
  - ☐ Envelope {http://schemas.xmlsoap.org/soap/envelope/}
  - ☐ Fault {http://schemas.xmlsoap.org/soap/envelope/}
  - ☐ getEmployee {http://EmployeeService}
  - ☐ getEmployeeResponse {http://EmployeeService}

Select map outputs
- EmployeeService_REST
- EmployeeServiceInterface
- IBM supplied message models
  - ☐ BLOB (BLOB domain message model supplied by IBM)
  - ☐ SOAP_Domain_Msg (SOAP domain message model supplied by IBM
  - ☑ JSON (JSON object message model supplied by IBM)
  - ☐ JSON (JSON array message model supplied by IBM)

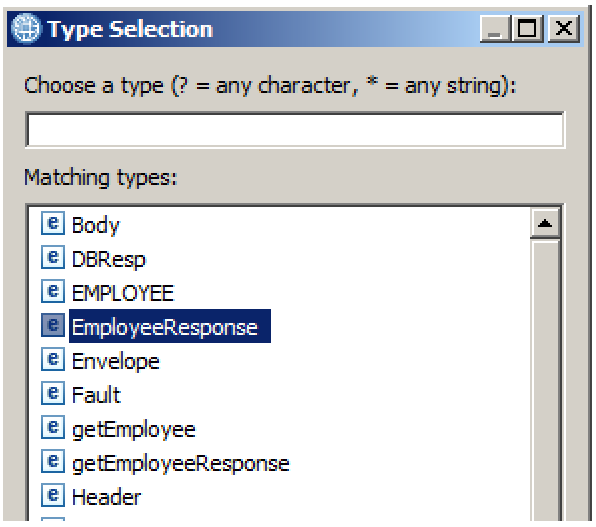
Physical Location:

16. The output JSON message needs to be Cast as EmployeeResponse. Fully expand the JSON output, and right-click the final "any".

    Select Cast.



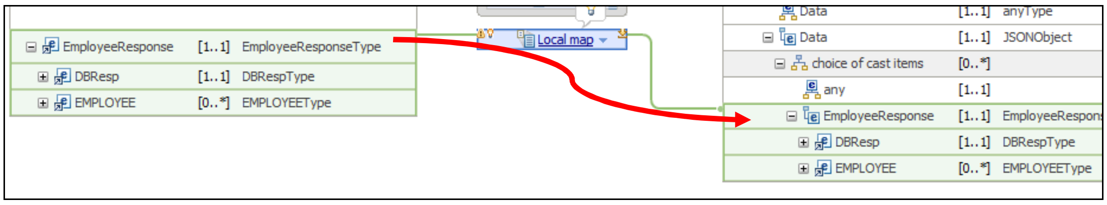
17. At the Type Selection, choose EmployeeResponse.



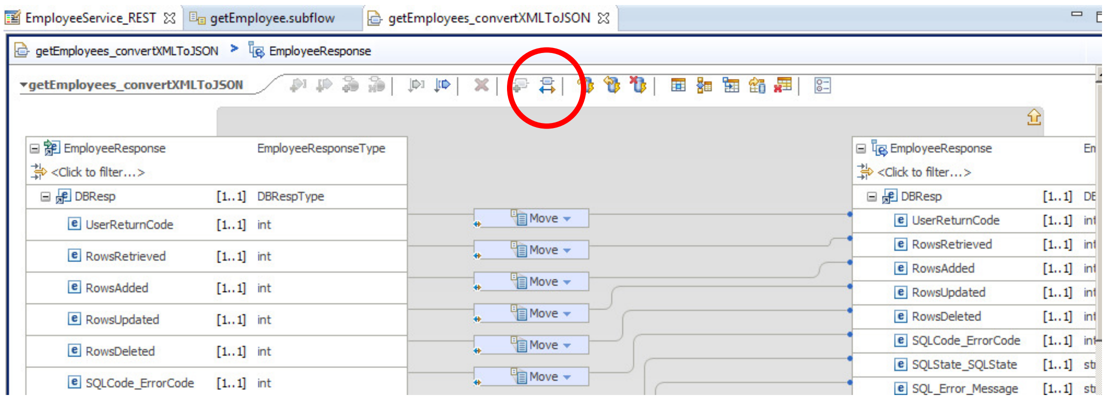
Click OK.

18. Map EmployeeResponse to EmployeeResponse.

    This will create a "Local Map" transform.


EmployeeResponse [1..1] EmployeeResponseType
DBResp [1..1] DBRespType
EMPLOYEE [0..*] EMPLOYEEType
Local map
Data [1..1] anyType
Data [1..1] JSONObject
choice of cast items [0..*]
any [1..1]
EmployeeResponse [1..1] EmployeeRespon
DBResp [1..1] DBRespType
EMPLOYEE [0..*] EMPLOYEEType

19. Click "Local Map" to create the required element mappings. Do this by using the Automap feature, and accept all the automatically generated mappings.



20. Save and close the map.

    Save the subflow.

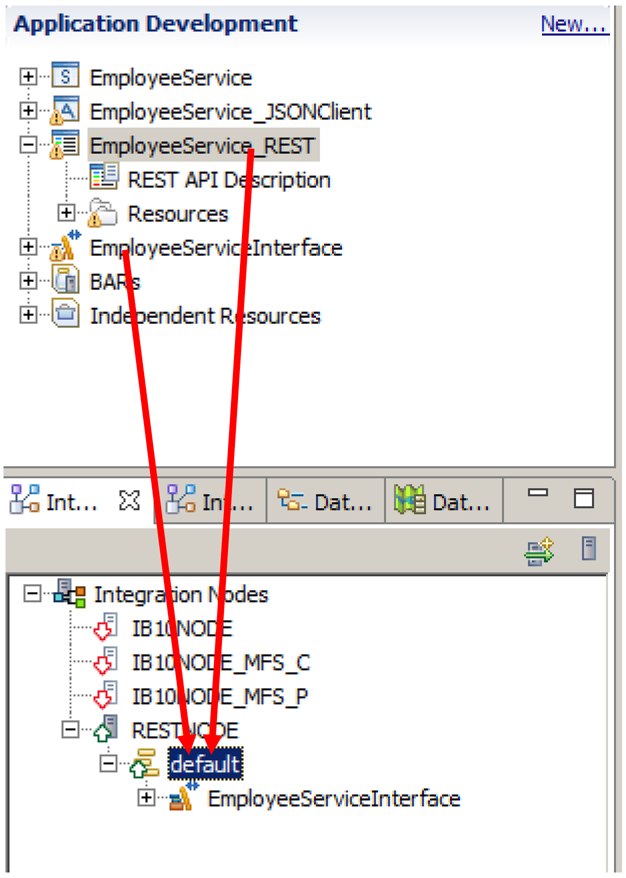
3. Test the EmployeeService REST API

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

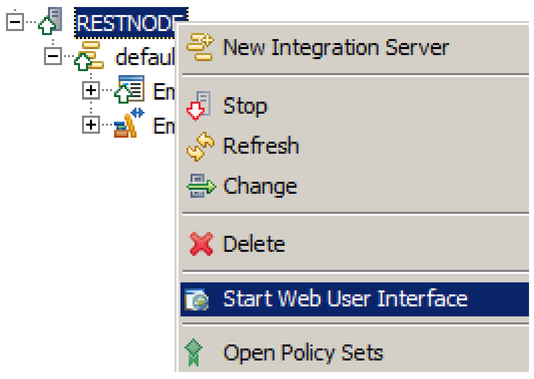
3.1 Deploy the service

1. In the navigator, deploy the EmployeeServiceInterface shared library to RESTNODE/default.

   Then deploy EmployeeService_REST to RESTNODE/default (drag/drop or right-click, deploy).



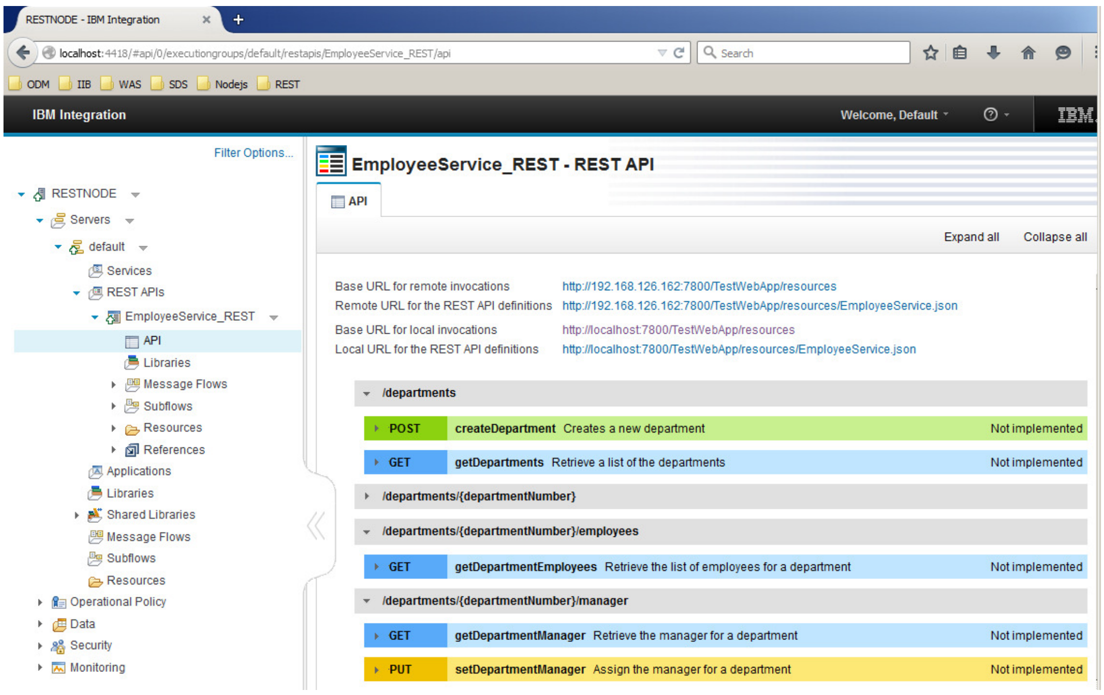
2. Open the IIB web UI by right-clicking RESTNODE and selecting Start Web User Interface.



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

   Under EmployeeService_REST, click "API", which will show you the available operations in IIB 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).

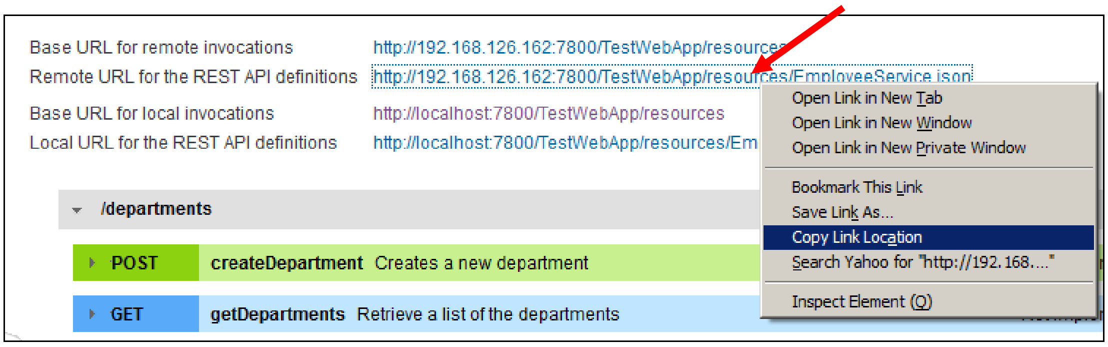


EmployeeService_REST - REST API

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/EmployeeService.json

/departments
POST createDepartment Creates a new department Not implemented
GET getDepartments Retrieve a list of the departments Not implemented
/departments/{departmentNumber}
/departments/{departmentNumber}/employees
GET getDepartmentEmployees Retrieve the list of employees for a department Not implemented
/departments/{departmentNumber}/manager
GET getDepartmentManager Retrieve the manager for a department Not implemented
PUT setDepartmentManager Assign the manager for a department Not implemented

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

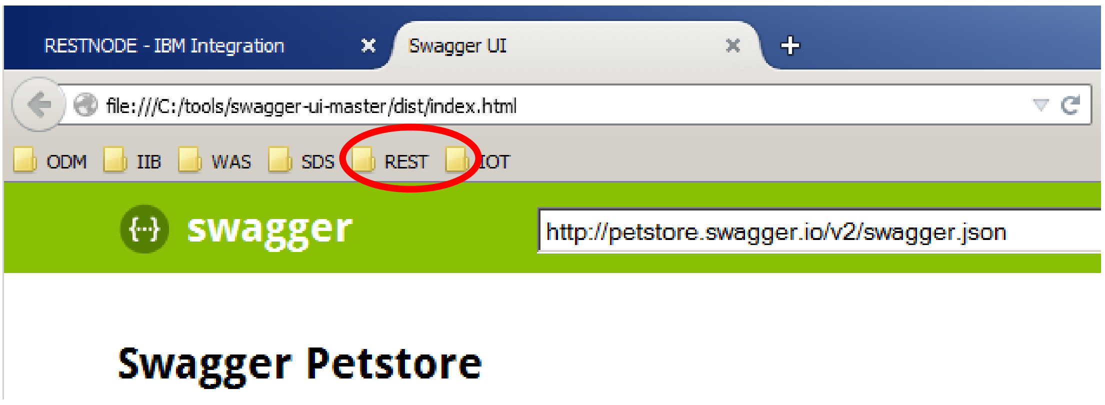


Base URL for remote invocations http://192.168.126.162:7800/TestWebApp/resources
Remote URL for the REST API definitions http://192.168.126.162:7800/TestWebApp/resources/EmployeeService.json
Base URL for local invocations http://localhost:7800/TestWebApp/resources
Local URL for the REST API definitions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

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

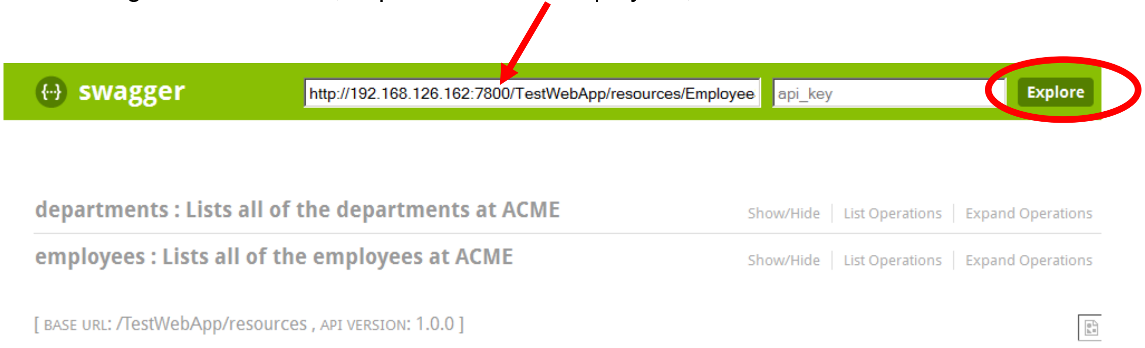
5. In Firefox, open a new tab, and open the SwaggerUI tool (using the bookmark in the REST folder).

   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

6. In the entry field, paste the contents of the clipboard and click Explore.

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



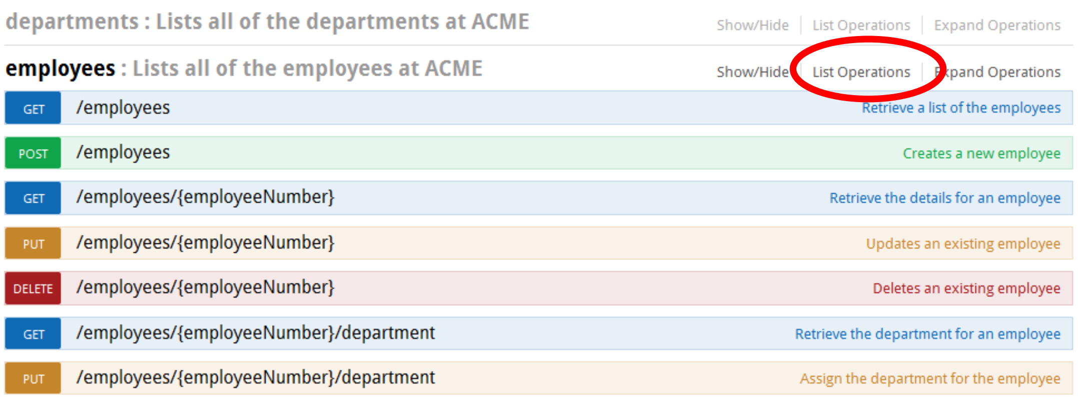
departments : Lists all of the departments at ACME — Show/Hide | List Operations | Expand Operations

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

[BASE URL: /TestWebApp/resources , API VERSION: 1.0.0]

7. 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.



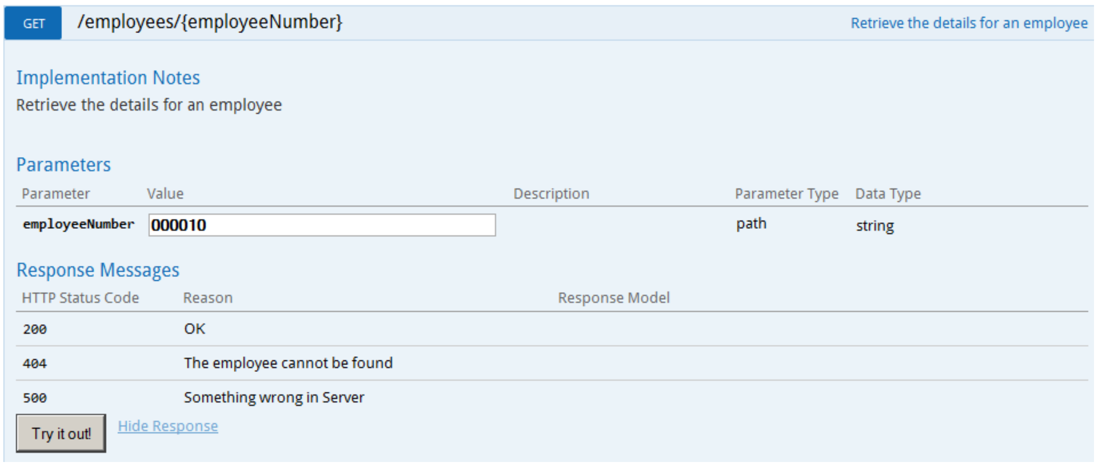
departments : Lists all of the departments at ACME

employees : Lists all of the employees at ACME

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

8. 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

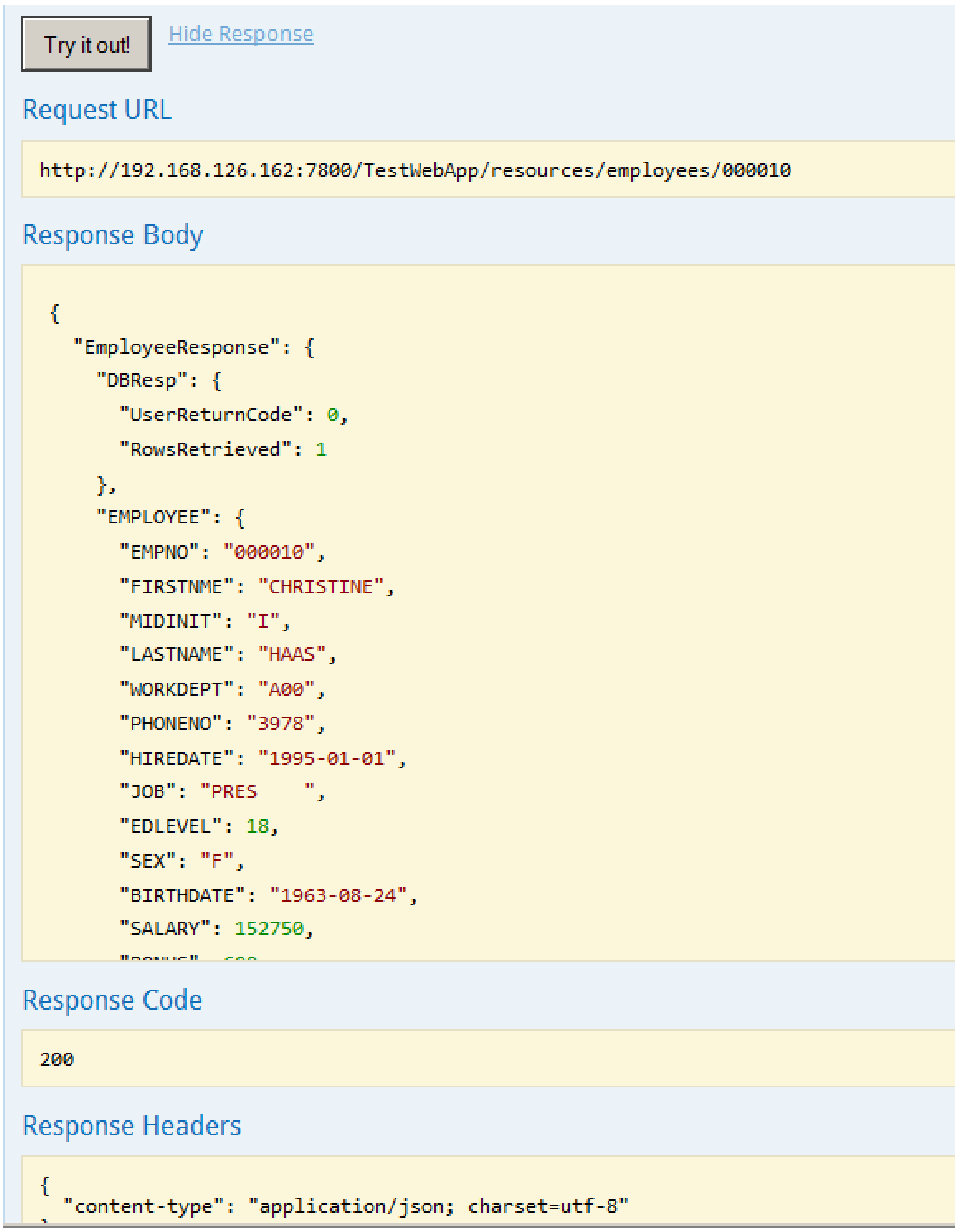
Parameter	Value	Description	Parameter Type	Data Type
employeeNumber	000010		path	string

Response Messages

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

9. 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

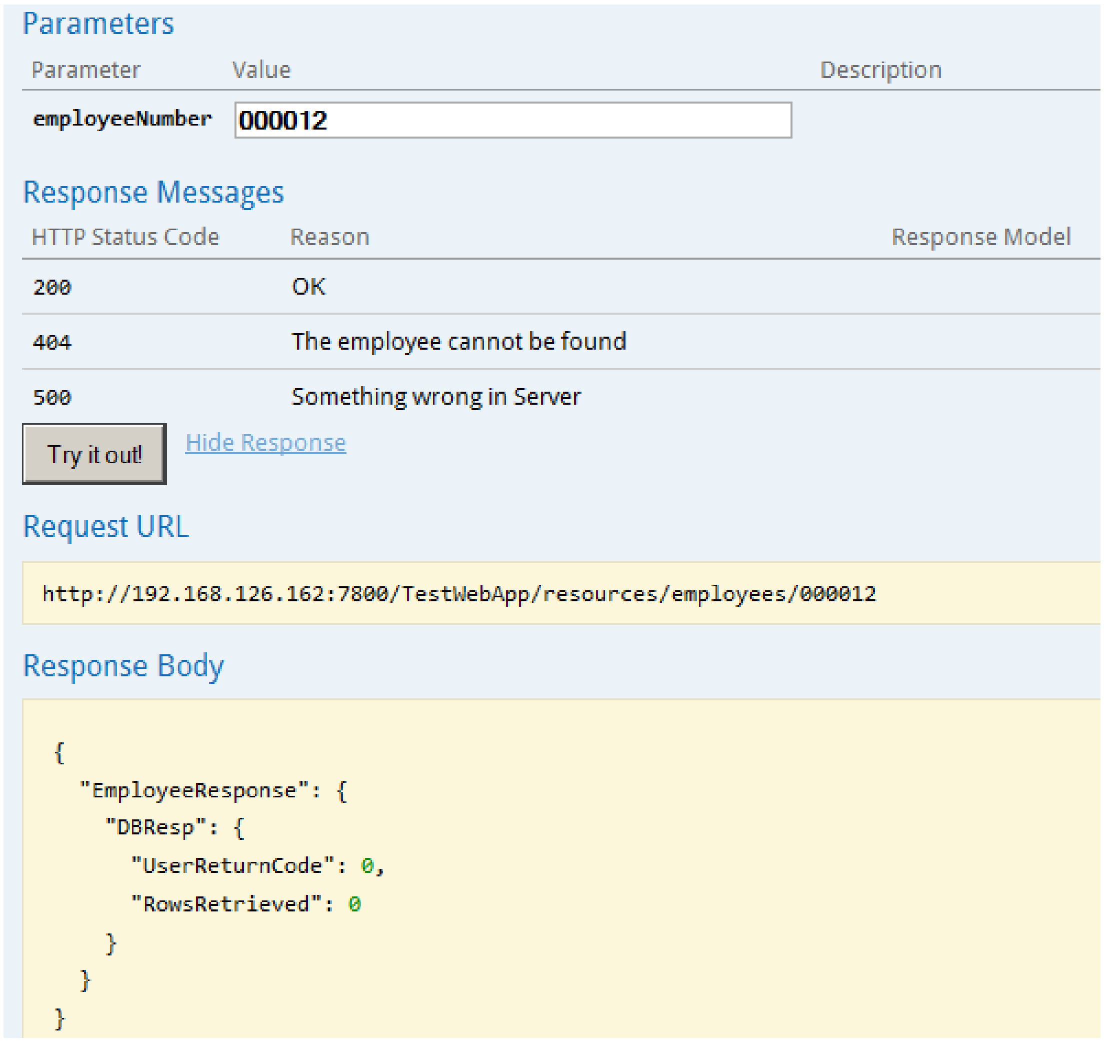
```
200
```

Response Headers

```
{
  "content-type": "application/json; charset=utf-8"
```

10. 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
employeeNumber	000012	

Response Messages

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

Request URL

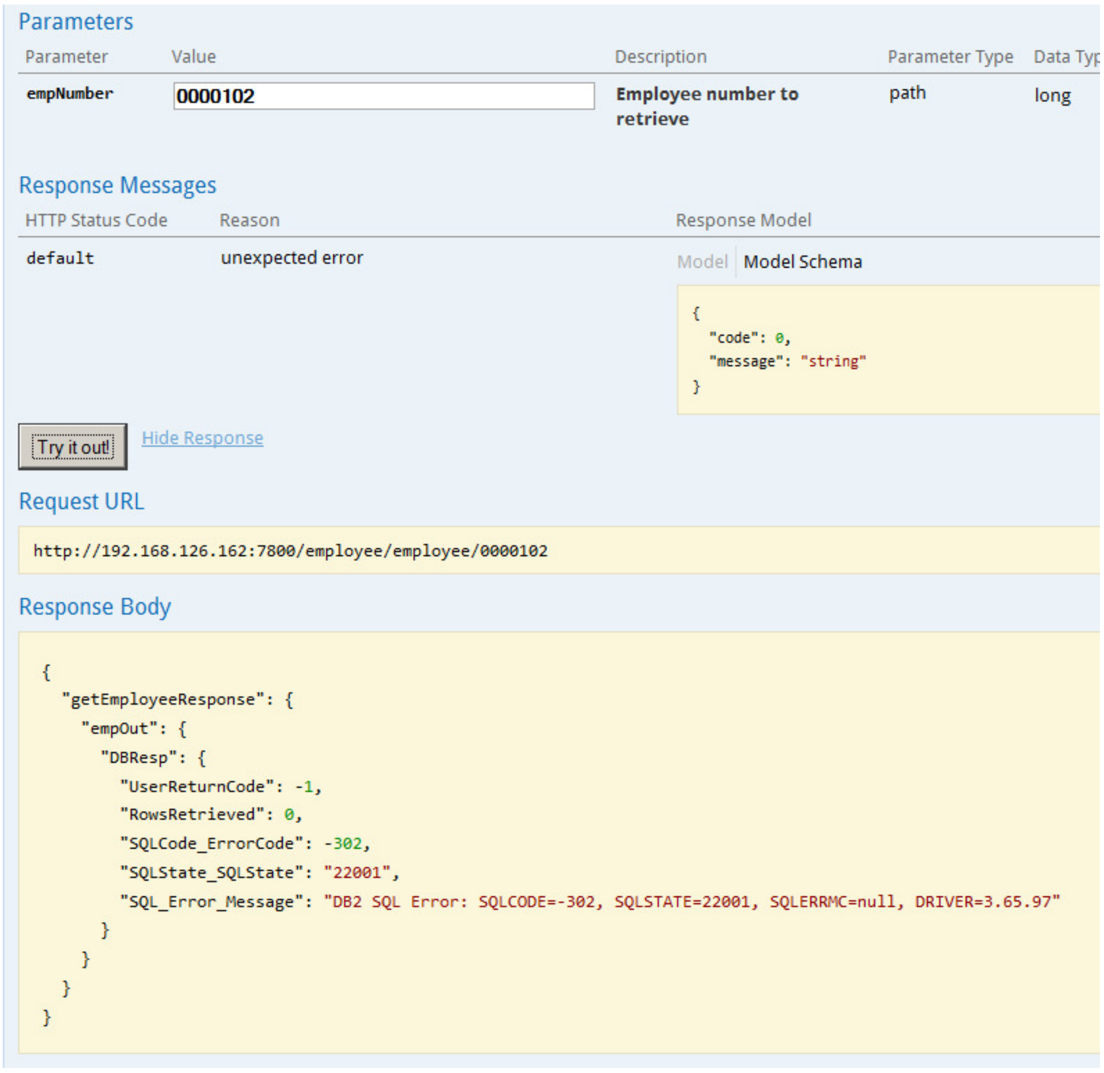
```
http://192.168.126.162:7800/TestWebApp/resources/employees/000012
```

Response Body

```
{
  "EmployeeResponse": {
    "DBResp": {
      "UserReturnCode": 0,
      "RowsRetrieved": 0
    }
  }
}
```

11. Now use an invalid length key, for example 0000102, which has 7 characters.

    You will see that although the request completes, the UserReturnCode = -1. This indicates that a database error has occurred, and the relevant SQL errors have been passed back to the REST client. In this case, the SQLCODE of -302 indicates that the key field is too long



Parameters

Parameter	Value	Description	Parameter Type	Data Typ
empNumber	0000102	Employee number to retrieve	path	long

Response Messages

HTTP Status Code	Reason	Response Model
default	unexpected error	Model Model Schema

```
{
  "code": 0,
  "message": "string"
}
```

Request URL

```
http://192.168.126.162:7800/employee/employee/0000102
```

Response Body

```
{
  "getEmployeeResponse": {
    "empOut": {
      "DBResp": {
        "UserReturnCode": -1,
        "RowsRetrieved": 0,
        "SQLCode_ErrorCode": -302,
        "SQLState_SQLState": "22001",
        "SQL_Error_Message": "DB2 SQL Error: SQLCODE=-302, SQLSTATE=22001, SQLERRMC=null, DRIVER=3.65.97"
      }
    }
  }
}
```

END OF LAB GUIDE