



IBM Integration Bus

Using the Loopback Connector to access MongoDB

Featuring:

MongoDB
Setting the Loopback Request "filterString"
The Loopback Request node

September 2016

Hands-on lab built at product
Version 10.0.0.6

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.....	4
2. CREATE THE HRDB DATABASE IN MONGODB.....	5
2.1 START THE MONGODB SERVER AND CLIENT SHELL.....	5
2.2 CREATE THE HRDB DATABASE.....	7
3. CONFIGURE THE IIB INSTALLATION.....	9
3.1 INSTALL THE LOOPBACK CONNECTOR.....	9
3.2 CONFIGURE DATA SOURCE FOR LOOPBACK CONNECTOR.....	10
4. CREATE AN IIB REST API TO ACCESS MONGODB.....	11
4.1 CREATE THE NEW REST API	11
4.2 ALTERNATIVE 1: SET LOCALENVIRONMENT "FILTERSTRING" USING MAPPING NODE	17
4.3 ALTERNATIVE 2: SET LOCALENVIRONMENT "FILTERSTRING" USING ESQL COMPUTE NODE.....	24
4.4 COMPLETE THE FLOW DEVELOPMENT	26
4.6 DEPLOY AND TEST THE REST API.....	28
END OF LAB GUIDE	29

1. Introduction and Preparation

1.1 Introduction

IIB v10.0.0.6 has introduced new functionality from StrongLoop. This function is exposed as a Loopback Connector. The first implementation of this is the Loopback Request node, and this can be used to access databases such as MongoDB.

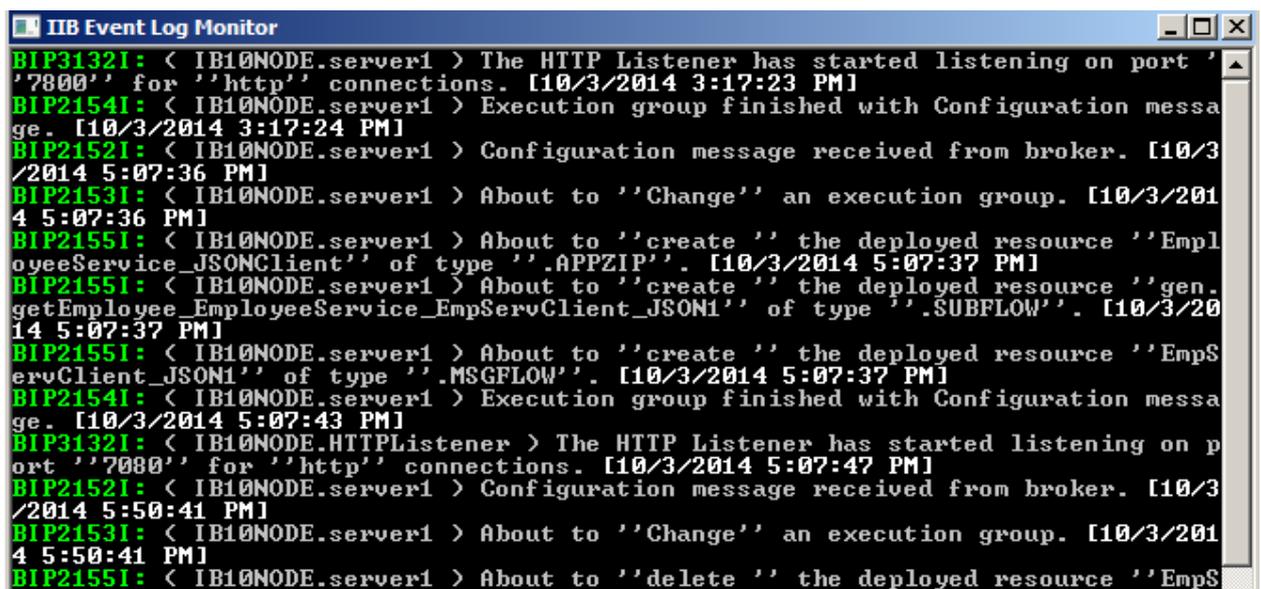
This lab provides a basic example of retrieving information from a MongoDB database. It is based on the previously built REST API, and will retrieve Employee documents from the HRDB database (EMPLOYEE container), located within a MongoDB instance.

IIB v10, fixpack 5 (10.0.0.5) has made significant changes in the Toolkit representation of REST APIs, the editor functions, and with the tools provided with the Mapping Node editor. This lab will make use of those functions to extract the required employee number from the input parameters. An approach is also provided to retrieve the same information using ESQL.

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.



```
IIB Event Log Monitor
BIP31321: < IB1@NODE.server1 > The HTTP Listener has started listening on port '
'7800' for 'http' connections. [10/3/2014 3:17:23 PM]
BIP21541: < IB1@NODE.server1 > Execution group finished with Configuration messa
ge. [10/3/2014 3:17:24 PM]
BIP21521: < IB1@NODE.server1 > Configuration message received from broker. [10/3
/2014 5:07:36 PM]
BIP21531: < IB1@NODE.server1 > About to 'Change' an execution group. [10/3/201
4 5:07:36 PM]
BIP21551: < IB1@NODE.server1 > About to 'create' the deployed resource 'Empl
oyeeService_JSONClient' of type '.APPZIP'. [10/3/2014 5:07:37 PM]
BIP21551: < IB1@NODE.server1 > About to 'create' the deployed resource 'gen.
getEmployee_EmployeeService_EmpServClient_JSON1' of type '.SUBFLOW'. [10/3/20
14 5:07:37 PM]
BIP21551: < IB1@NODE.server1 > About to 'create' the deployed resource 'EmpS
ervClient_JSON1' of type '.MSGFLOW'. [10/3/2014 5:07:37 PM]
BIP21541: < IB1@NODE.server1 > Execution group finished with Configuration messa
ge. [10/3/2014 5:07:43 PM]
BIP31321: < IB1@NODE.HTTPListener > The HTTP Listener has started listening on p
ort '7800' for 'http' connections. [10/3/2014 5:07:47 PM]
BIP21521: < IB1@NODE.server1 > Configuration message received from broker. [10/3
/2014 5:50:41 PM]
BIP21531: < IB1@NODE.server1 > About to 'Change' an execution group. [10/3/201
4 5:50:41 PM]
BIP21551: < IB1@NODE.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.

1.3 Configure TESTNODE_iibuser for REST APIs

The instructions in this lab guide are based on a Windows implementation, with a user named "iibuser".
The Windows VMWare image on which this lab is based is not available outside IBM, so you will need to provide your own software product installations where necessary.

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. Cross-Origin Resource Scripting (CORS) must be enabled for the IIB node to execute REST applications. This is also 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.

1. Ensure that TESTNODE_iibuser is started.
2. Check that CORS has been enabled on the IIB node by running the following command in an Integration Console:

```
mqsireportproperties TESTNODE_iibuser
-e default
-o HTTPConnector
-r
```

3. If CORS is enabled, you will see the following lines (amongst others):

```
corsEnabled='true'
corsAllowOrigins='*'
corsAllowCredentials='false'
corsExposeHeaders='Content-Type'
corsMaxAge='-1'
corsAllowMethods='GET,HEAD,POST,PUT,PATCH,DELETE,OPTIONS'
corsAllowHeaders='Accept,Accept-Language,Content-Language,Content-Type'
```

4. If CORS has not been enabled, run the following commands:

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

mqsistop TESTNODE_iibuser

mqsistart TESTNODE_iibuser
```

2. Create the HRDB database in MongoDB

This chapter will provide some very basic instructions to get you started with MongoDB, sufficient to perform this lab. There are many good instructional documents on the internet, starting with this site: <https://www.mongodb.org/>.

Note - on the IIB Workshop Windows system, MongoDB has been installed into the Program Files folder, and c:\Program Files\MongoDB\server\3.2\bin has been added to the PATH in the Windows System Environment Variables.

2.1 Start the MongoDB server and client shell

1. In a Windows Command Prompt, navigate to :

```
c:\student10\Loopback\mongodb\commands
```

Run the command:

```
startMongoDB
```

For info, this will run the MongoDB command:

```
mongod.exe --dbpath c:\student10\Loopback\mongodb\data\db
```

This command will start the MongoDB server, and create any new databases in the specified folder. The folder specified in the dbpath parameter must exist, but does not need any specific preparation or configuration.

No defaults have been changed, so the MongoDB server will start with the client listener 27017. The DOS window will be held open at this point. Do not close this window, or the MongoDB server will terminate.

```
C:\student10\Loopback\mongodb\commands>startMongoDB.cmd
C:\student10\Loopback\mongodb\commands>mongod --dbpath c:\student10\loopback\mon
godb\data\db
2016-05-03T08:32:49.109+0100 I CONTROL [main] Hotfix KB2731284 or later update
is not installed. will zero-out data files
2016-05-03T08:32:49.114+0100 I CONTROL [initandlisten] MongoDB starting : pid=9
92 port=27017 dbpath=c:\student10\loopback\mongodb\data\db 64-bit host=BETAWORKS
-ESB10
2016-05-03T08:32:49.118+0100 I CONTROL [initandlisten] targetMinOS: Windows 7/W
indows Server 2008 R2
2016-05-03T08:32:49.120+0100 I CONTROL [initandlisten] db version v3.2.5
2016-05-03T08:32:49.120+0100 I CONTROL [initandlisten] git version: 34e65e5383f
7ea1726332cb175b73077ec4a1b02
2016-05-03T08:32:49.122+0100 I CONTROL [initandlisten] OpenSSL version: OpenSSL
1.0.1p-fips 9 Jul 2015
2016-05-03T08:32:49.125+0100 I CONTROL [initandlisten] allocator: tcmalloc
2016-05-03T08:32:49.126+0100 I CONTROL [initandlisten] modules: none
2016-05-03T08:32:49.128+0100 I CONTROL [initandlisten] build environment:
2016-05-03T08:32:49.131+0100 I CONTROL [initandlisten] distmod: 2008plus-ss
l
2016-05-03T08:32:49.133+0100 I CONTROL [initandlisten] distarch: x86_64
2016-05-03T08:32:49.135+0100 I CONTROL [initandlisten] target_arch: x86_64
2016-05-03T08:32:49.136+0100 I CONTROL [initandlisten] options: < storage: < db
Path: "c:\student10\loopback\mongodb\data\db" > >
```

2. Start a Mongo client shell.

Open a new Windows Command Prompt, and execute the command `"mongo"`.

This will use the default port of 27017, and connect to the started server.

Note that the mongo client will initially connect to the server, and will connect to the **test** database.

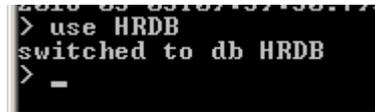
```
C:\Users\iibuser>mongo
2016-05-03T09:39:58.417+0100 I CONTROL [main] Hotfix KB2731284 or later update is not installed, wi
ll zero-out data files
MongoDB shell version: 3.2.5
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
Server has startup warnings:
2016-05-03T09:39:50.790+0100 I CONTROL [initandlisten]
2016-05-03T09:39:50.790+0100 I CONTROL [initandlisten] ** WARNING: Insecure configuration, access c
ontrol is not enabled and no --bind_ip has been specified.
2016-05-03T09:39:50.791+0100 I CONTROL [initandlisten] **           Read and write access to data an
d configuration is unrestricted.
2016-05-03T09:39:50.792+0100 I CONTROL [initandlisten] **           and the server listens on all av
ailable network interfaces.
2016-05-03T09:39:50.793+0100 I CONTROL [initandlisten]
>
```

2.2 Create the HRDB database

With MongoDB, it is not necessary to explicitly define a database. A database is created on first reference, so you will do this now.

1. As mentioned above, the default database connection is "test", so switch to the required database name by issuing the mongo shell command :

```
use HRDB
```



```
> use HRDB
switched to db HRDB
>
```

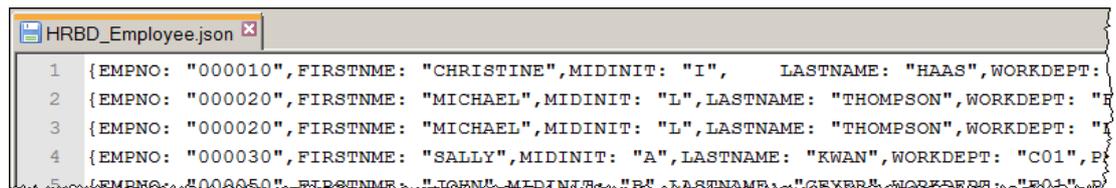
2. The lab has provided a set of data for the HRDB database. In Windows Explorer, locate the file

```
c:\student10\Loopback\mongodb\createdb\
  HRDB_Employee_batch_load.json
```

Open this file with Notepad++.

You will see the familiar contents of the HRDB EMPLOYEE data in JSON format.

Each line has a "{" initiator and "}" terminator.



```
1 {EMPNO: "000010",FIRSTNME: "CHRISTINE",MIDINIT: "I",   LASTNAME: "HAAS",WORKDEPT:
2 {EMPNO: "000020",FIRSTNME: "MICHAEL",MIDINIT: "L",LASTNAME: "THOMPSON",WORKDEPT: "F
3 {EMPNO: "000020",FIRSTNME: "MICHAEL",MIDINIT: "L",LASTNAME: "THOMPSON",WORKDEPT: "I
4 {EMPNO: "000030",FIRSTNME: "SALLY",MIDINIT: "A",LASTNAME: "KWAN",WORKDEPT: "C01",P
5 {EMPNO: "000050",FIRSTNME: "JOHN",MIDINIT: "B",LASTNAME: "CRIVER",WORKDEPT: "F01",P
```

Close the file without saving any changes.

3. You will now import this json data into a "Collection" called **EMPLOYEE** in the **HRDB** database.

In a new Windows command prompt (not the window running the mongo shell), navigate to:

```
c:\student10\Loopback\mongodb\createdb\
```

Enter the command:

```
mongoimport --db HRDB
             --collection EMPLOYEE
             --drop
             --file HRDB_Employee_batch_load.json
```

This command will create a collection called **EMPLOYEE** and import the json data in **HRDB_Employee_batch_load.json** (deleting any previous collection).

4. The mongo import command will respond similar to the following:

```
2016-05-12T16:34:12.710+0100    connected to: localhost
2016-05-12T16:34:12.714+0100    dropping: HRDB.EMPLOYEE
2016-05-12T16:34:12.742+0100    imported 44 documents
```

5. Check that you can retrieve an entry from the Collection.

In the window running the Mongo client shell, execute the command:

```
db.EMPLOYEE.find ( { EMPNO: "000010" } )
```

This should return a single document containing the details of employee 000010;

```
> db.EMPLOYEE.find({EMPNO: "000010"})

{ "_id" : ObjectId("57347fce015454d44dea9d45"), "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, "BONUS" : 1000, "COMM" : 4220 }
```

6. Finally, you can select all documents, by using the command

```
db.EMPLOYEE.find ( )
```

```
> db.EMPLOYEE.find()

{ "_id" : ObjectId("5734a274015454d44dea9d70"), "EMPNO" : "000020", "FIRSTNME" :
"MICHAEL", "MIDINIT" : "L", "LASTNAME" : "THOMPSON", "WORKDEPT" : "B01", "PHONENO" :
"3476", "HIREDATE" : "2003-10-10", "JOB" : "MANAGER ", "EDLEVEL" : 18, "SEX" : "M",
"BIRTHDATE" : "1978-02-02", "SALARY" : 94250, "BONUS" : 800, "COMM" : 3300 }
{ "_id" : ObjectId("5734a274015454d44dea9d71"), "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, "BONUS" : 1000, "COMM" : 4220 }
{ "_id" : ObjectId("5734a274015454d44dea9d72"), "EMPNO" : "000020", "FIRSTNME" :
"MICHAEL", "MIDINIT" : "L", "LASTNAME" : "THOMPSON", "WORKDEPT" : "B01", "PHONENO" :
"3476", "HIREDATE" : "2003-10-10"..... etc
```

7. Now that you have loaded some test data into the MongoDB database HRDB, the next chapter will show you how to access this from IIB.

3. Configure the IIB installation

You can use a LoopBackRequest node in a message flow to access or create records through a LoopBack connector. First, you must install your chosen connector to work with IBM Integration Bus, and then configure the data source for the connector. You can then specify the security credentials that are required to access it. The following topics describe the steps involved in completing those tasks:

1. Installing the LoopBack connector
2. Configuring the data source and models for your LoopBack connector
3. Optionally - specifying security credentials for connecting to a secured data source

3.1 Install the LoopBack Connector

1. In an Integration Bus Command Console, change directory to the `node_modules` folder of the work path directory (defined by the `MQSI_WORKPATH` environment variable):

```
cd \  
cd %MQSI_WORKPATH%\node_modules
```

Install the required loopback connector for MongoDB by running the command

```
npm install loopback-connector-mongodb
```

This will create the folder **node-modules**, which contains the loopback connector for mongodb. ("npm" is a nodejs command, which is installed on the workshop Windows system.)

The output of the command should look like this. You can use Windows Explorer to check the contents of the `connectors\node_modules` folder.

```
C:\IBM\IIB\10.0.0.6>cd\  
C:\>cd %MQSI_WORKPATH%\node_modules  
C:\ProgramData\IBM\MQSI\node_modules>npm install loopback-connector-mongodb  
loopback-connector-mongodb@1.15.2 loopback-connector-mongodb  
├── async@1.5.2  
├── debug@2.2.0 <ms@0.7.1>  
├── loopback-connector@2.4.0  
└── mongodb@2.2.9 <es6-promise@3.2.1, readable-stream@2.1.5, mongodb-core@2.0.11>  
C:\ProgramData\IBM\MQSI\node_modules>
```

3.2 Configure data source for Loopback connector

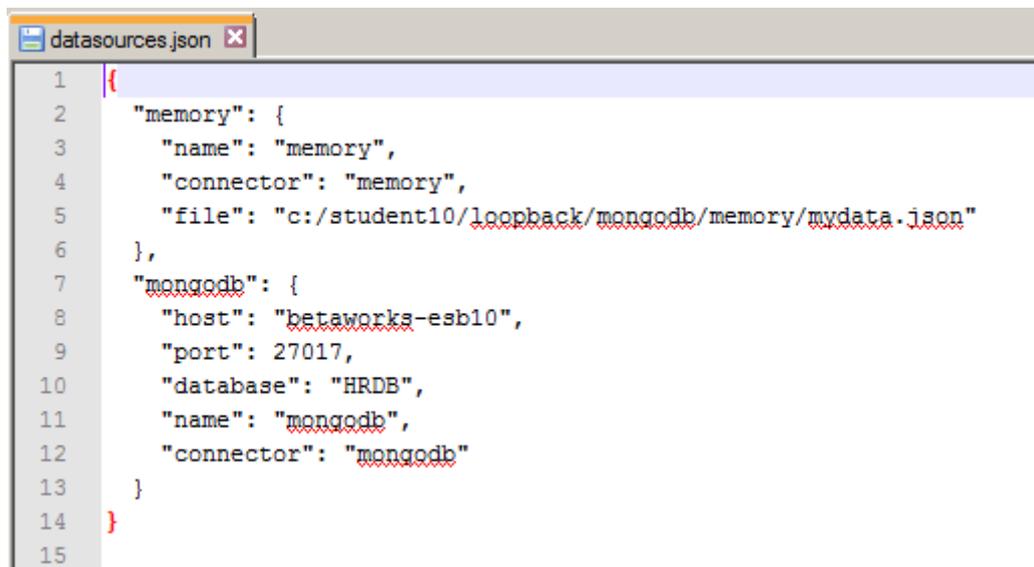
When you have installed a LoopBack connector to be used with IBM Integration Bus, you must configure the data source for the connector, by adding a connector-specific data-source stanza to the `datasources.json` file.

1. The creation process for the loopback connector does not automatically create the `datasources.json` connections details, so this must be created manually.

In this example, we have provided the required `datasources` file, so using Windows Explorer, copy the file `c:\student10\loopback\mongodb\config\datasources.json`.

Copy this file into the folder into the `MQSI_WORKPATH\connectors\loopback` folder. On the provided Windows workshop system, this folder will be **c:\ProgramData\IBM\MQSI\connectors\loopback**.

Using Notepad++, open this file and take a look at the contents. The required stanza is the "mongodb" stanza, which corresponds to the data source name in the Loopback node properties in the IIB Toolkit. If you are using your own installation, you will need to change the hostname and port values accordingly.



```
1 {
2   "memory": {
3     "name": "memory",
4     "connector": "memory",
5     "file": "c:/student10/loopback/mongodb/memory/mydata.json"
6   },
7   "mongodb": {
8     "host": "betaworks-esb10",
9     "port": 27017,
10    "database": "HRDB",
11    "name": "mongodb",
12    "connector": "mongodb"
13  }
14 }
15 }
```

4. Create an IIB REST API to access MongoDB

4.1 Create the new REST API

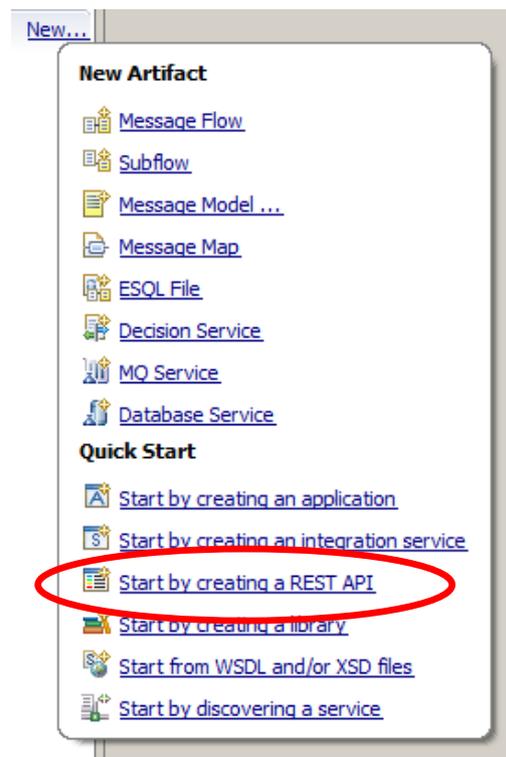
1. In the IIB Toolkit, create a new workspace called

`c:\user\iibuser\IBM\IIBT10\workspace_loopback`

Use File, Switch workspace, Other to do this.

Note that it is important to restart the IIB Toolkit at this point, and to create a new workspace. The changes that you made to the MQSI_WORKPATH in section 3 can only be picked up by the IIB Toolkit after a restart.

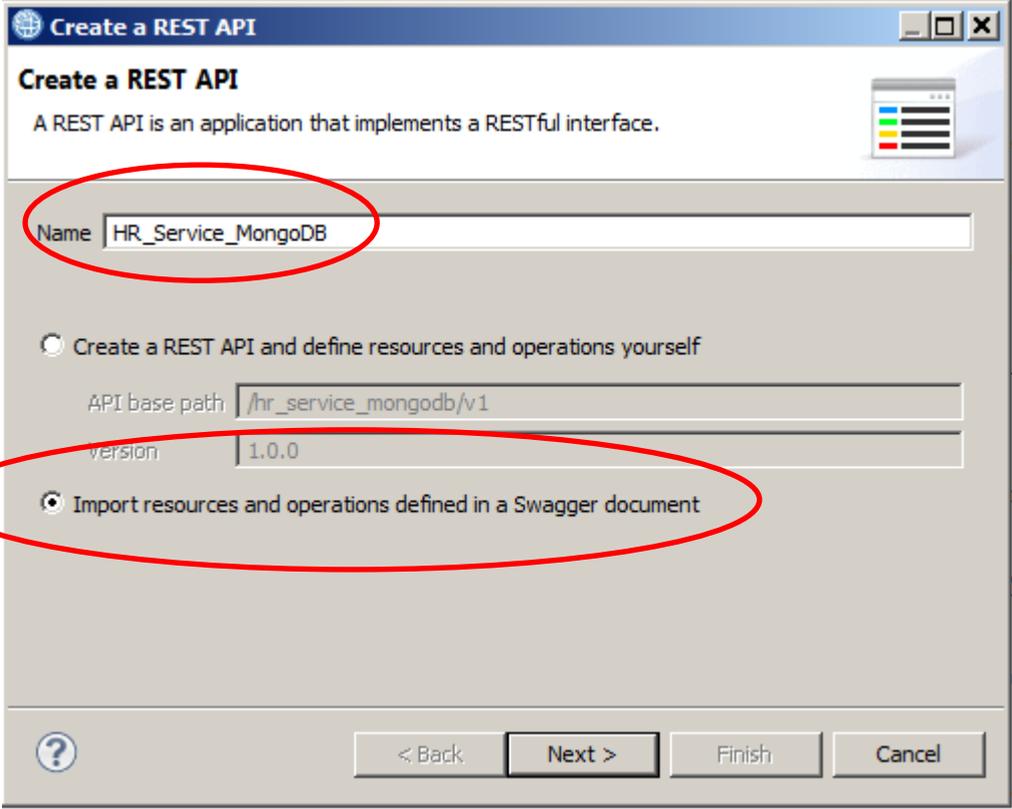
2. In the workspace, create a new REST API..



3. Name the new service HR_Service_MongoDB

Select "Import resources and operations defined in a Swagger document".

Click Next.



Create a REST API

A REST API is an application that implements a RESTful interface.

Name:

Create a REST API and define resources and operations yourself

API base path:

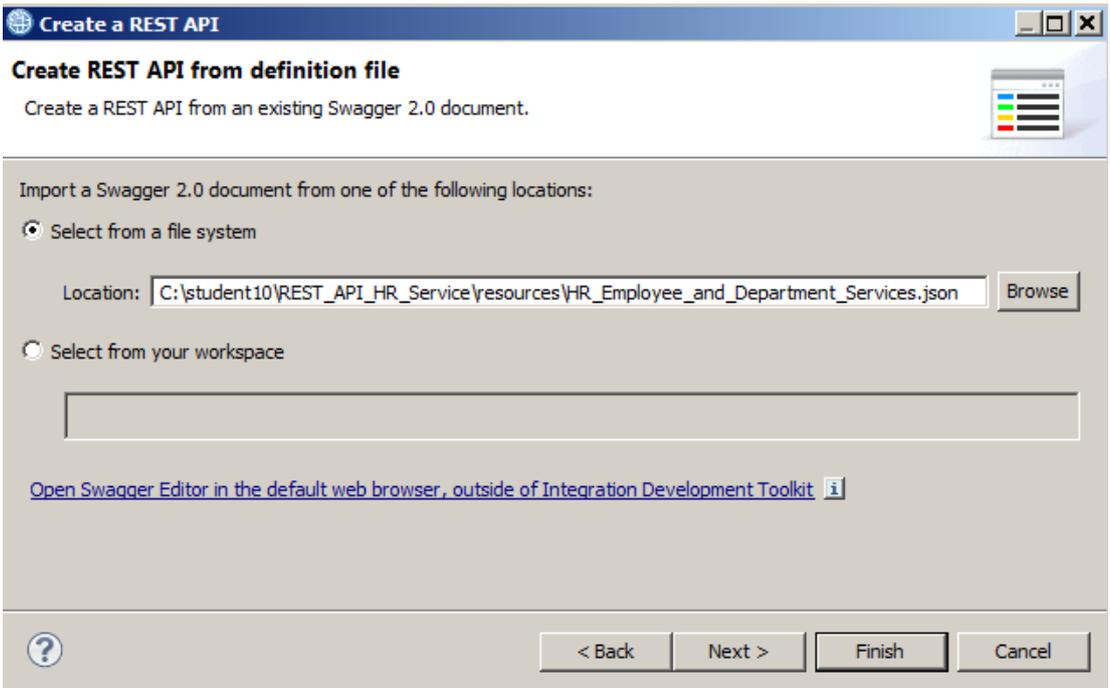
version:

Import resources and operations defined in a Swagger document

< Back Next > Finish Cancel

4. Using the Browse button, import the JSON document

```
c:\student10\REST_API_HR_Service\resources\  
HR_Employee_and_Department_Services.json
```



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: Browse

Select from your workspace

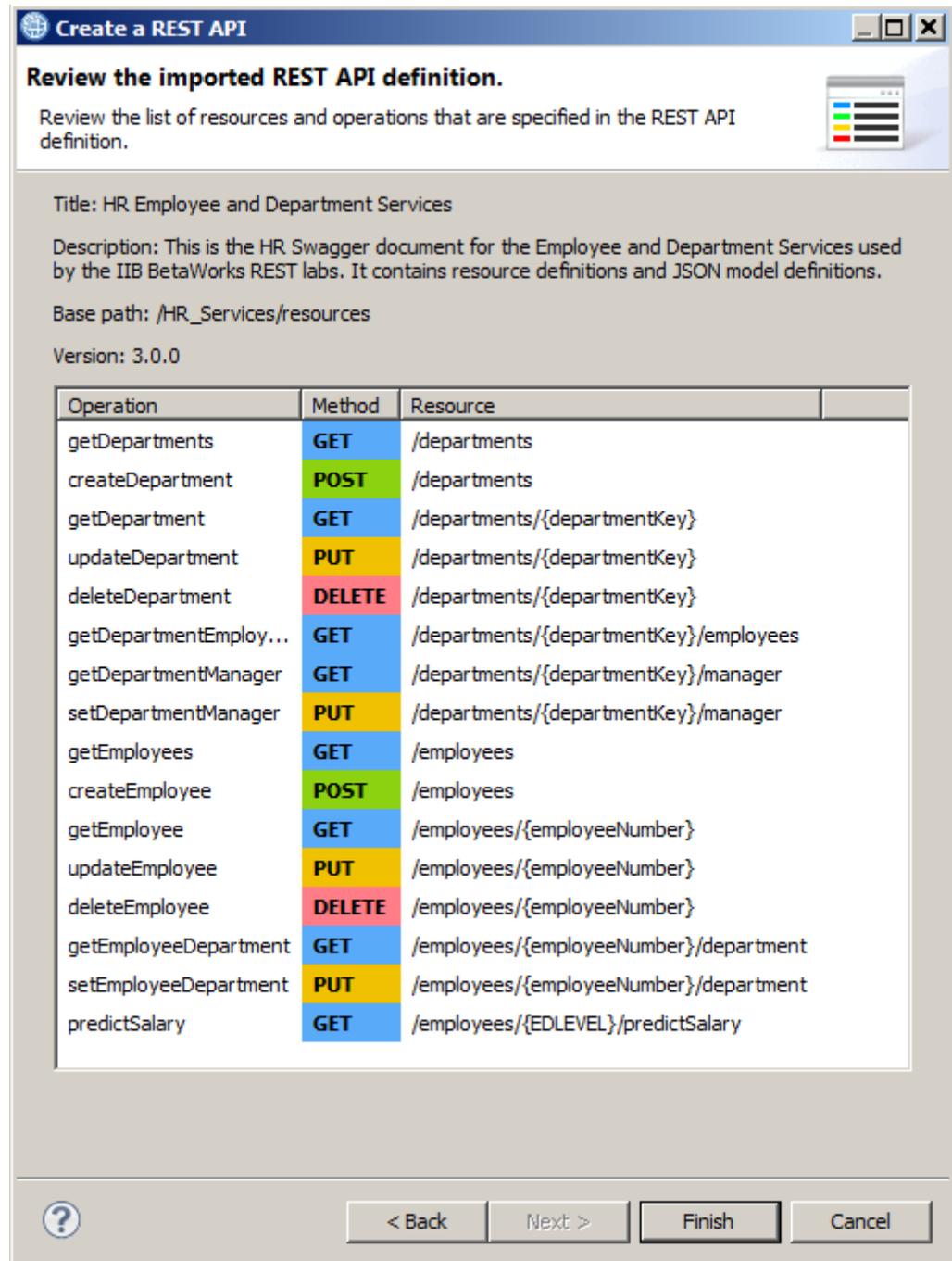
[Open Swagger Editor in the default web browser, outside of Integration Development Toolkit](#) 

< Back Next > Finish Cancel

5. 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.



Create a REST API

Review the imported REST API definition.

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

Title: HR Employee and Department Services

Description: This is the HR Swagger document for the Employee and Department Services used by the IIB BetaWorks REST labs. It contains resource definitions and JSON model definitions.

Base path: /HR_Services/resources

Version: 3.0.0

Operation	Method	Resource
getDepartments	GET	/departments
createDepartment	POST	/departments
getDepartment	GET	/departments/{departmentKey}
updateDepartment	PUT	/departments/{departmentKey}
deleteDepartment	DELETE	/departments/{departmentKey}
getDepartmentEmploy...	GET	/departments/{departmentKey}/employees
getDepartmentManager	GET	/departments/{departmentKey}/manager
setDepartmentManager	PUT	/departments/{departmentKey}/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
predictSalary	GET	/employees/{EDLEVEL}/predictSalary

Buttons: ? < Back Next > Finish Cancel

6. The swagger document has now been imported into the Integration Toolkit. The import process has also created a base REST application and a message flow that implements the REST API.

The imported and generated items are split into five main sections in the REST API editor:

- Header - containing the base URL for the REST API, title and description
- Resources - containing all the resources from the swagger document, and all of the operations that are contained within each resource
- Model Definitions - schema definitions for the input and output JSON objects
- Error Handling - options to add some elements of runtime security
- Security - basic security parameters

HR_Service

- ▶ Header

- ▼ Resources
 - ▶ /departments
 - ▶ /departments/{departmentKey}
 - ▶ /departments/{departmentKey}/employees
 - ▶ /departments/{departmentKey}/manager
 - ▶ /employees
 - ▶ /employees/{EDLEVEL}/predictSalary
 - ▶ /employees/{employeeNumber}
 - ▶ /employees/{employeeNumber}/department

- ▼ Model Definitions

Name
⊕ <Enter a unique name to create a new model>
⊕ {...} EMPLOYEE
⊕ {...} DEPARTMENT
⊕ {...} DBRESP
⊕ {...} EmployeeResponse
⊕ {...} DepartmentResponse
⊕ {...} DetailedResponse

- As an example of Resources, expand **/employees/{employeeNumber}**. (You may wish to collapse the **/department** resource, for readability).

You will see three operations, GET, PUT and DELETE.

For some of the operations (for example, the updateEmployee PUT operation in this resource), the Schema type has been set (in this case to EMPLOYEE).

For some other operations (for example the getEmployee GET operation), the input parameter is specified (EMPNO). The Schema Type of the successful (200) operation has been set to EmployeeResponse (originally specified in the swagger doc).

You can use the Schema Type dropdowns to change the required schema for the operation. The available values are derived from the Model Definitions section. If no schema type is specified, the REST operation can dynamically specify the format of the output message.

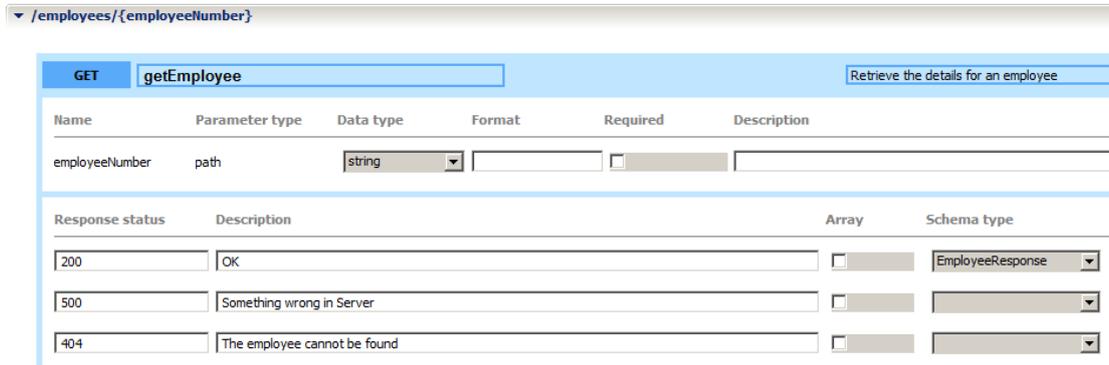
The screenshot displays the configuration for three REST API operations under the resource `/employees/{employeeNumber}`:

- GET getEmployee:**
 - Parameter: `employeeNumber` (path, string, required).
 - Response 200: Description "OK", Schema type `EmployeeResponse`.
 - Response 500: Description "Something wrong in Server".
 - Response 404: Description "The employee cannot be found".
- PUT updateEmployee:**
 - Parameter: `employeeNumber` (path, string, required).
 - Request body: Schema type `EMPLOYEE`.
 - Response 200: Description "Updated".
 - Response 500: Description "A problem occurred updating the employee".
 - Response 400: Description "There was a problem with the request".
 - Response 404: Description "The employee cannot be found".
- DELETE deleteEmployee:**
 - Description: "Deletes an existing employee in the database."

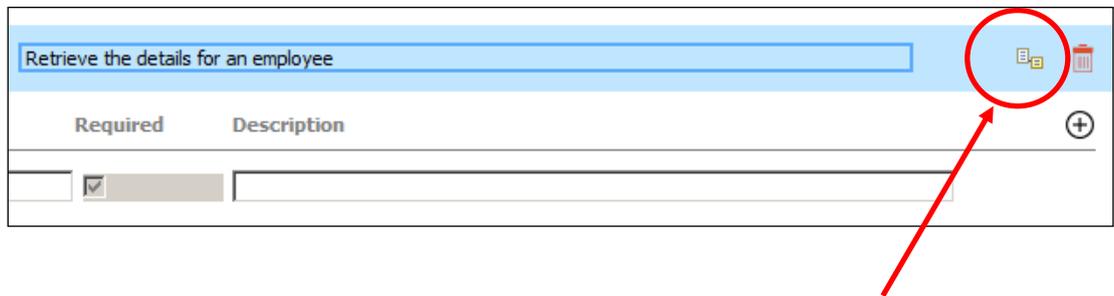
Save the updated REST API (ctrl-S).

- Expand the REST API Resources, and position the editor at the **/employees/{employeeNumber}** resource.

The getEmployee operation is the first operation in this resource.



- In the top right part of the description, click on the icon to "Create a subflow for the operation".



This will create a skeleton subflow where you will provide the logic to implement the operation:



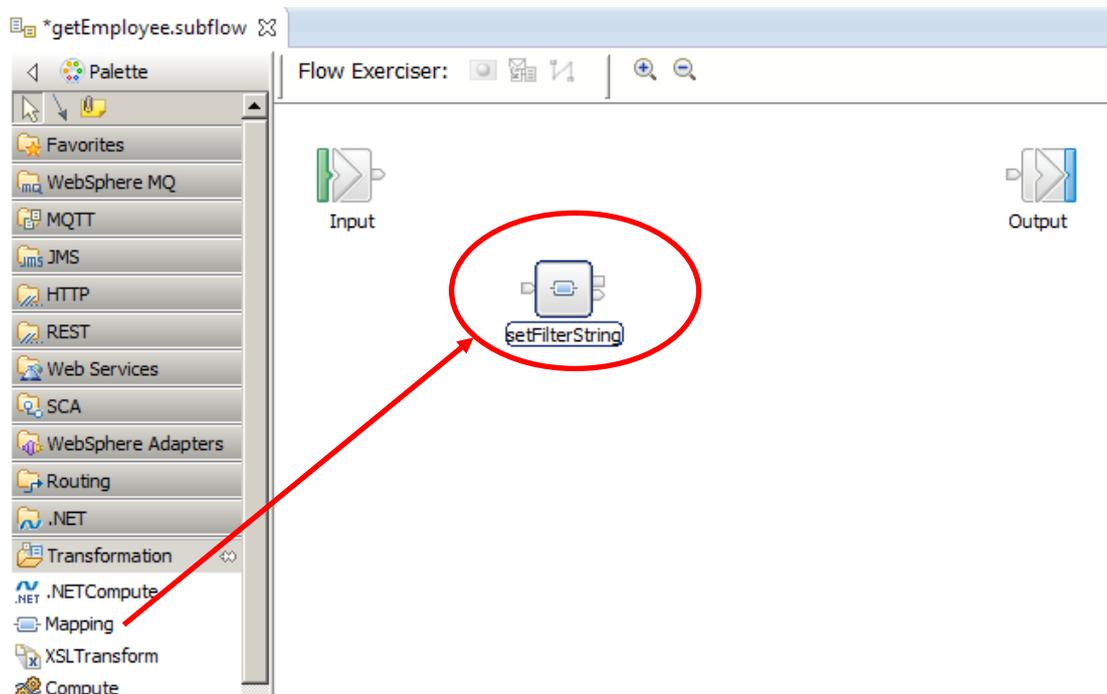
Two alternatives are provided in this lab:

1. In the first alternative, you will use a new Mapping node to set the value of the LocalEnvironment filterString element that is used by the Loopback Request node.
2. In the second alternative, you will use an ESQL Compute node to set the value of the filterString element.

If you wish to use ESQL, skip to "4.3 Alternative 2 - using ESQL".

4.2 Alternative 1: Set LocalEnvironment "filterString" using Mapping Node

1. Add a new Mapping node to the flow editor; call it setFilterString.



- Open the new map. At the New Message Map window, accept the default (Message Map for REST API operation getEmployee), and click Finish.

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 with the input and output for REST API operation getEmployee
- Simple message map called by a message flow node
- Submap called by another map

Container: EmployeeService_REST New

Map name: getEmployee_setFilterString

Map organization

- Use default broker schema

Schema: (default broker schema)

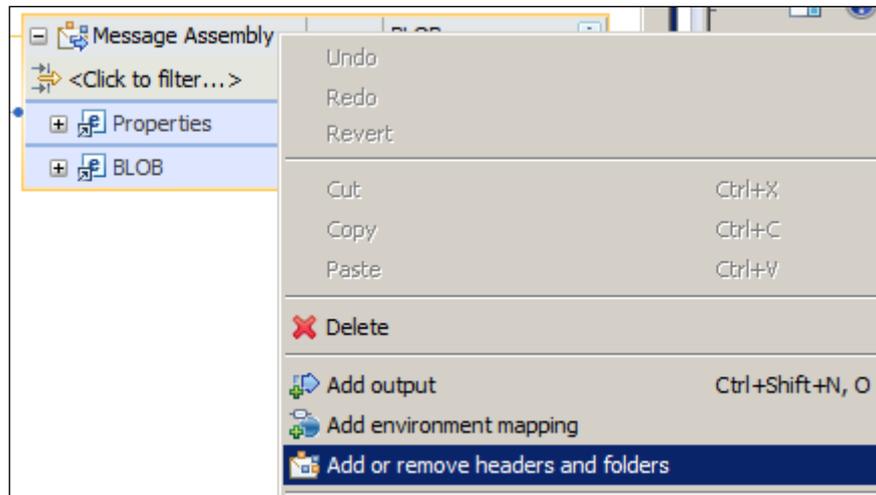
? < Back Next > Finish Cancel

- Note that LocalEnvironment has already been added to the input message assembly.

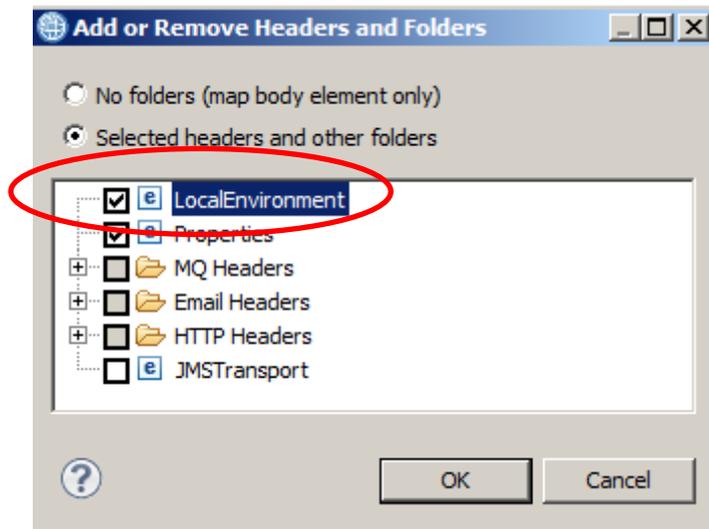
Expanding this, and the REST folder, will show the input parameter `employeeNumber`. You will use this when setting the value of the output **filterString** parameter for the MongoDB access.

[-] [e] REST	[0..1]	_LocalEnvironmentRESTType
[-] [e] Input	[0..1]	_RESTInputType
[e] Method	[0..1]	string
[e] Operation	[0..1]	string
[e] Path	[0..1]	string
[e] URI	[0..1]	string
[-] [e] Parameters	[0..1]	<Anonymous>
[-] [e] choice of cast items	[0..*]	
[e] any	[1..1]	
[e] employeeNumber	[1..1]	string
[+] [e] Response	[0..1]	_RESTResponseType

- On the output message assembly, the LocalEnvironment has not been added, so add it now.
Right-click the Message Assembly title, and select "Add or remove headers and folders".



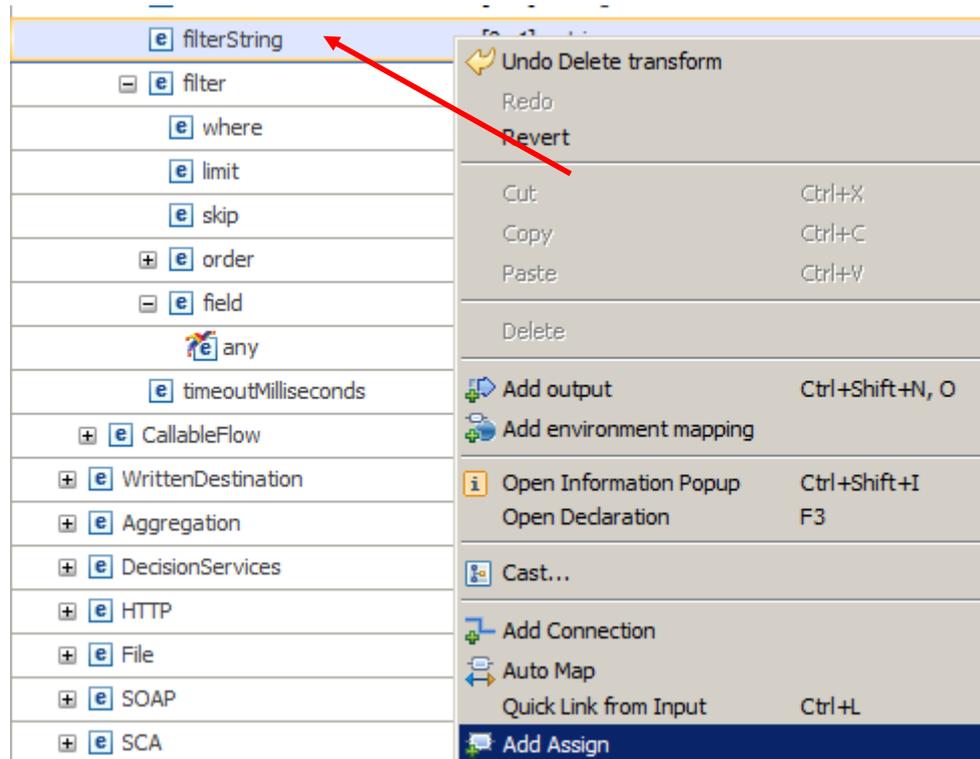
- Select LocalEnvironment, then click OK.



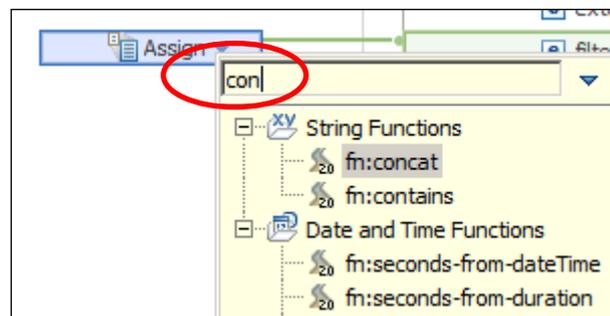
- In the output LocalEnvironment, fully expand the **Destination/Loopback/Request** folder. You will see several elements. For this scenario, you will use the filterString element.

[-] [e] Loopback	[0..1]	_LoopbackDestinationType
[-] [e] Request	[0..1]	_LoopbackRequestType
[e] operation	[0..1]	string
[e] object	[0..1]	string
[e] id	[0..1]	string
[e] externalIdName	[0..1]	string
[e] externalId	[0..1]	string
[e] filterString	[0..1]	string
+ [e] filter	[0..1]	_LoopbackRequestFilterType
[e] timeoutMilliseconds	[0..1]	positiveInteger

7. Right-click filterString, and select "Add Assign" from the context menu.

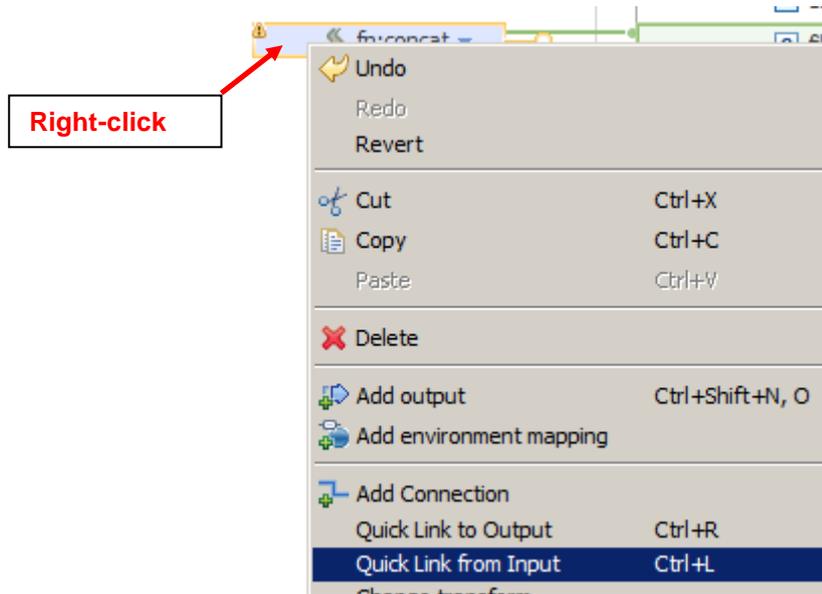


8. Click the blue drop-down arrow on the Assign transform, and using the search field, change the transform to a "fn:concat".



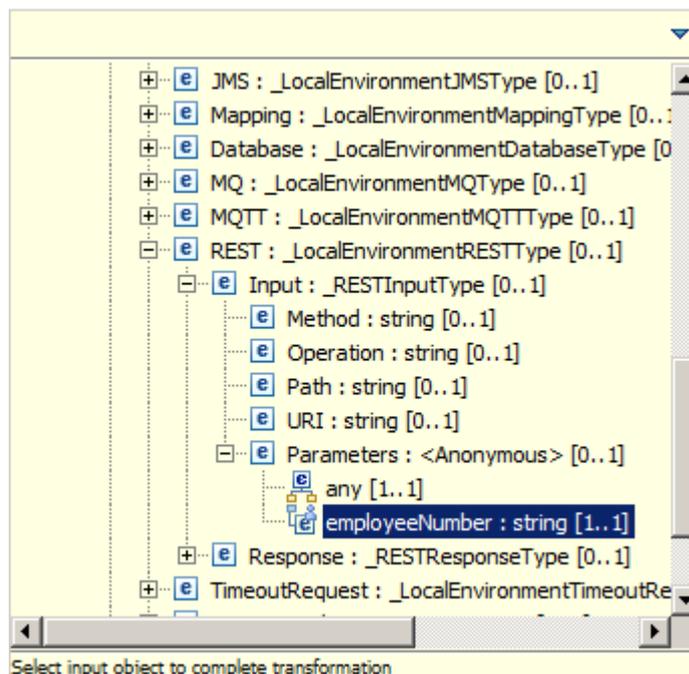
- The Concat transform needs to take input from the EMPNO input element, so a connection to this has to be provided.

Right-click the "fn:concat" transform and select "Quick link from Input".



- In the yellow pop-up window, collapse the Properties folder, expand the LocalEnvironment folder, and locate the REST/Input/Parameters folder.

Select (click) the employeeNumber element.



- Set the value of string2 to \$employeeNumber, by using the drop-down arrow that will be shown when you select this element.

Click here:

Parameters:

Name	Type	Value
string1	xs:string	{'where': {'EMPNO':"
string2	xs:string	employeeNumber
string3	xs:string	<div style="border: 1px solid black; padding: 2px;"> ⌵ employeeNumber ⌵ Edit Custom XPath Expression Parameter... </div>

Result:

Parameters:

Name	Type	Value
string1	xs:string	{'where': {'EMPNO':"
string2	xs:string	\$employeeNumber
string3	xs:string	"

- Finally, set the value of string3 to

' " } } ' }

Parameters:

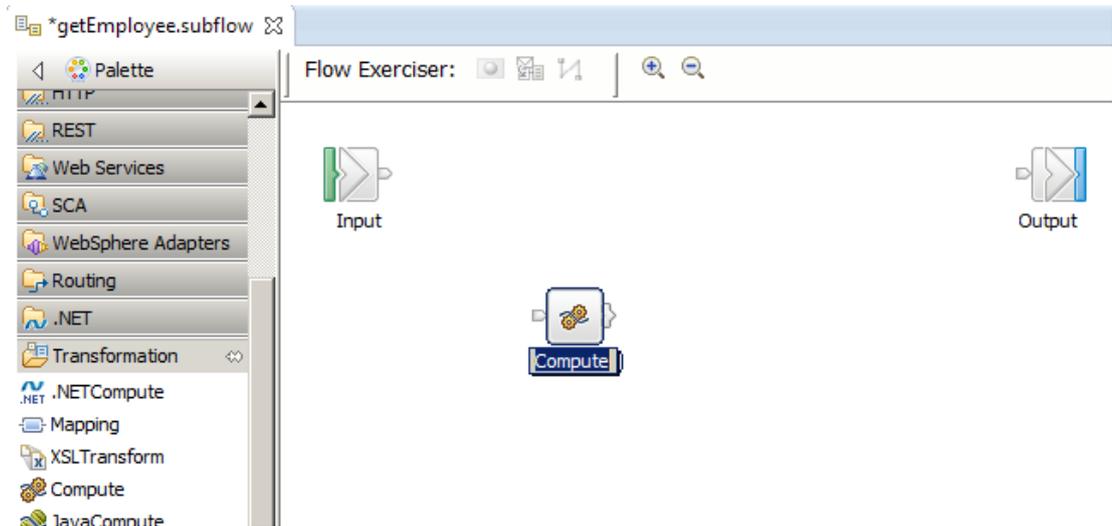
Name	Type	Value
string1	xs:string	{'where': {'EMPNO':"
string2	xs:string	\$employeeNumber
string3	xs:string	"}}}

Save and close the map.

- Skip to 4.4 Complete the Flow Development.

4.3 Alternative 2: Set LocalEnvironment "filterString" using ESQL Compute Node

1. Add a new Compute (ESQL) node to the flow editor.

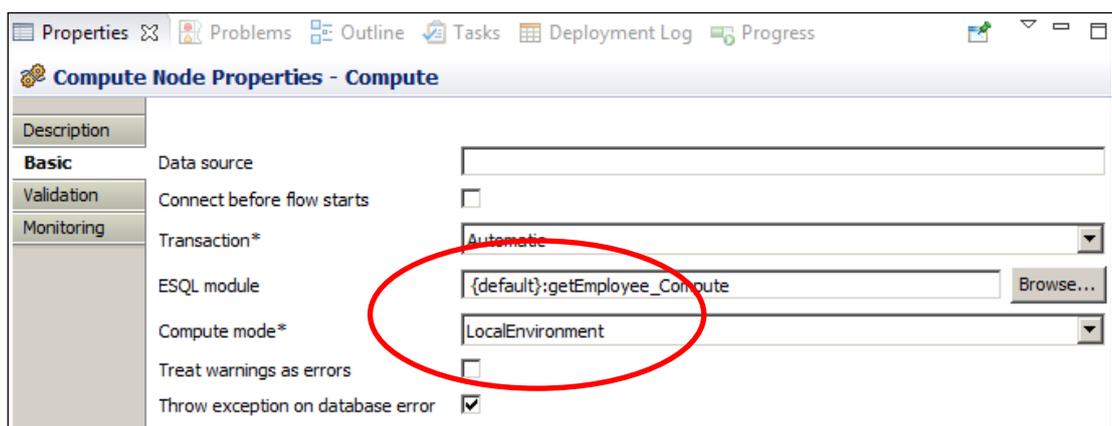


2. Select the Compute node, and in the node properties, set Compute mode to

"LocalEnvironment".

This is required because the ESQL in the Compute node will set the value of an element in the Loopback folder of the LocalEnvironment. This needs to be propagated to the Loopback Request node, so that it can be used to pass the query to the MongoDB database.

You can, of course, set the Compute Mode to other values, providing that the LocalEnvironment is included.



3. Open the Compute node, and add a new line of ESQL after the comment line, as shown:

```
-- CALL CopyEntireMessage();

set OutputLocalEnvironment.Destination.Loopback.Request.filterString =
  '{"where": {"EMPNO":'
  || InputLocalEnvironment.REST.Input.Parameters.employeeNumber
  || '}}';
```

Note, you can copy/paste this line from the file:

```
c:\student10\loopback\mongodb\esql\setFilterString.esql.txt
```

For reference, the ESQL will create a Loopback query element, contained in the **LocalEnvironment/Destination/Loopback/Request/** folder.

As an example, the query will have the following form:

```
filterString = '{"where": {"EMPNO":"000010"}}'
```

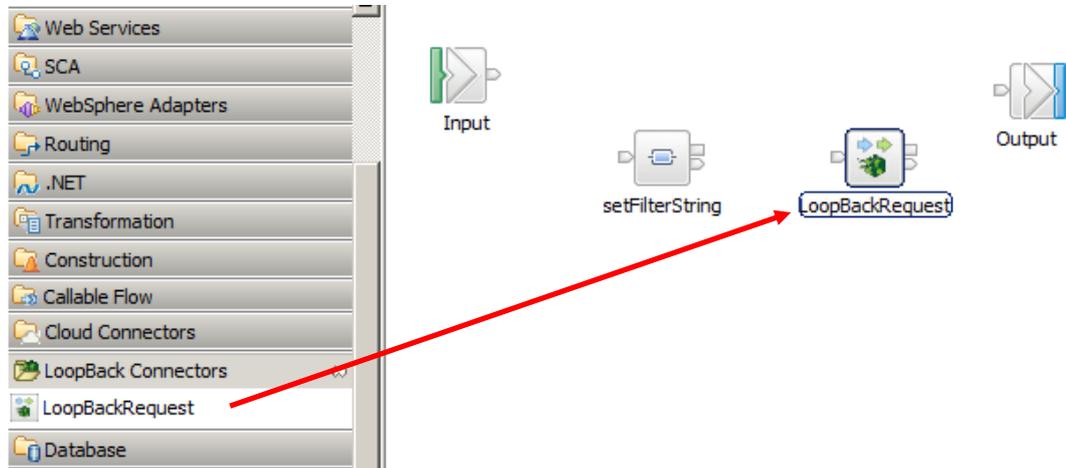
Save and close the ESQL editor.

4.4 Complete the flow development

You will now complete the development of the subflow by using the LoopBack Request node.

The screen captures in this section assume that the Mapping Node has been used to set the filterString, but apply equally if you have used an ESQL Compute node.

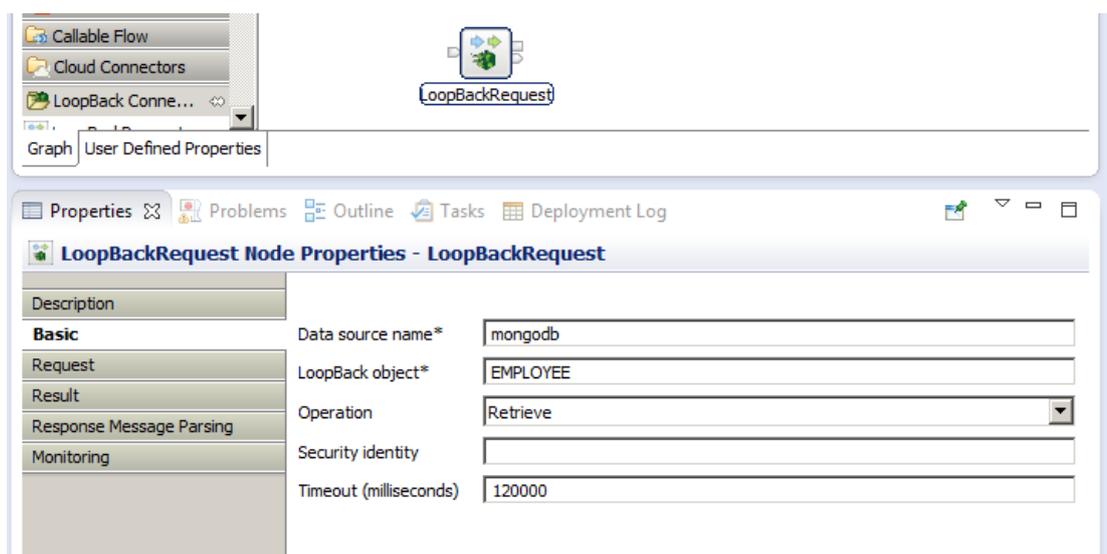
1. From the Loopback Connectors drawer, drop a Loopback Request node onto the flow editor.



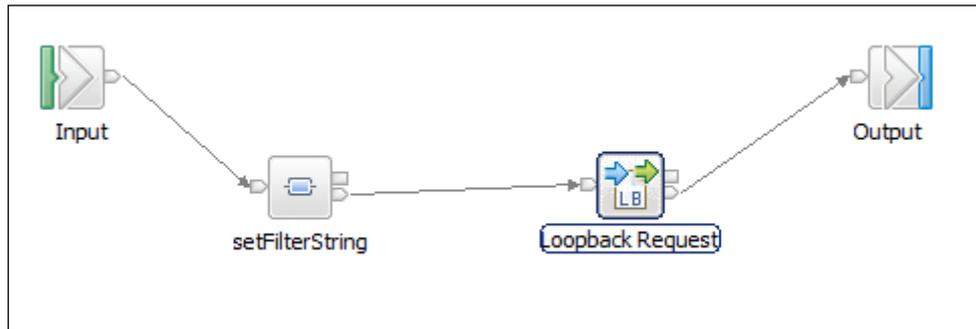
2. Select the Loopback Request node, and select the node Properties.

On the Basic tab, set the following values:

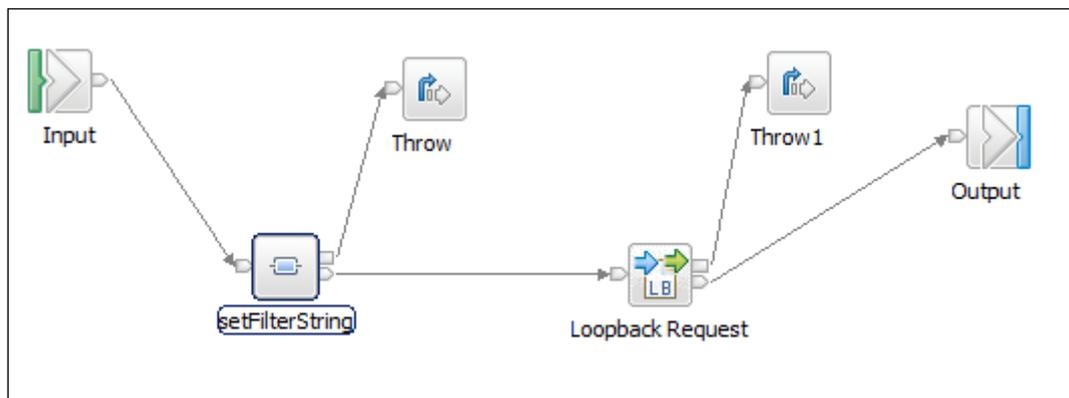
- Name of the data source in the datasources.json file to connect to:
 - **mongodb**
- Loopback object
 - **EMPLOYEE** (corresponding to the HRDB EMPLOYEE container)
- Operation
 - **Retrieve**
- Timeout
 - **1200** (to avoid long timeouts during development)



3. Connect the nodes as shown.



4. Connect Throw nodes (Construction folder) to the failure terminals of each of the nodes. Add suitable message text using the properties tab. This will help with any problem determination you may need when you come to test the REST API:



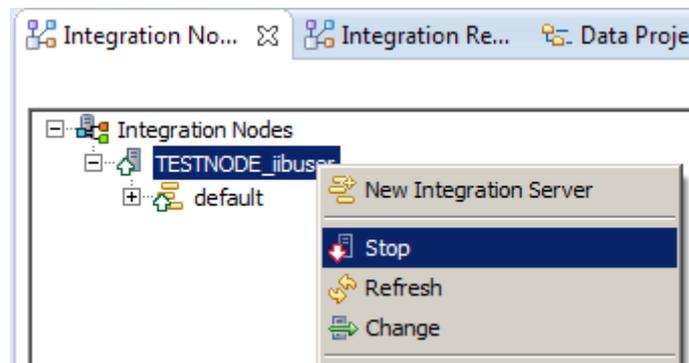
Save the subflow.

4.6 Deploy and test the REST API

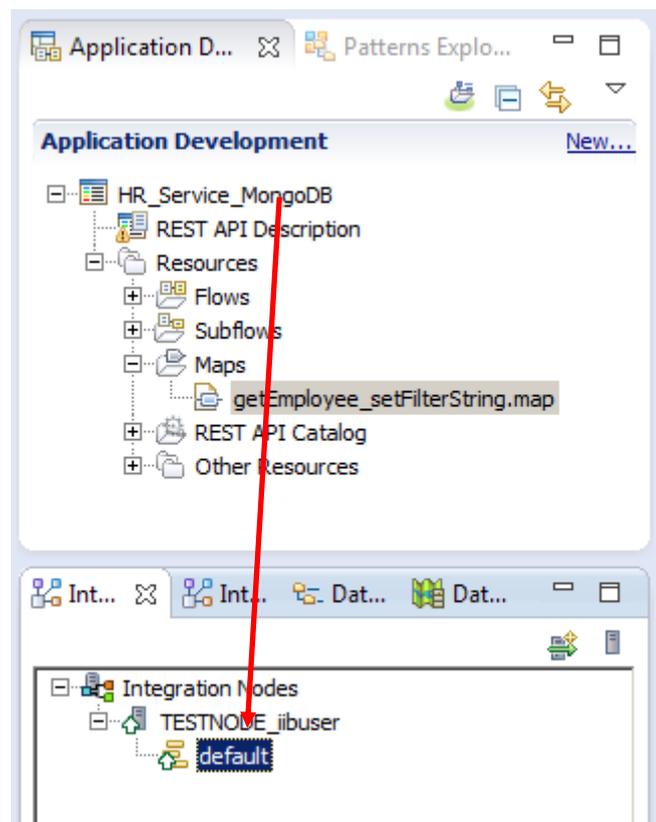
You will now deploy and test the updated REST API. This should retrieve information from the MongoDB HRDB database.

1. First, to activate the changes you made earlier (so that the IIB node can pick up the loopback connector from the MQSI_WORKPATH), stop and restart TESTNODE_ibuser.

You can do this using the Integration Toolkit (right-click node, Stop, then Start), or a variety of other ways.

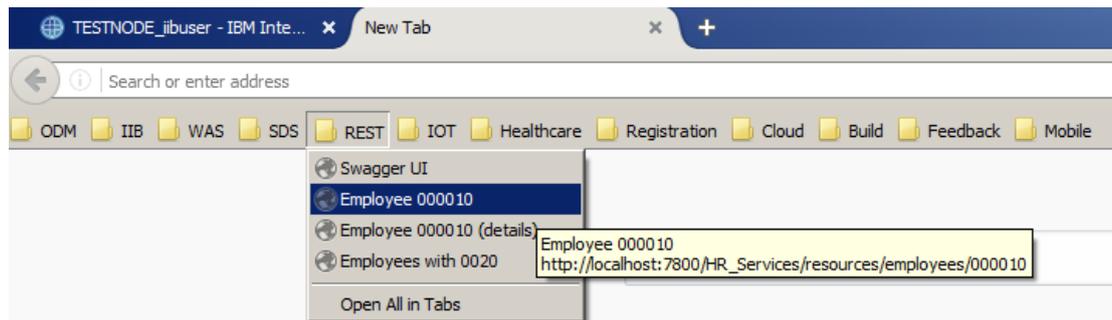


2. In the Integration Toolkit, deploy the HR_Service_MongoDB REST API by dragging and dropping onto the default server.



- In the supplied Firefox browser, use the shortcut in the REST folder to use the URL:

`http://localhost:7800/HR_Services/resources/employees/000010`



- This should retrieve the document corresponding the EMPNO = "000010".



- Terminate the mongo shell window, and the mongod server window, using Ctrl-C in both cases.

END OF LAB GUIDE