**Lab Book**

**TIBCO BusinessWorks 6**

Contents

[1 Software Perquisites 4](#_Toc462390289)

[1.1 TIBCO BW Installation 4](#_Toc462390290)

[1.2 TIBCO EMS Installation 4](#_Toc462390291)

[1.3 GEMS Installation 4](#_Toc462390292)

[1.4 MySQL Installation 4](#_Toc462390293)

[1.5 SoapUI Installation 4](#_Toc462390294)

[1.6 Notepad++ 4](#_Toc462390295)

[2 Lab Exercises 4](#_Toc462390296)

[2.1 Lab Exercise-1 4](#_Toc462390297)

[2.1.1 Problem Definition 4](#_Toc462390298)

[2.1.2 Solution Overview 5](#_Toc462390299)

[2.1.3 Required Artifacts 5](#_Toc462390300)

[2.1.4 Development Steps 5](#_Toc462390301)

[2.1.5 Testing 18](#_Toc462390302)

[2.2 Lab Exercise-2 19](#_Toc462390303)

[2.2.1 Problem Definition 19](#_Toc462390304)

[2.2.2 Solution Overview 19](#_Toc462390305)

[2.2.3 Required Artifacts 20](#_Toc462390306)

[2.2.4 Development Steps 20](#_Toc462390307)

[2.2.5 Testing 25](#_Toc462390308)

[2.3 Lab Exercise-3 28](#_Toc462390309)

[2.3.1 Problem Definition 28](#_Toc462390310)

[2.3.2 Solution Overview 28](#_Toc462390311)

[2.3.3 Required Artifacts 29](#_Toc462390312)

[2.3.4 Development Steps 29](#_Toc462390313)

[2.3.5 Testing 38](#_Toc462390314)

[3 Homework 42](#_Toc462390315)

[3.1 Homework-1 42](#_Toc462390316)

[3.2 Homework-2 42](#_Toc462390317)

[3.3 Homework-3 43](#_Toc462390318)

# Software Perquisites

## TIBCO BW Installation

Download and install TIBCO ActiveMatrix BusinessWorks 6.X from <https://tap.tibco.com/>

Refer Installation Guide available at <https://docs.tibco.com/products/a_z_products> for any help on installation.

## TIBCO EMS Installation

Download and install TIBCO Enterprise Message Service.

Refer Installation Guide available at <https://docs.tibco.com/products/a_z_products> for any help on installation.

## GEMS Installation

Download and install GEMS from <https://community.tibco.com/modules/graphical-administration-tool-tibcor-ems>. Configure GEMS to connect to default instance of EMS (7222).

## MySQL Installation

Download and install MySQL server from <http://dev.mysql.com/downloads/>.

Download and install MySQL sample database *employee* from <https://dev.mysql.com/doc/employee/en/>.

Configure MySQL Workbench to connect to *employee* database.

## SoapUI Installation

Download and install Open Source version of SoapUI from <https://www.soapui.org/>.

## Notepad++

Download and install Notepad++ from <https://notepad-plus-plus.org>.

# Lab Exercises

## Lab Exercise-1

### Problem Definition

Capgemini has multiple offices across the globe. Capgemini uses a centralized application – Employee Data Management System (EDMS) –for managing employee data. Whenever a new employee joins at any office, a message on a particular queue gets published.

The solution should read the message from the queue create record in the database.

### Solution Overview

Topics Covered: Project Creation, JMS, JDBC, and Logger.

### Required Artifacts

Queue Name: Capgemini.TIBCOTraining.Exercise1.Queue

XML Schema: EmployeeData.xsd

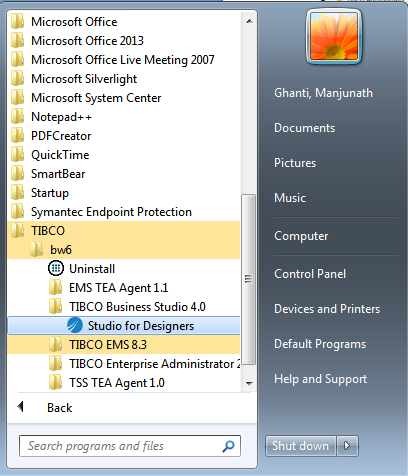


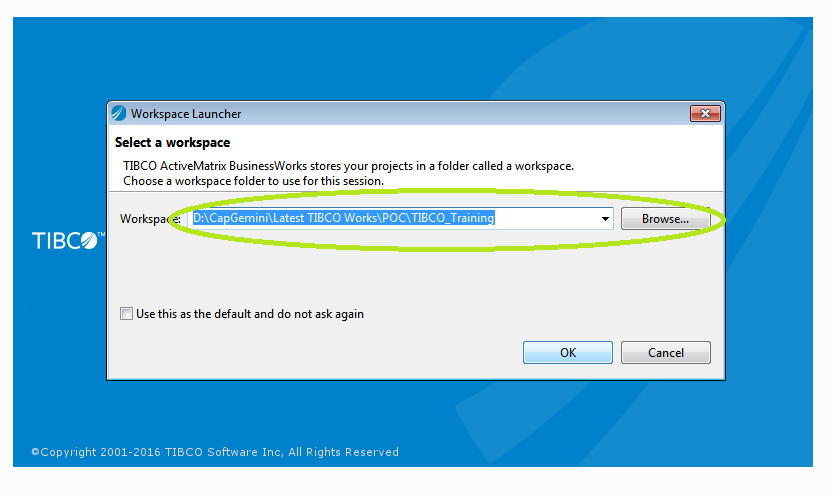
Database Table:

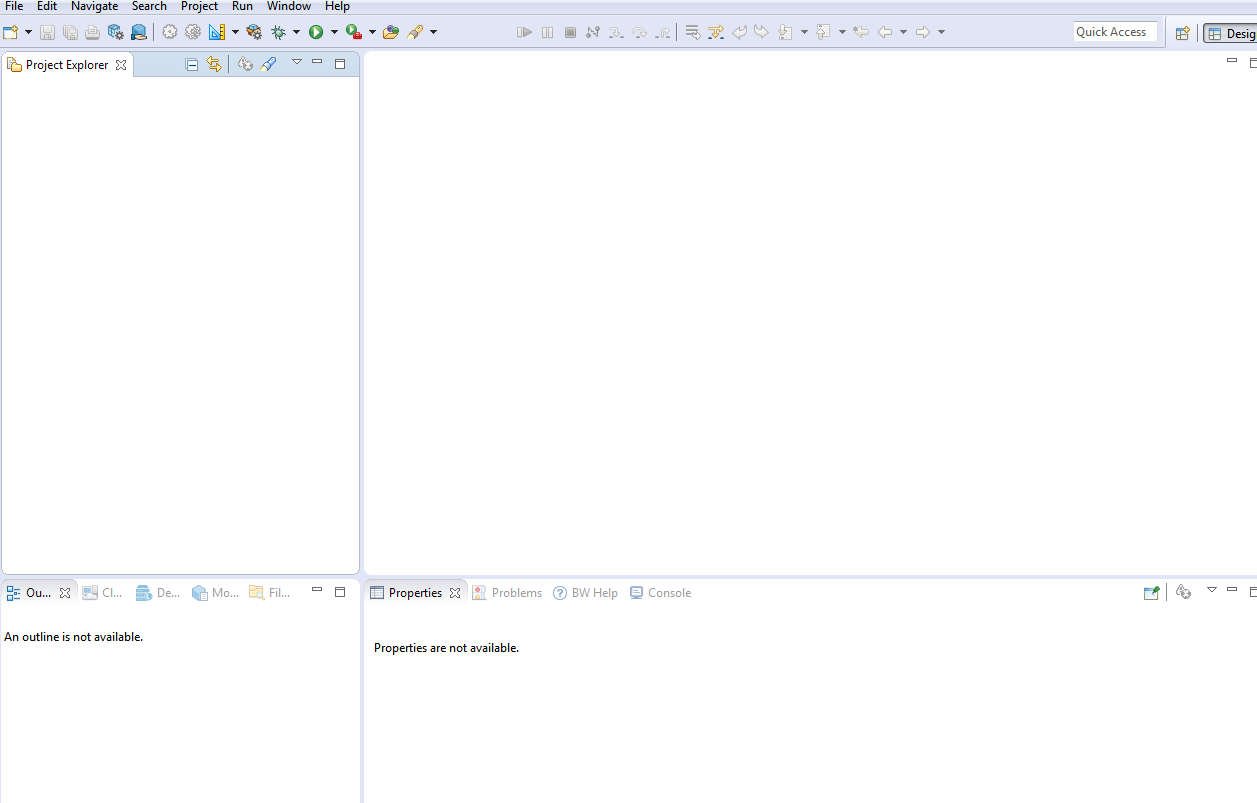


### Development Steps

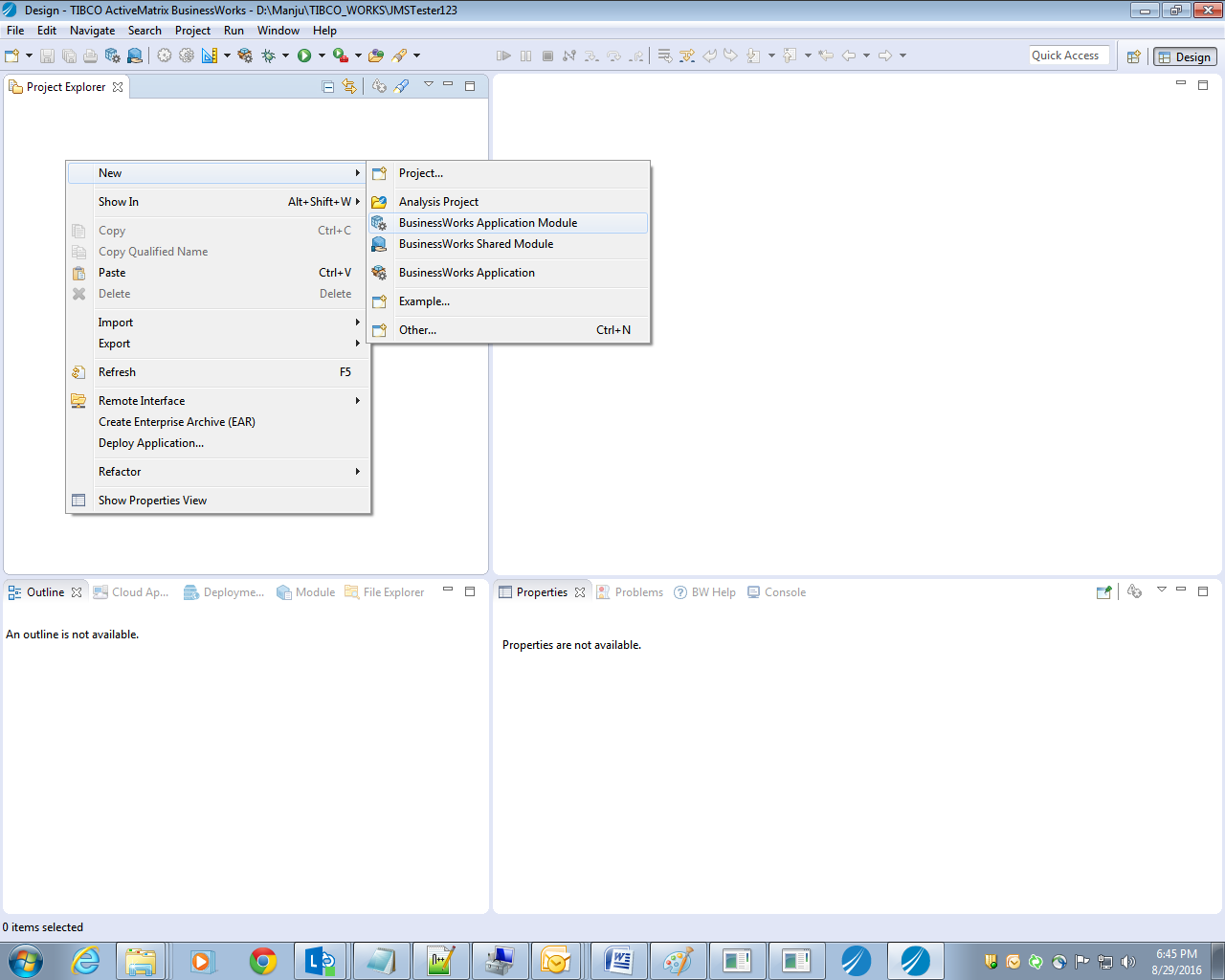
* Open Business Studio by clicking on Start 🡪 All Programs 🡪 TIBCO 🡪bw6 🡪TIBCO Business Studio 4.0 🡪 Studio for Designer

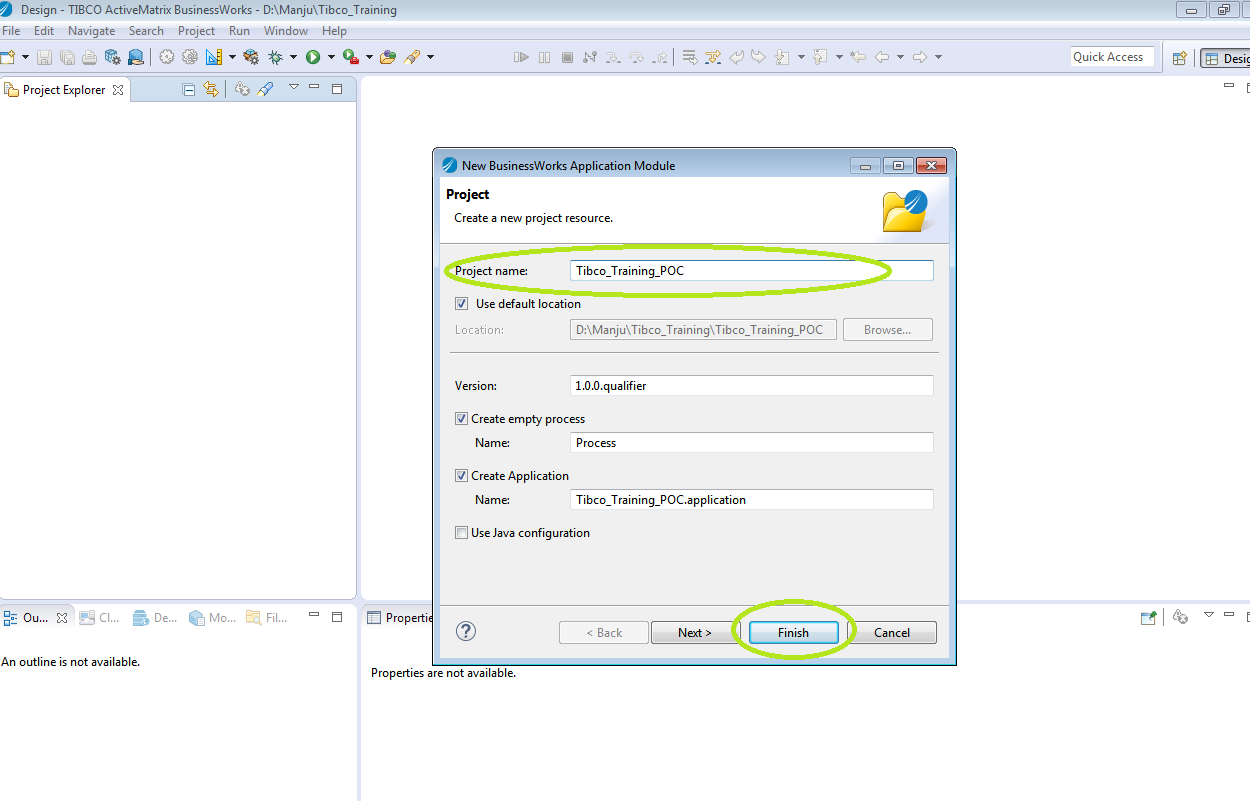
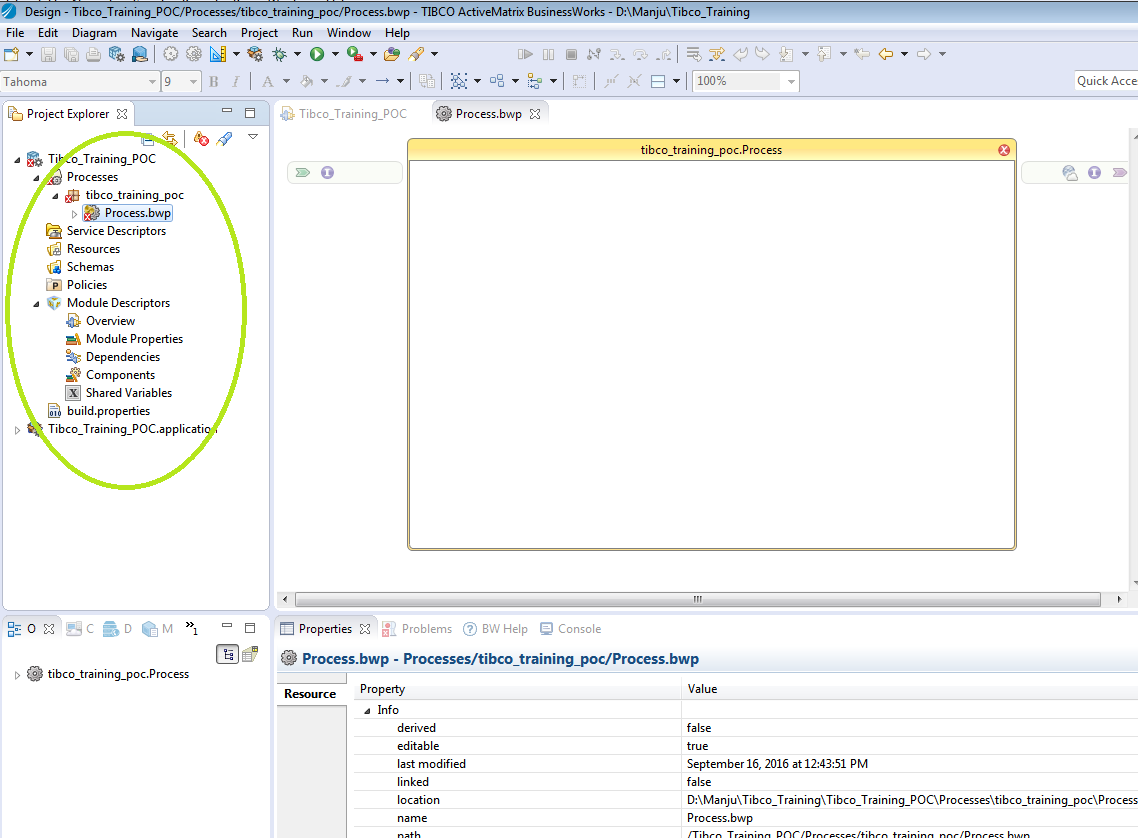


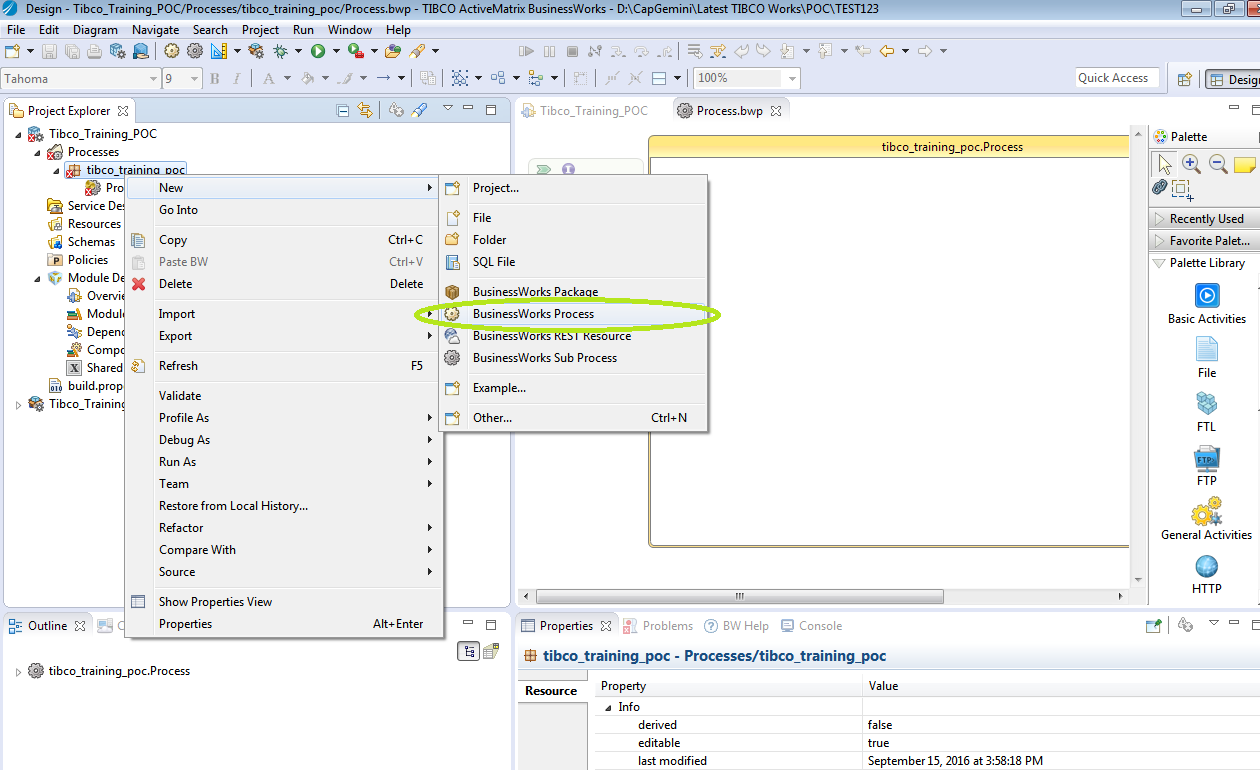
* After launching the designer refer the below screenshot which will ask us for the project workspace name and the directory where we want to create the workspace. Click on ok and proceed further. Refer the below screenshot.
* Once after providing the appropriate workspace name, empty workspace will look as below

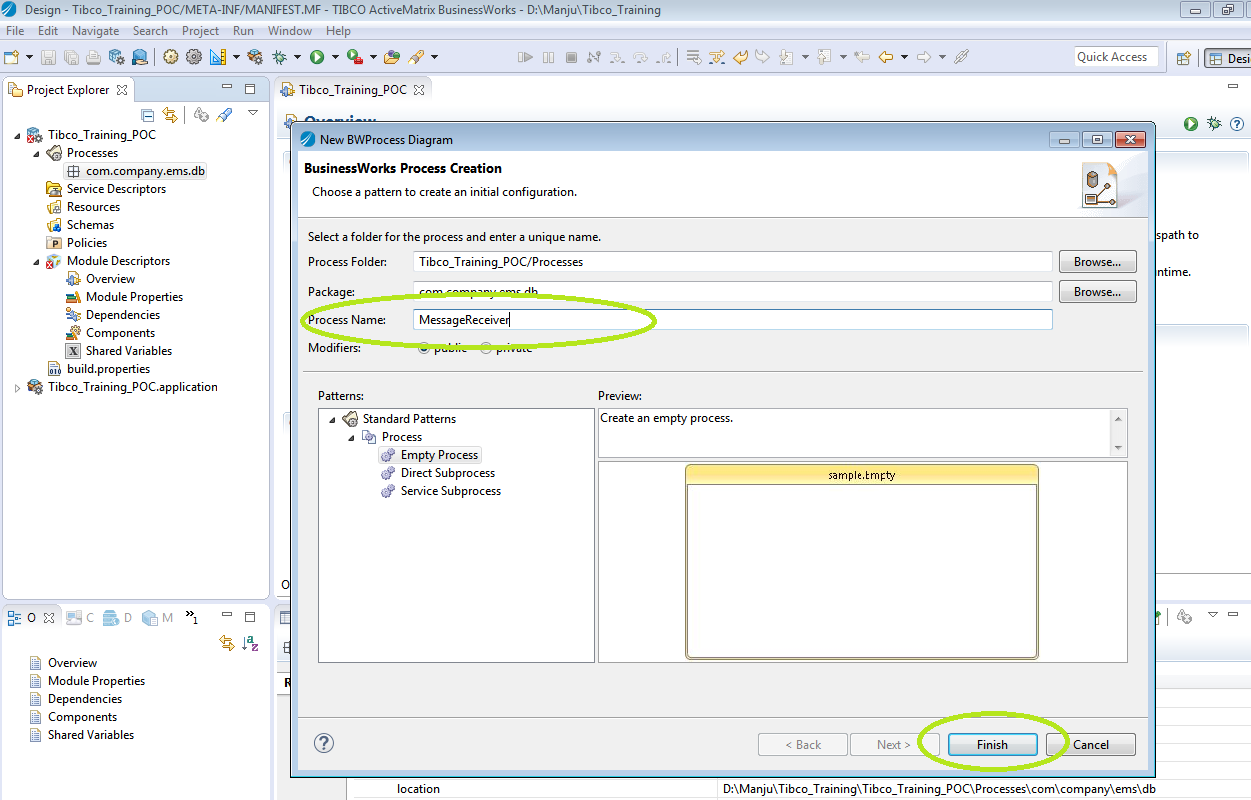


* In TIBCO Business Events Studio, click on File 🡪 New 🡪 Business Works Application Module to create a new module



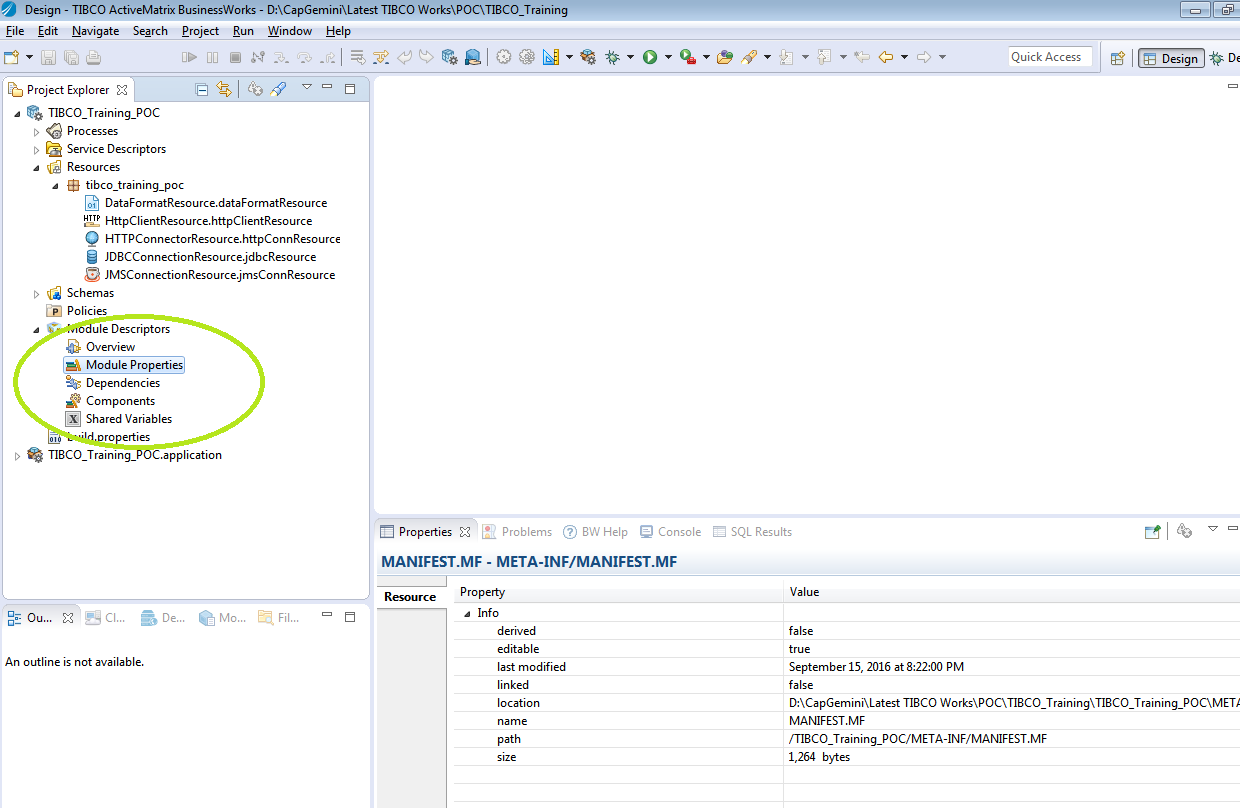
* Give the name to the module and click on finish as shown
* After adding the project name the template will be looking as below with one default process
* If required create the necessary packages and add the process to the created package

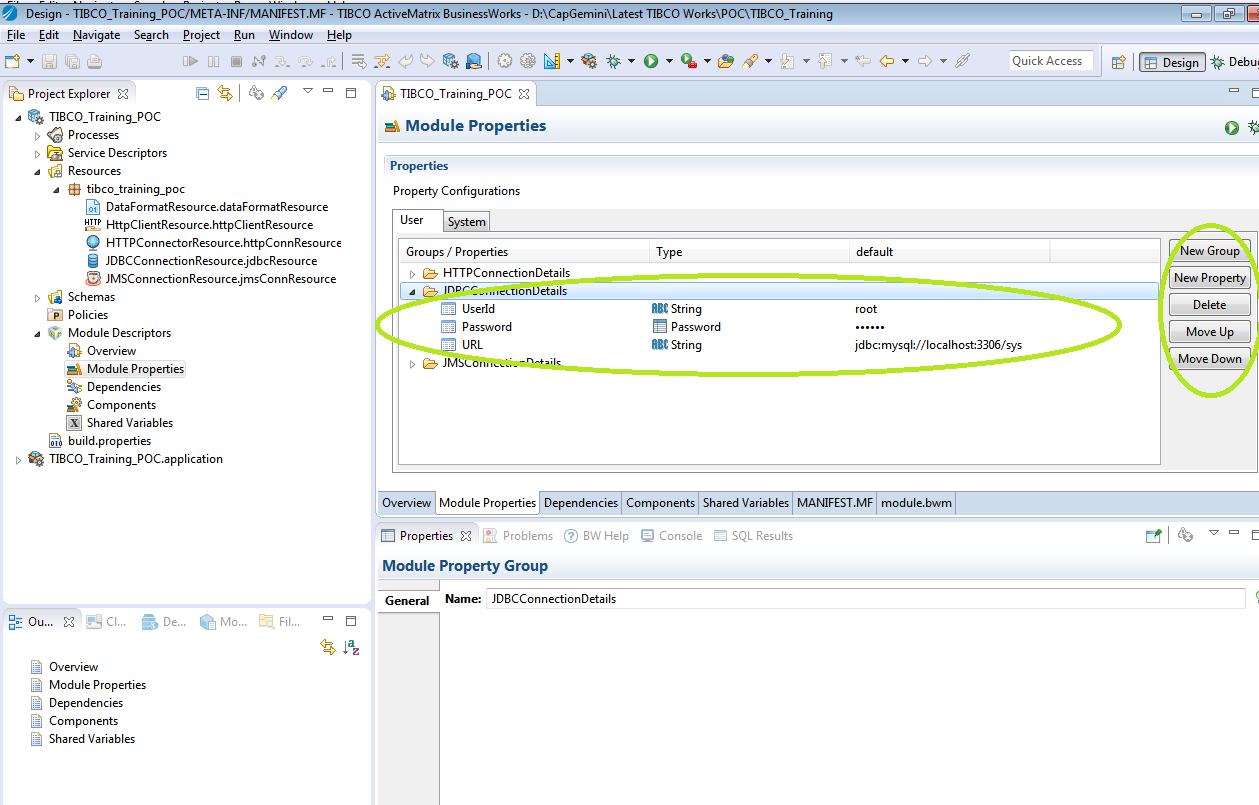
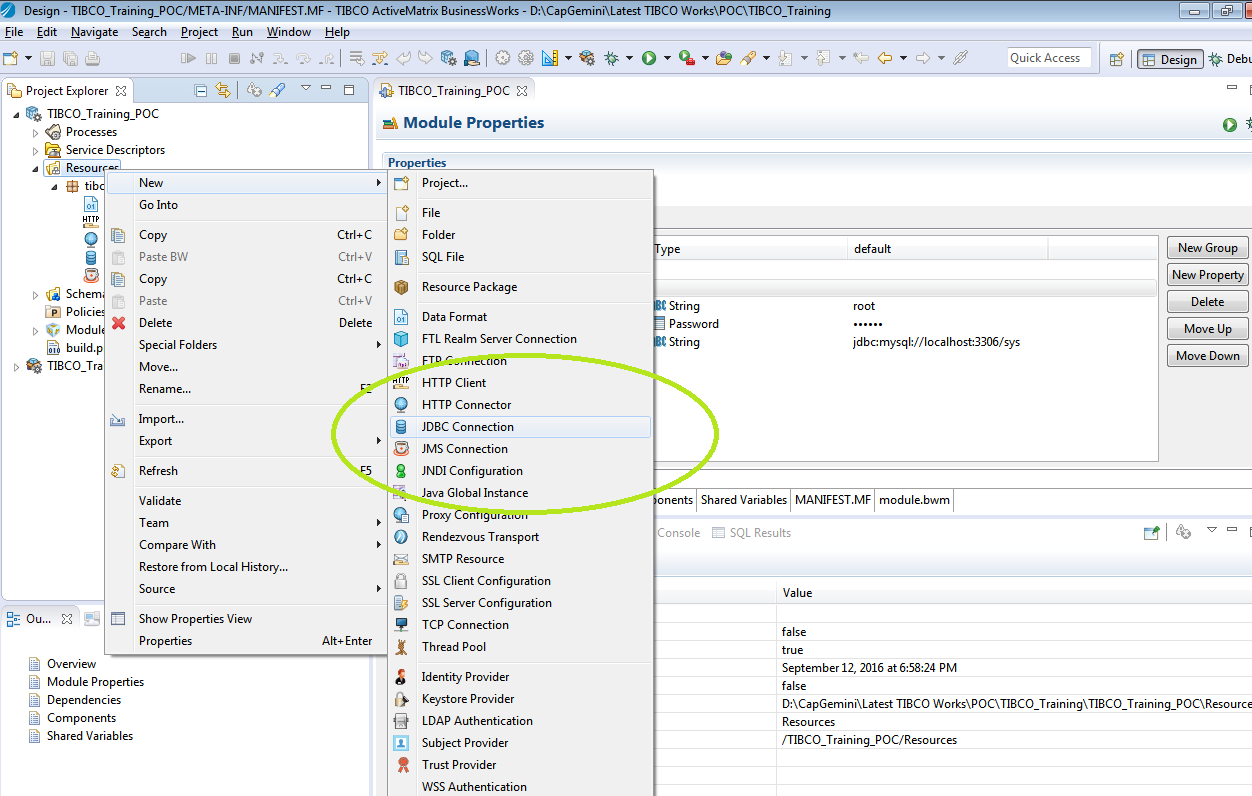


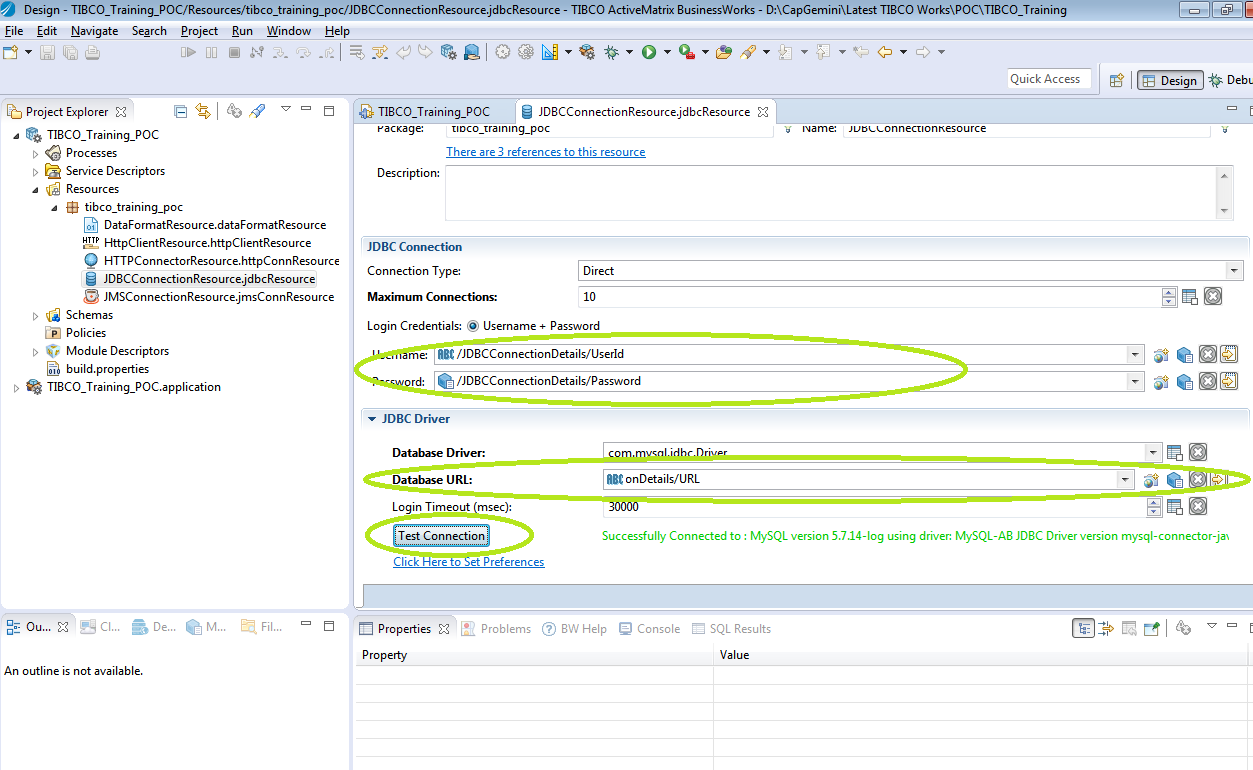
* Add the process to the created package. Click on finish
* The project template will be looking as below with process defined within it;



* Now we will create the shared connection details which are required to connect to the EMS and Database servers. Once after configuring test the connections successfully.
* For creating the shared connections we are referring to the global connections and to create this go for the module properties which is under module descriptors tab and double click on it as shown below

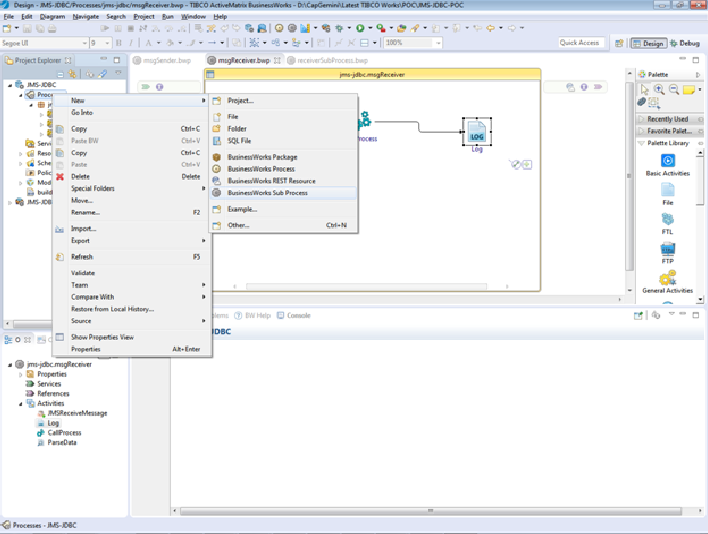


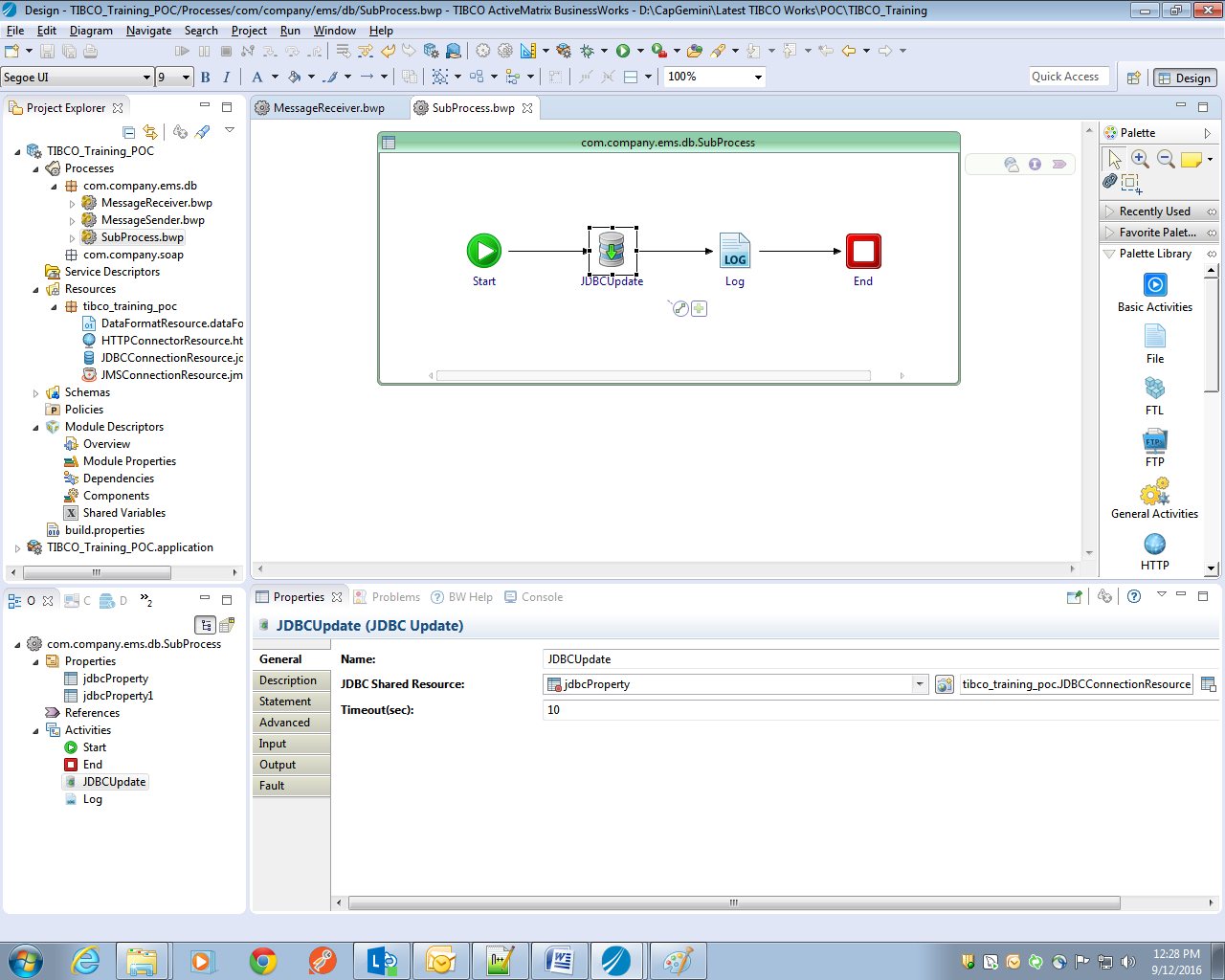
* After that provide all the necessary details to connect to the servers. For reference here we are showing the jdbc connection details defined;
* Now under resource tab create a shared resource details
* Now provide all the details referring to the jdbc\jms connections and test the connections successfully;
* Please refer the below screenshots;



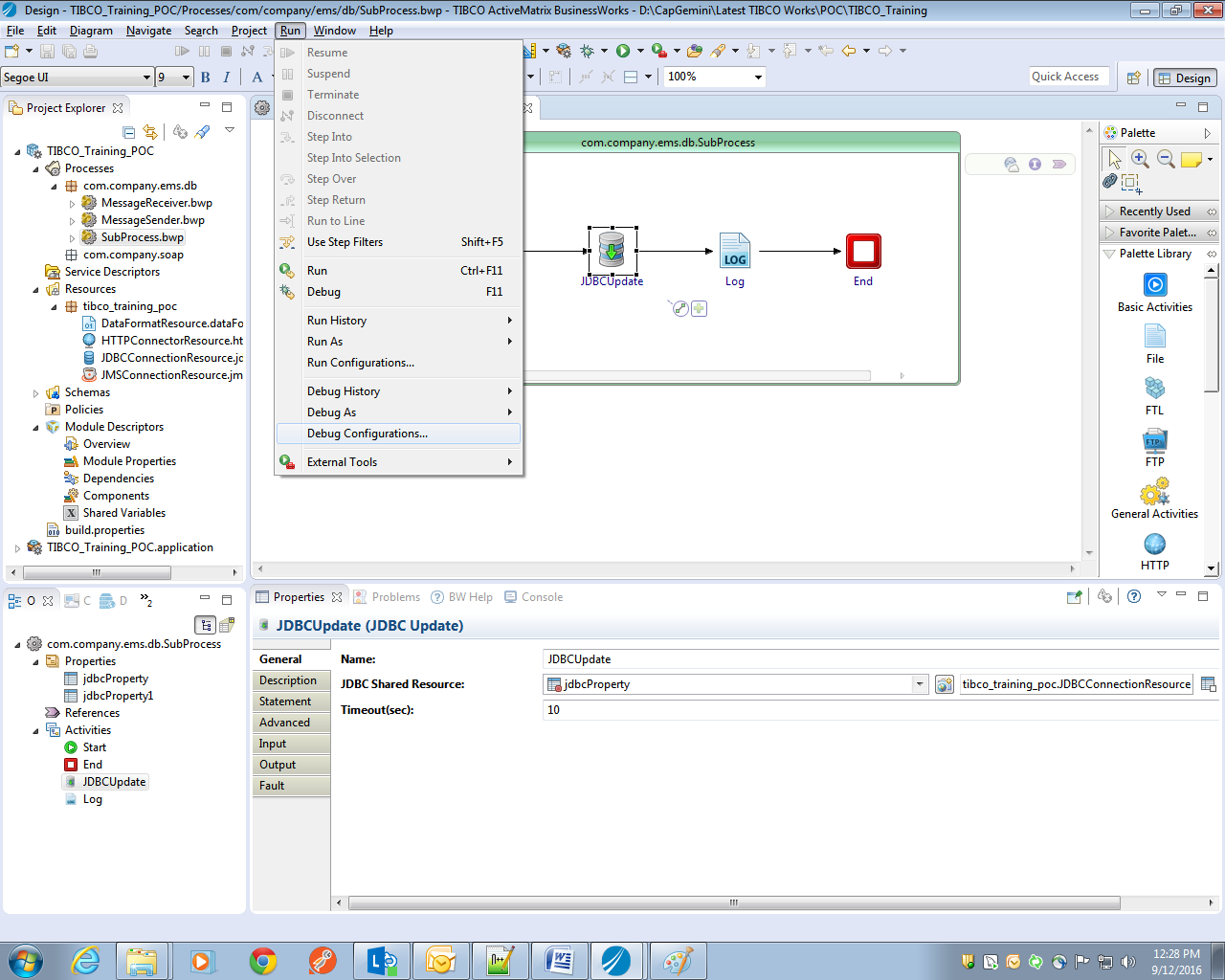
* Now let’s proceed further to create a sender process which will read a text content from particular file and then send it to the queue which we created earlier in our case(queue: Capgemini.TIBCOTraining.Exercise1.Queue).
* Now as a part of receiver process we will be using the JmsReceiveMessage activity to receive the msg from the queue Capgemini.TIBCOTraining.Exercise1.Queue to which we have sent the message in our sender process.
* Once after receiving the message successfully we will be transforming the data to match with the database table structure. For this case we will be using the ParseData activity which will convert the comma separated values into the defined structure.
* Now we will call the sub-process which will read the data and insert it into the database.
* The below screenshot depicts the above scenario;

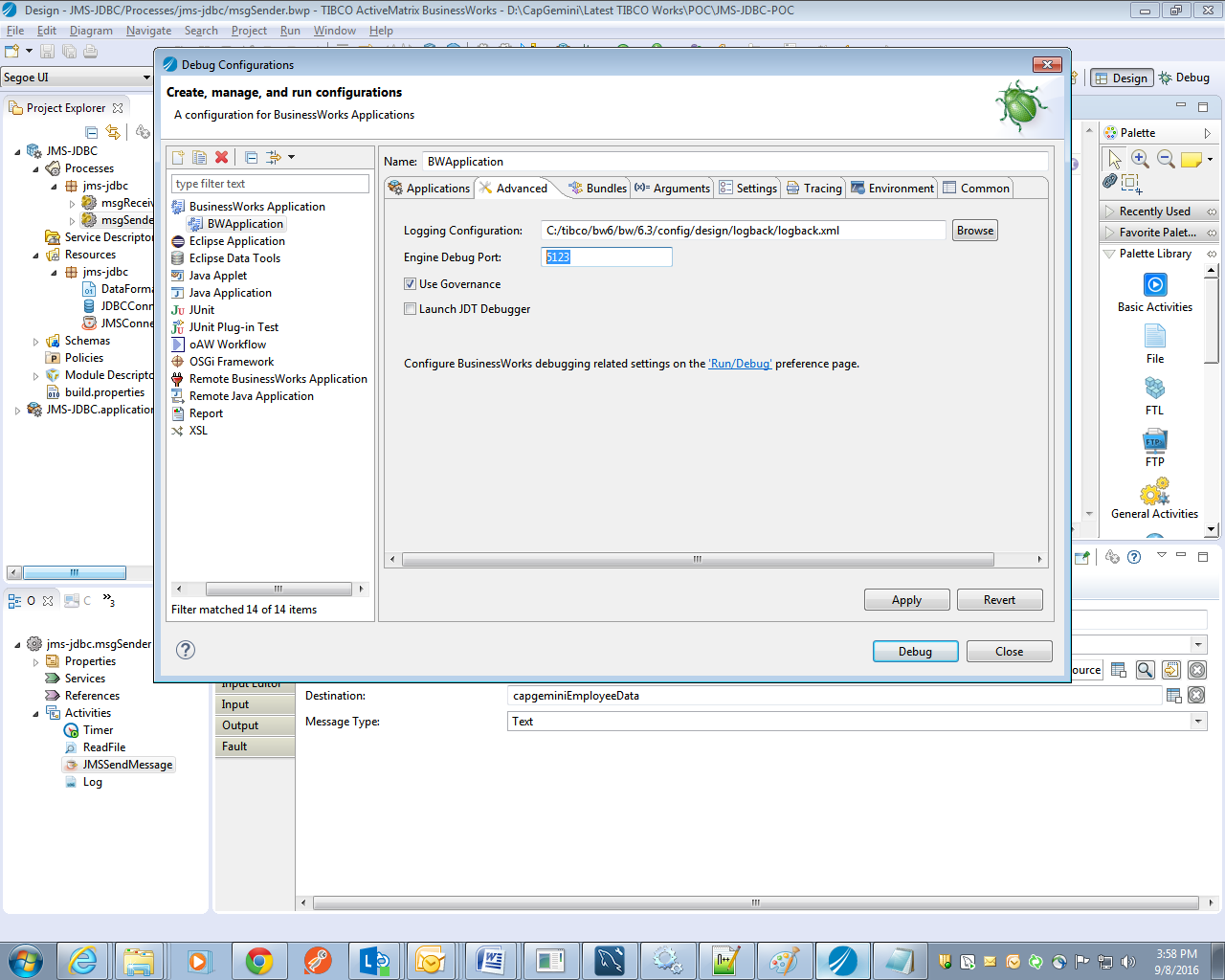


* For inserting the values into the database we will be required to configure the shared jdbc connection in the called process, which we created in early steps.
* For creating the sub-process right click on the Process and select for Business works sub-process as shown below;
* Map all the required details from output to the inputs and validate the project. Once after successful validation go ahead for testing the application.
* The sub-process will be looking as below;



### Testing

* For testing the applications go for run tab which is at the right top of the window and go for debug the configurations.   
  
* Once after that go for the advanced tab of the debug configurations and change the port number which will be available for use.



* Once it is done apply the changes and click on debug button.
* The application will get triggered and the logs will be written on the console.
* Here I will be reading a file which consists of the following content

## Lab Exercise-2

### Problem Definition

Employee Data Management System (EDMS) application exposes services to the other Capgemini applications. One of the Capgemini HR applications consumes this service to create new employee record in EDMS.

This solution should build a SOAP service to create an employee record in database.

### Solution Overview

This can configured either by using HTTP as a transport or JMS as a transport. In our case we will be using an HTTP transport communication to the service.

**Consumer**

**Server**

Employee service

**MySQL**

Request

Over HTTP

Response

Topics Covered: Project Creation, Invoke, SOAP service

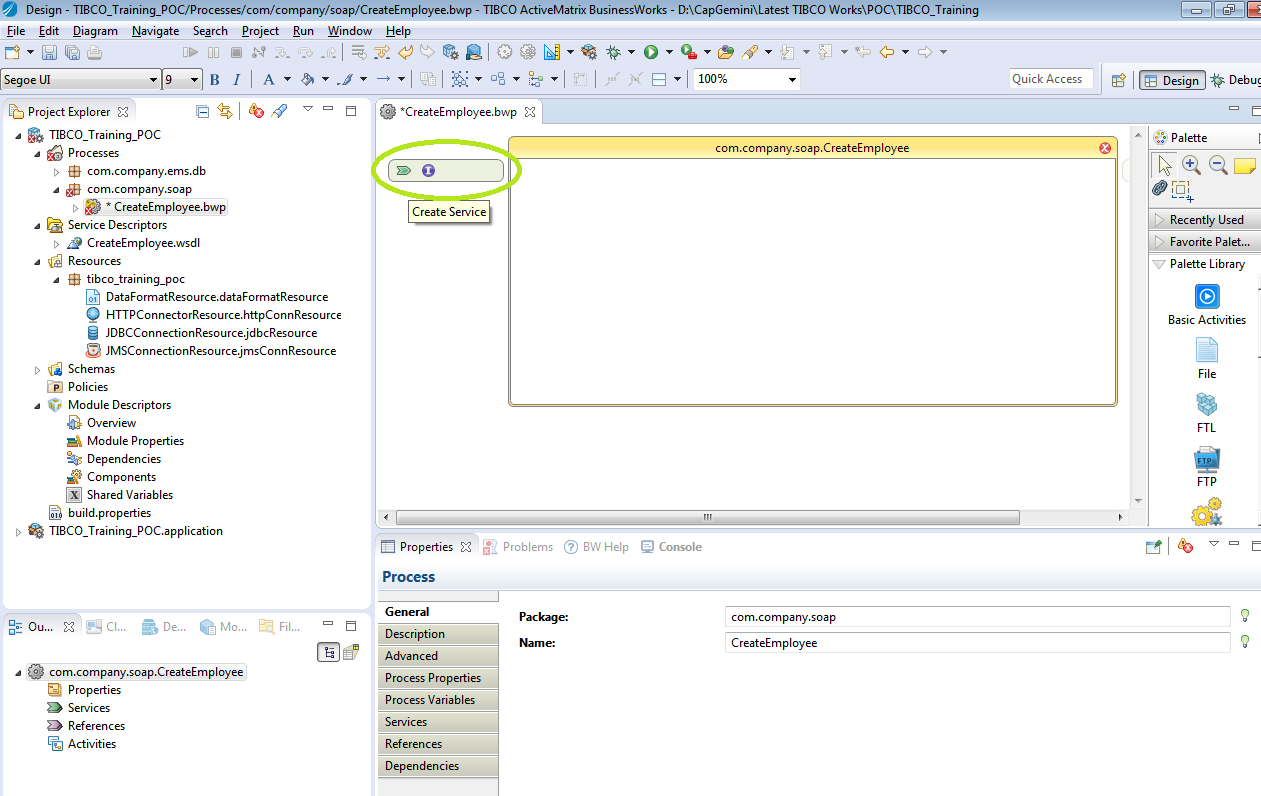
### Required Artifacts

SOAP UI software can be downloaded from the following **https://www.soapui.org/**

Here we will be using our same Employee schema which we have used in our previous applications.

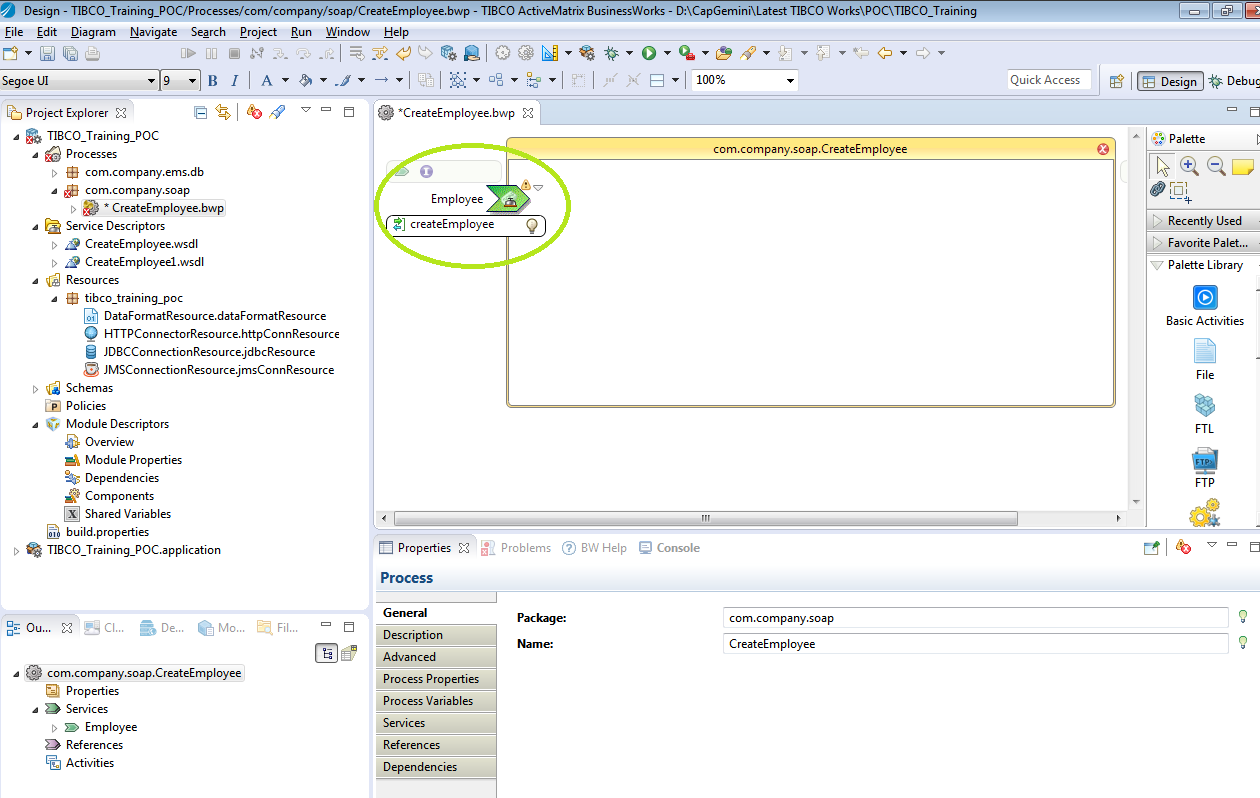
### Development Steps

* After successful creation of JMS and JDBC applications let's look into the creation of SOAP POC without any security.
* For this purpose we have created our own package **com.company.soap** where we will create a soap process.
* Create a regular Business Works process which we have done in our previous cases.
* Refer the below screenshot to create a service and service level operations.

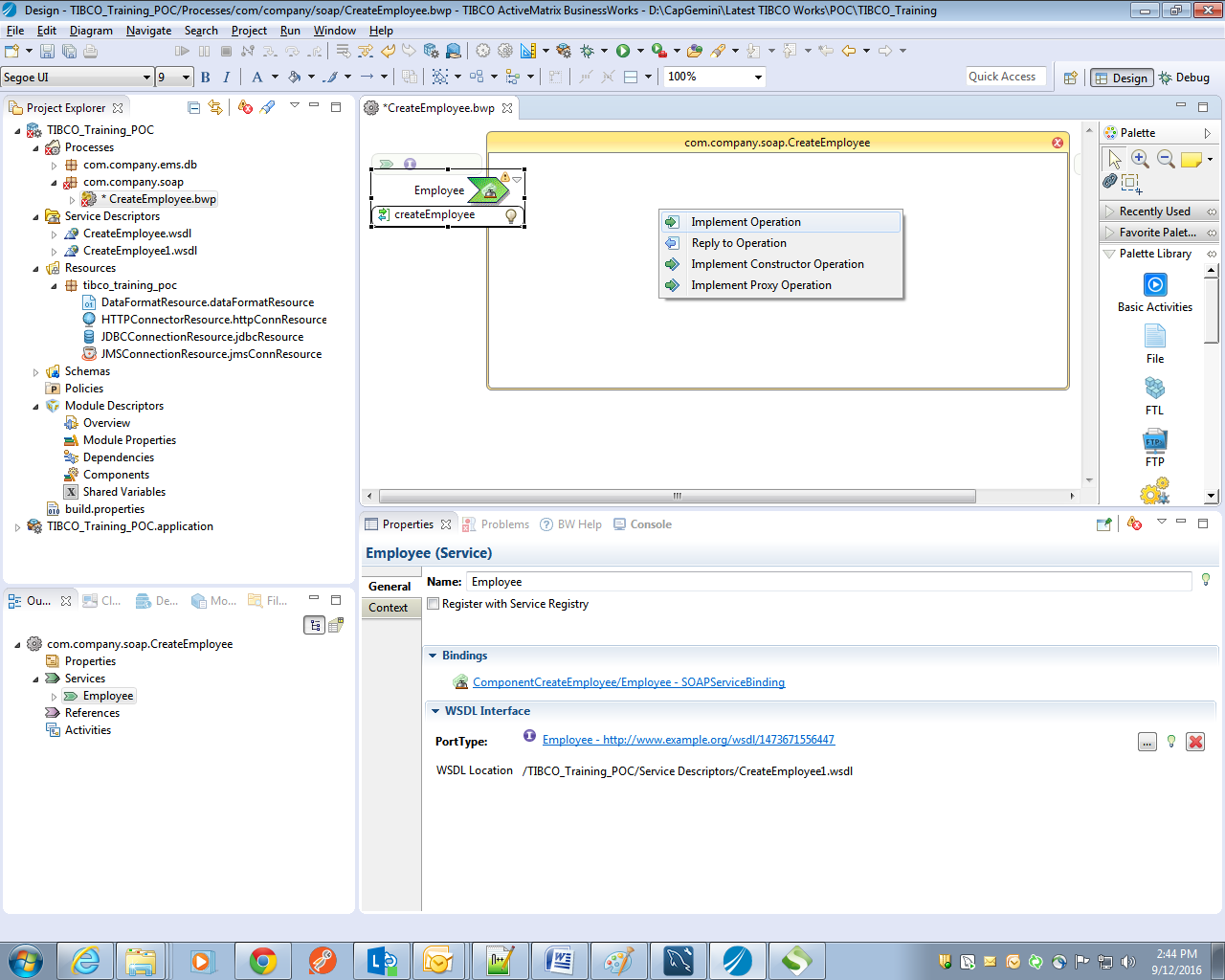


*In this case we will be creating only one service where we will be calling the sub-process to add the employee into the database.*

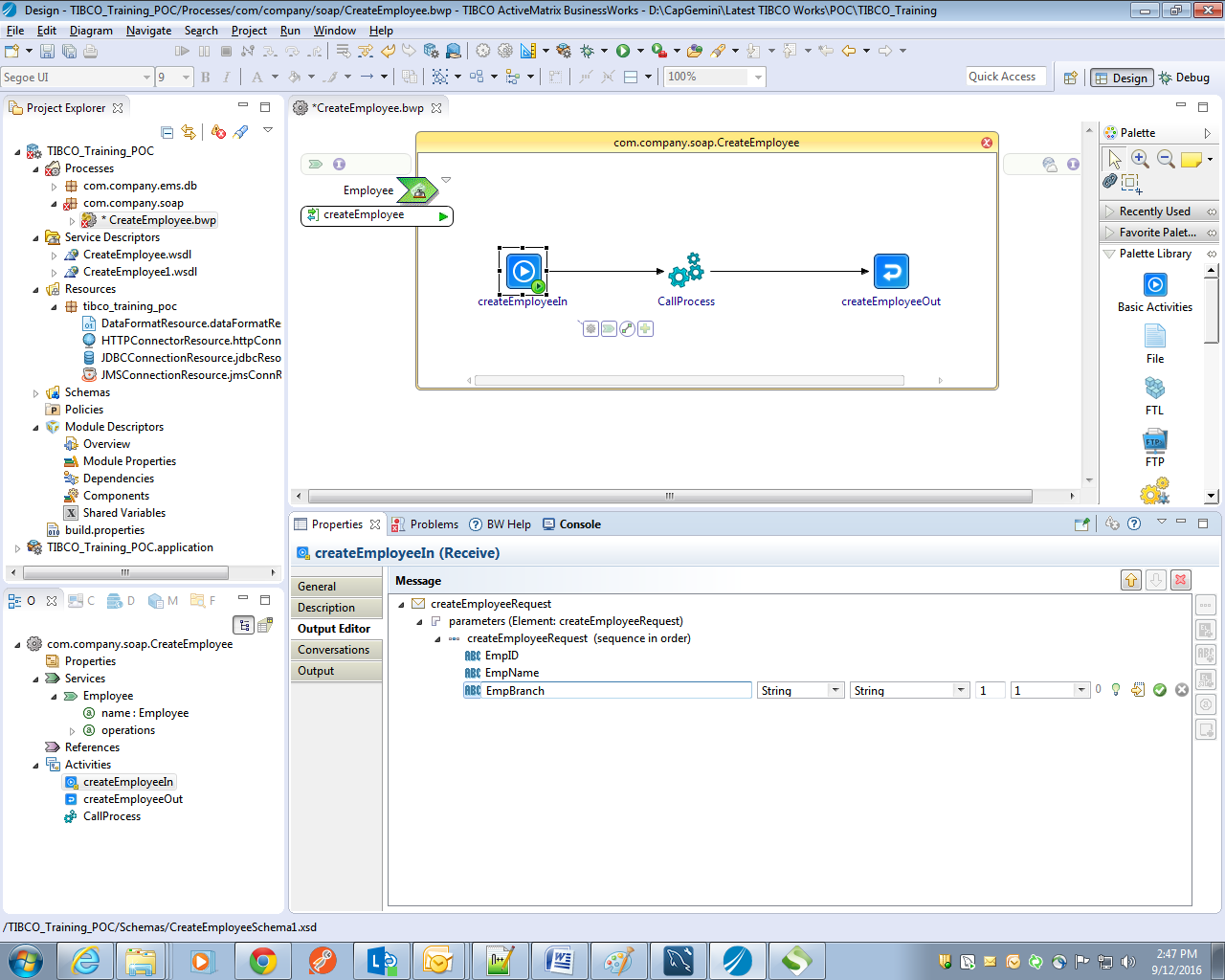
* Here, Employee is the service name and createEmployee is the operation name which we created;



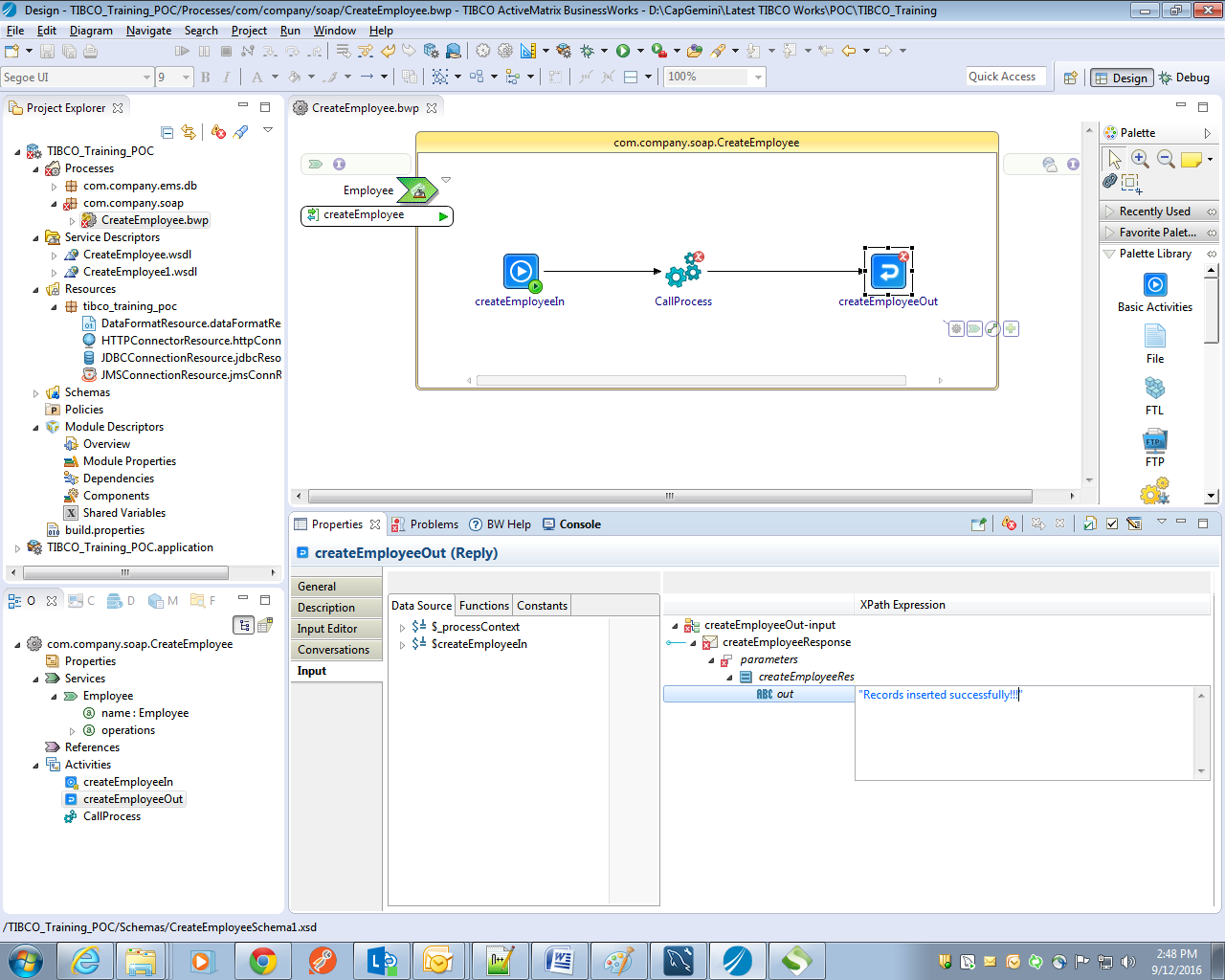
* Now go ahead and implement the operation

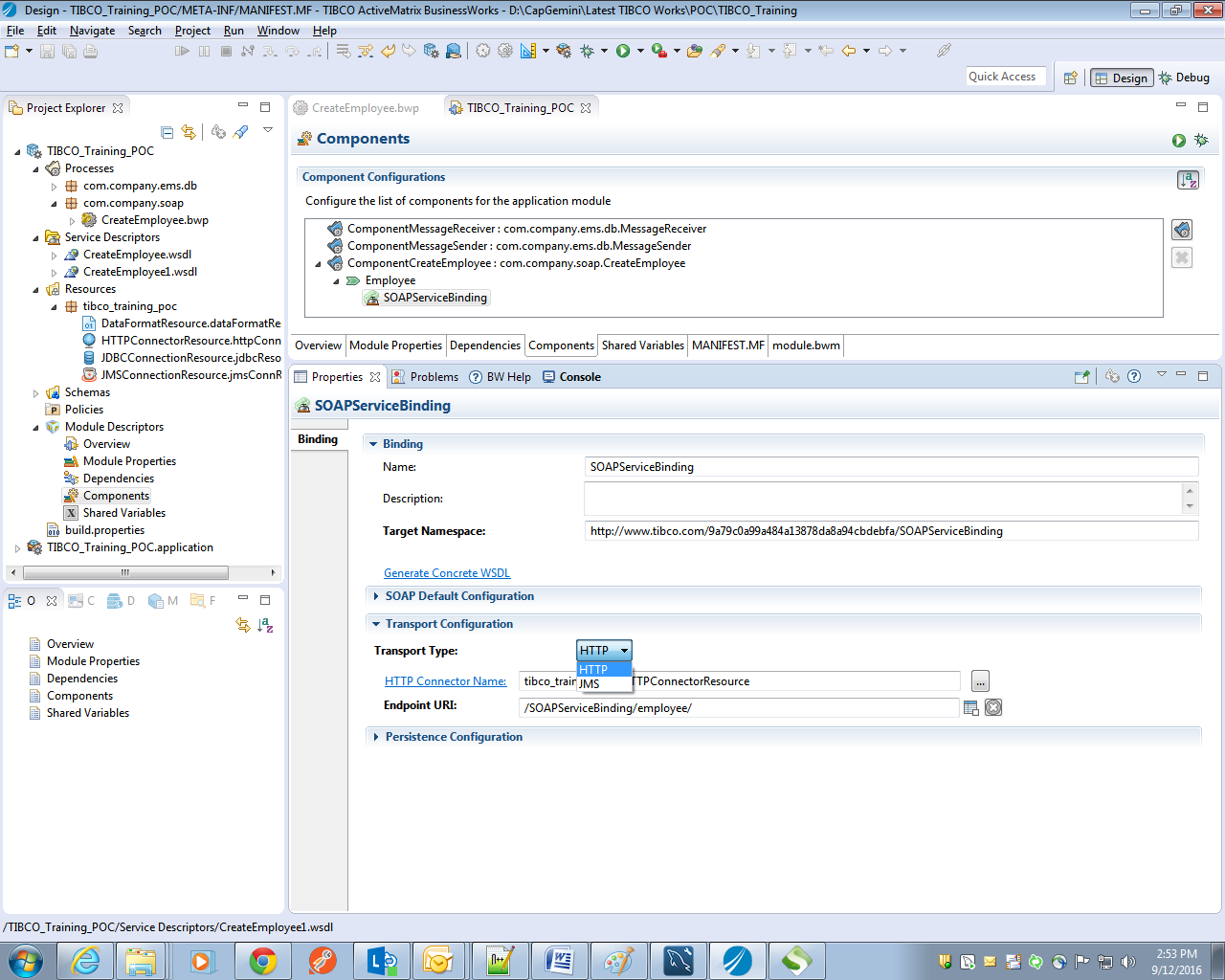


* Since we are going to insert the record using soap we will create our own schema where we will pass the required values to the receive activity;



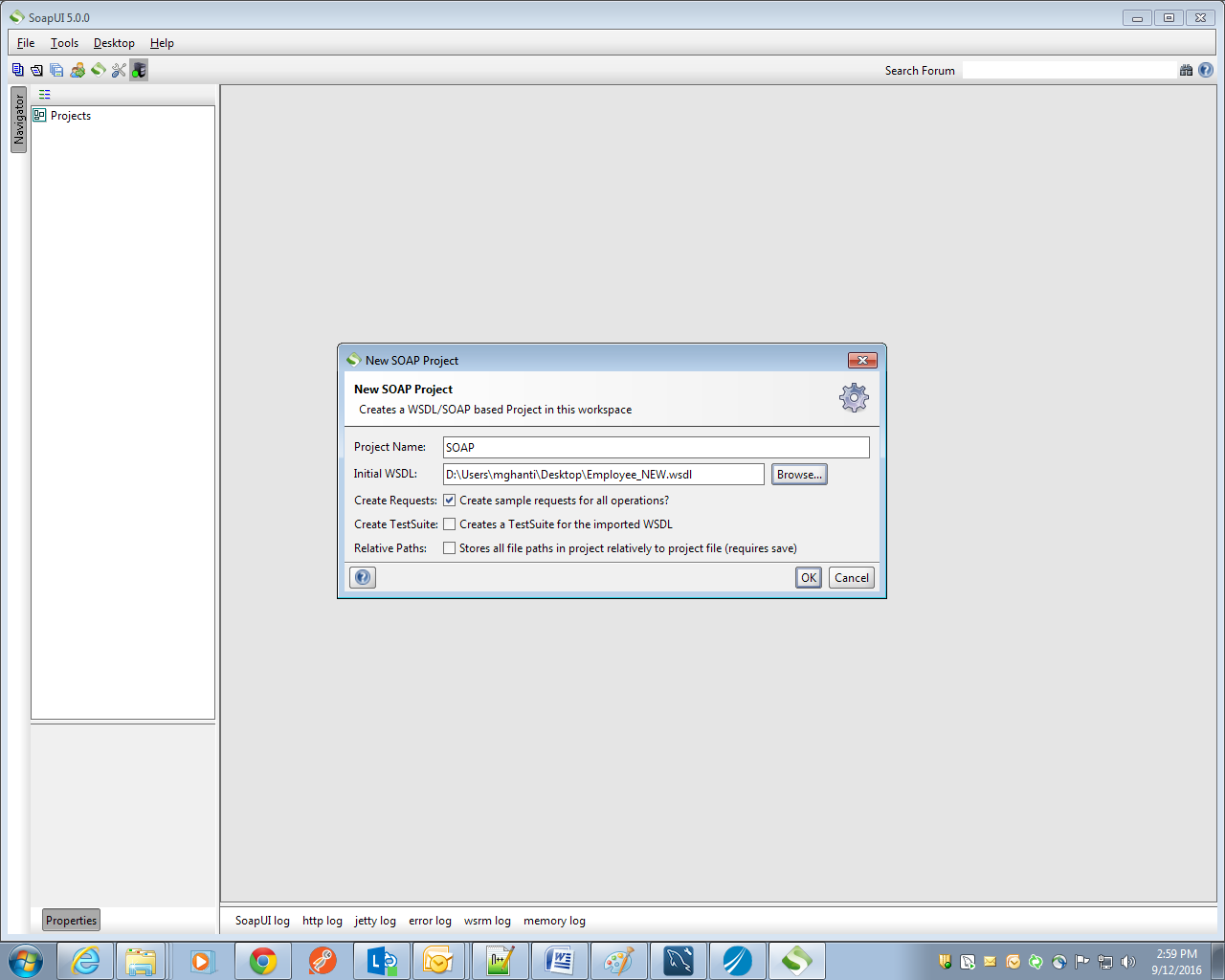
* Now map all the input fields to the call process.
* Once after successful completion of SOAP request response we will write an info message saying “Records have been inserted successfully!!!” as shown below;



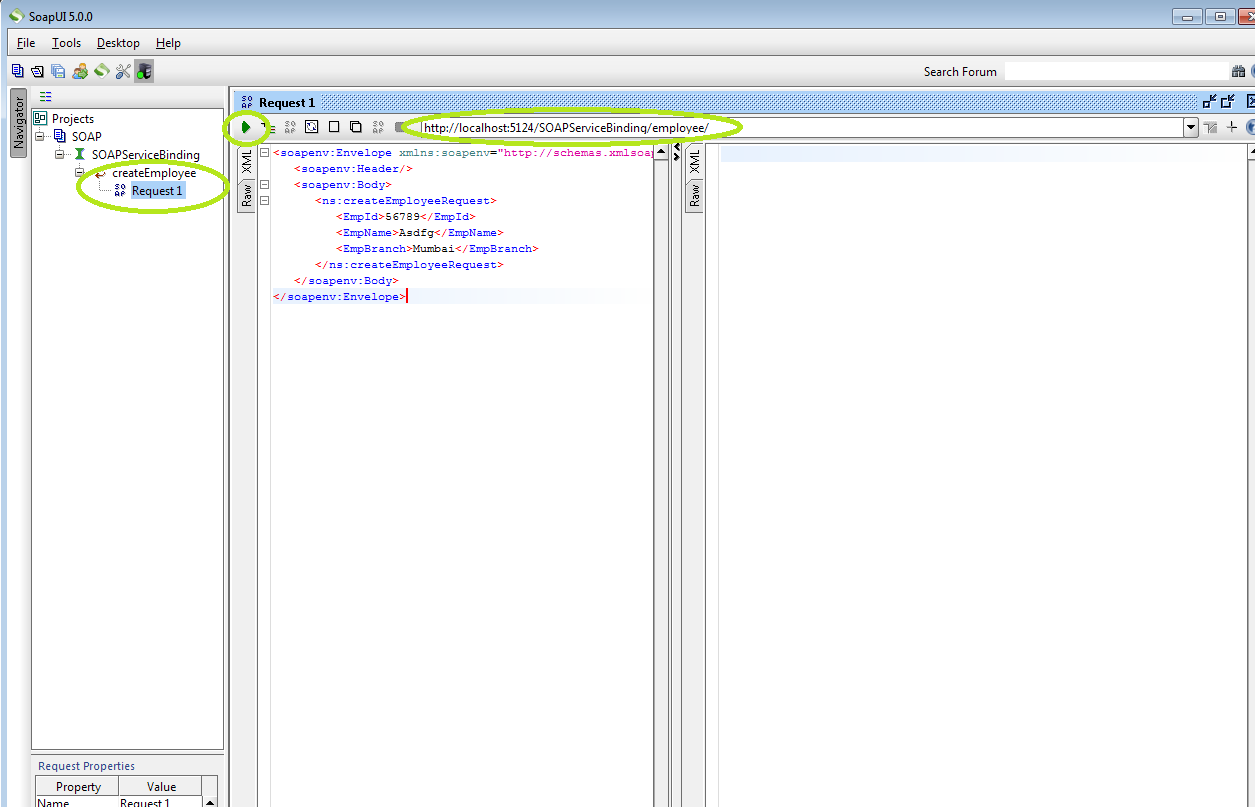
* Now we will go ahead to add the binding to the process where it will be a transport we will be using to connect to the service. In this case we have will be using HTTP as a transport. Under module properties we have components tab which will ask to add the service binding to our process which we created earlier. 
* Now generate the concrete WSDL which is under Binding tab and we will be using this WSDL for our testing purpose.

### Testing

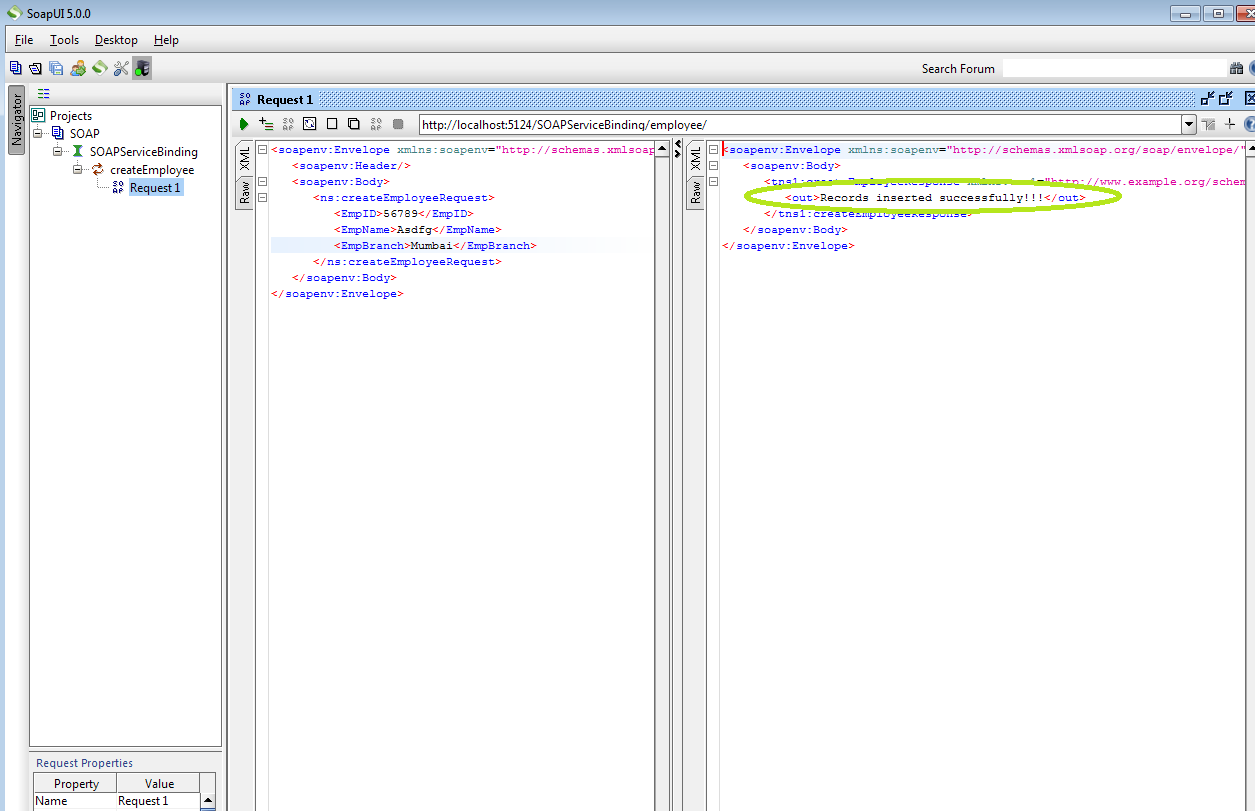
* Start the service process which we have done before for our JMS-JDBC application.
* Once after that we need to test this SOAP application where we need to send an HTTP request. Here we will be using the SOAP UI Tester software. We can get the trail version of this from the internet.
* Create a new project from the file and browse for the WSDL which we created in our earlier development phase;



* Once after hitting the operation name it will ask for the input request to the service. Provide the required details



* Once after that hit the URL with the request. We can see the response from the service which will give us the success response.



* Now we can see that our process has been triggered successfully.
* 

## Lab Exercise-3

### Problem Definition

Employee Data Management System (EDMS) application exposes resources to the other Capgemini applications. One of the Capgemini HR applications consumes this resource to create new employee record in EDMS.

This solution should build a REST service to create an employee record in database.

### Solution Overview

This can be achieved by creating a REST service over HTTP and expose it.

**Consumer**

**Server**

Employee service

**MySQL**

Request

Over HTTP

Response

Topics Covered: Project Creation, REST service, JSON, Swagger

### Required Artifacts

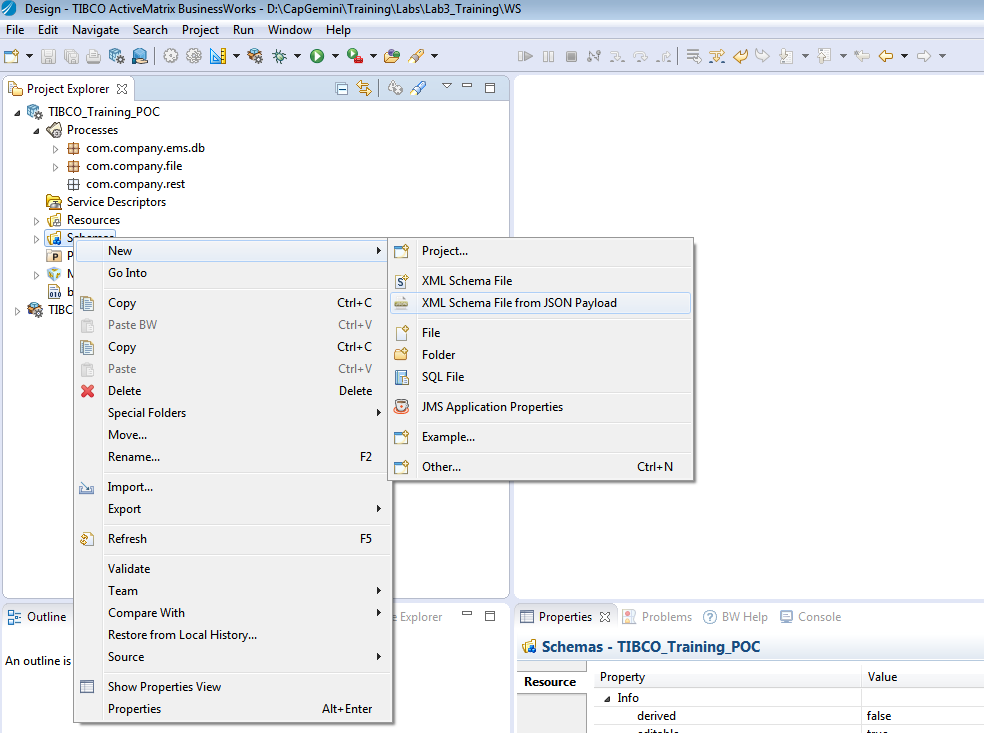
We will use attached JSON schema:



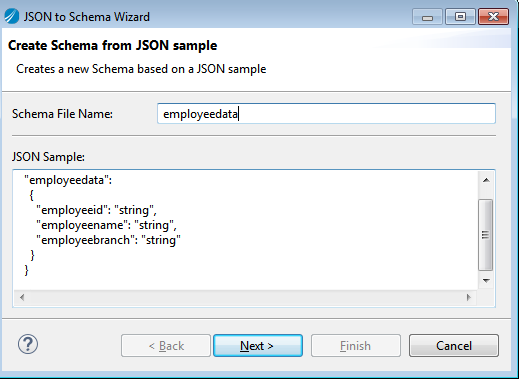
Here we will be using our same Employee schema which we have used in our previous applications.

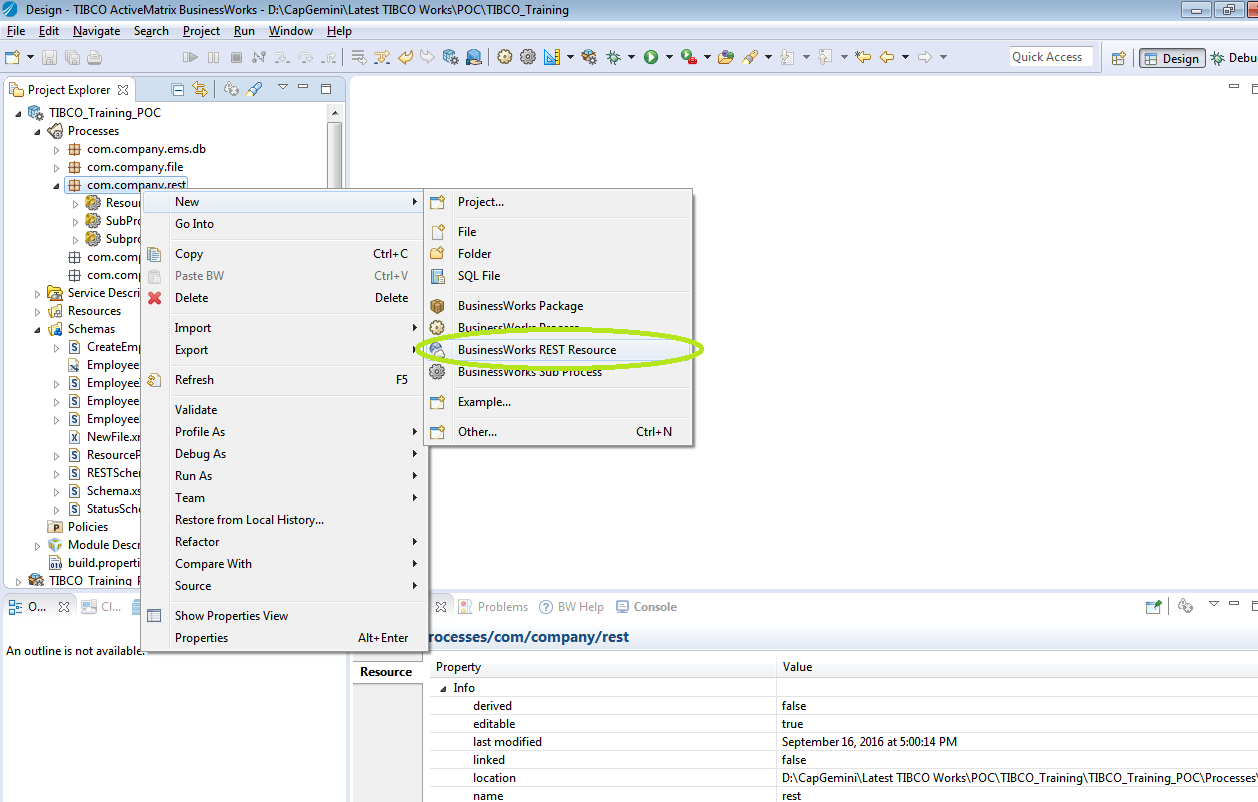
### Development Steps

* After successful creation of SOAP service and references applications let's look into the creation of REST POC.
* For this purpose we have created our own package **com.company.rest** where we will create a rest service.
* Refer the below screenshot to create a rest service and reference.
* Creating xml schema from JSON Payload involves below steps:



Enter JSON schema as below, click Next and then Finish



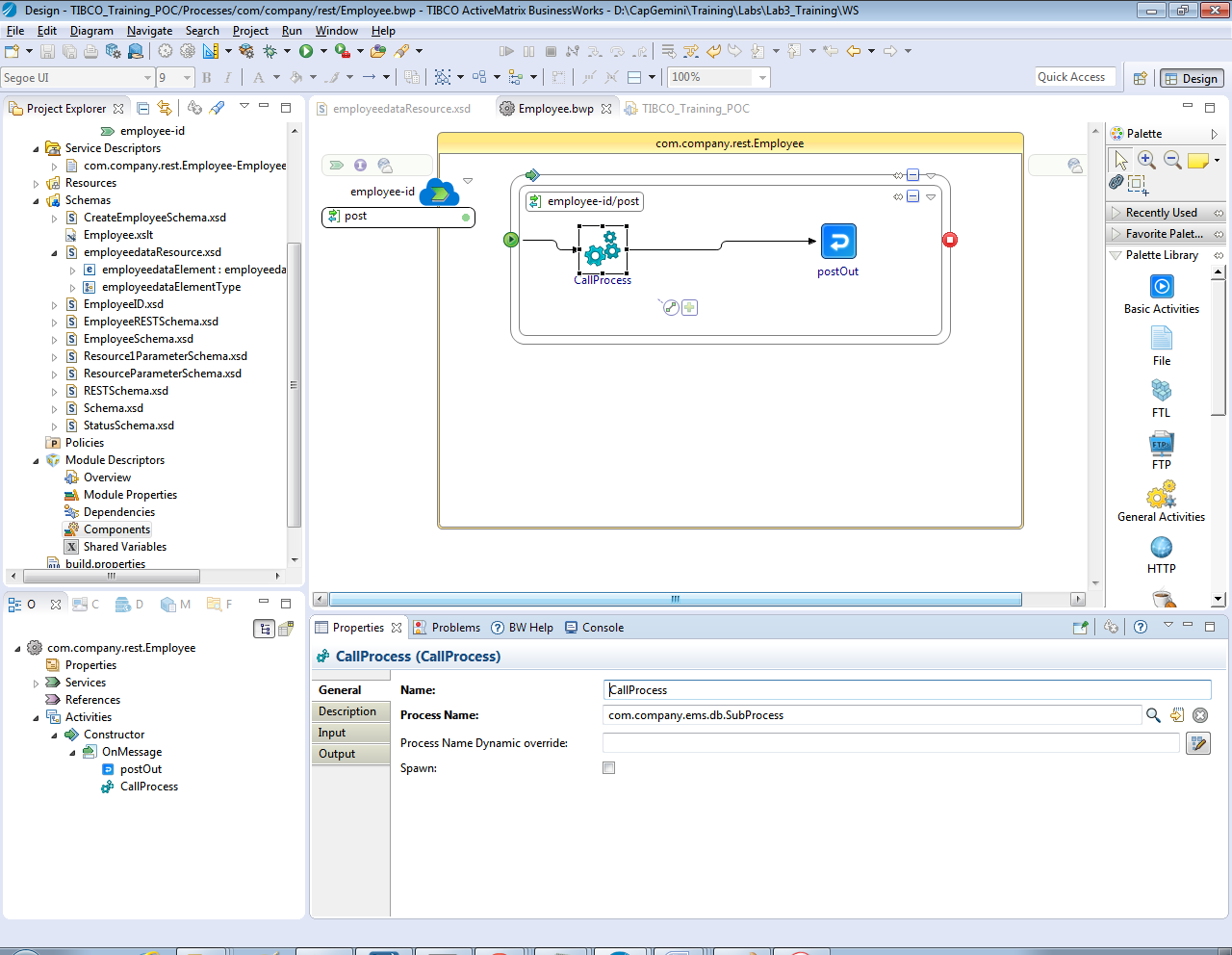
* Right click on the package and create a BuisnessWorks REST Resource:
* Provide appropriate Resource Name, path parameters to be passed in Resource Service Path, select the operations as show below:

# 

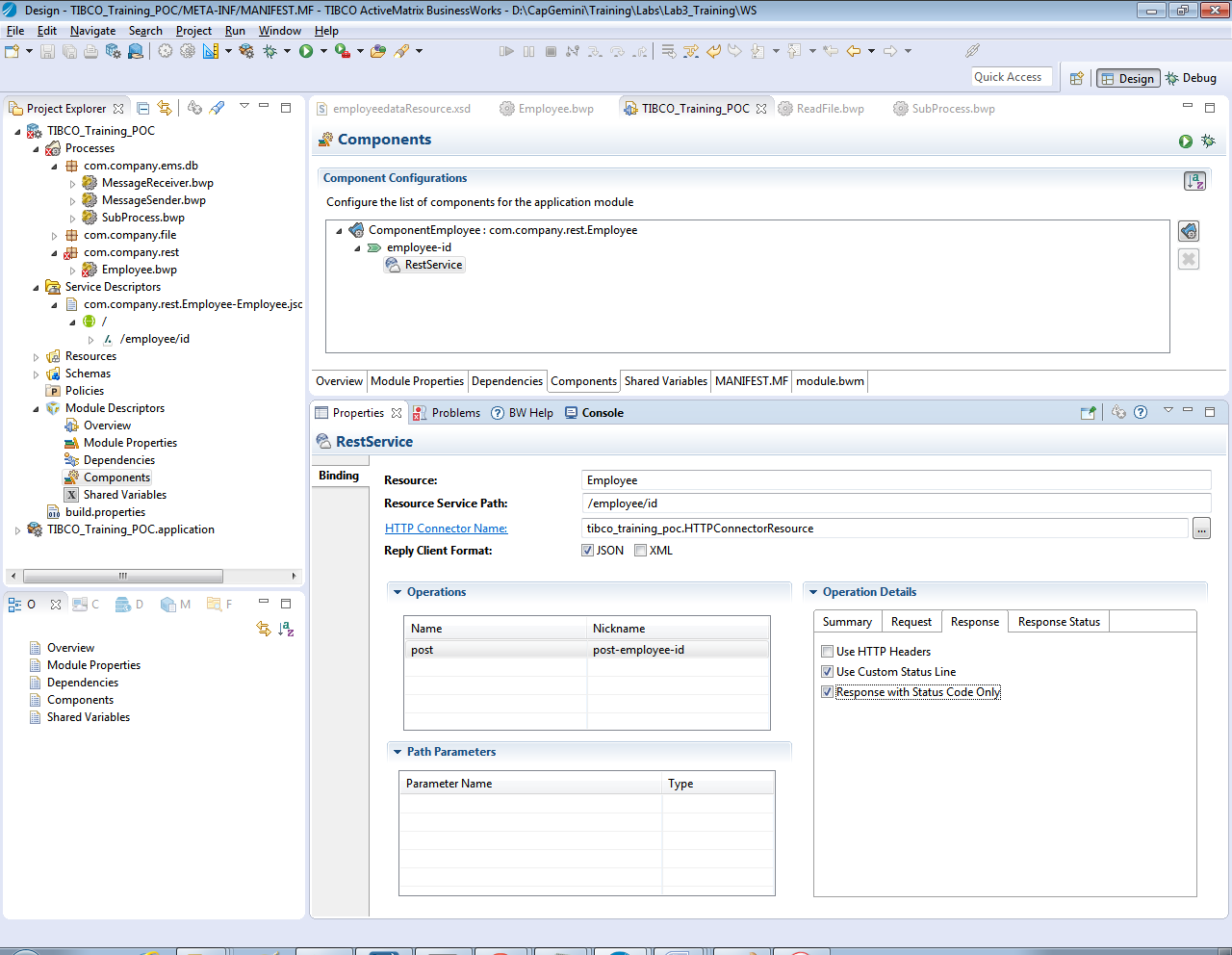
* Rest service will be created as shown below:

# 

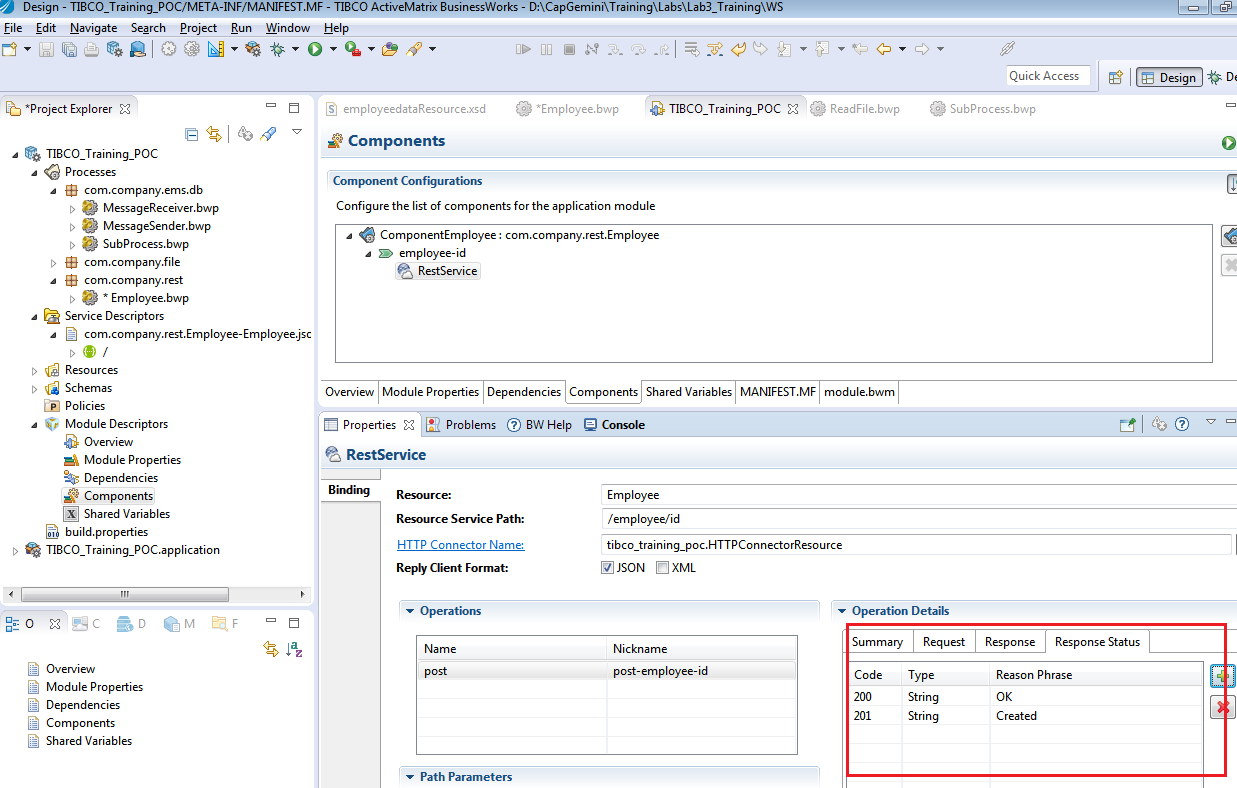
* Add a call process to call sub process created in Lab 1 to insert data into database:



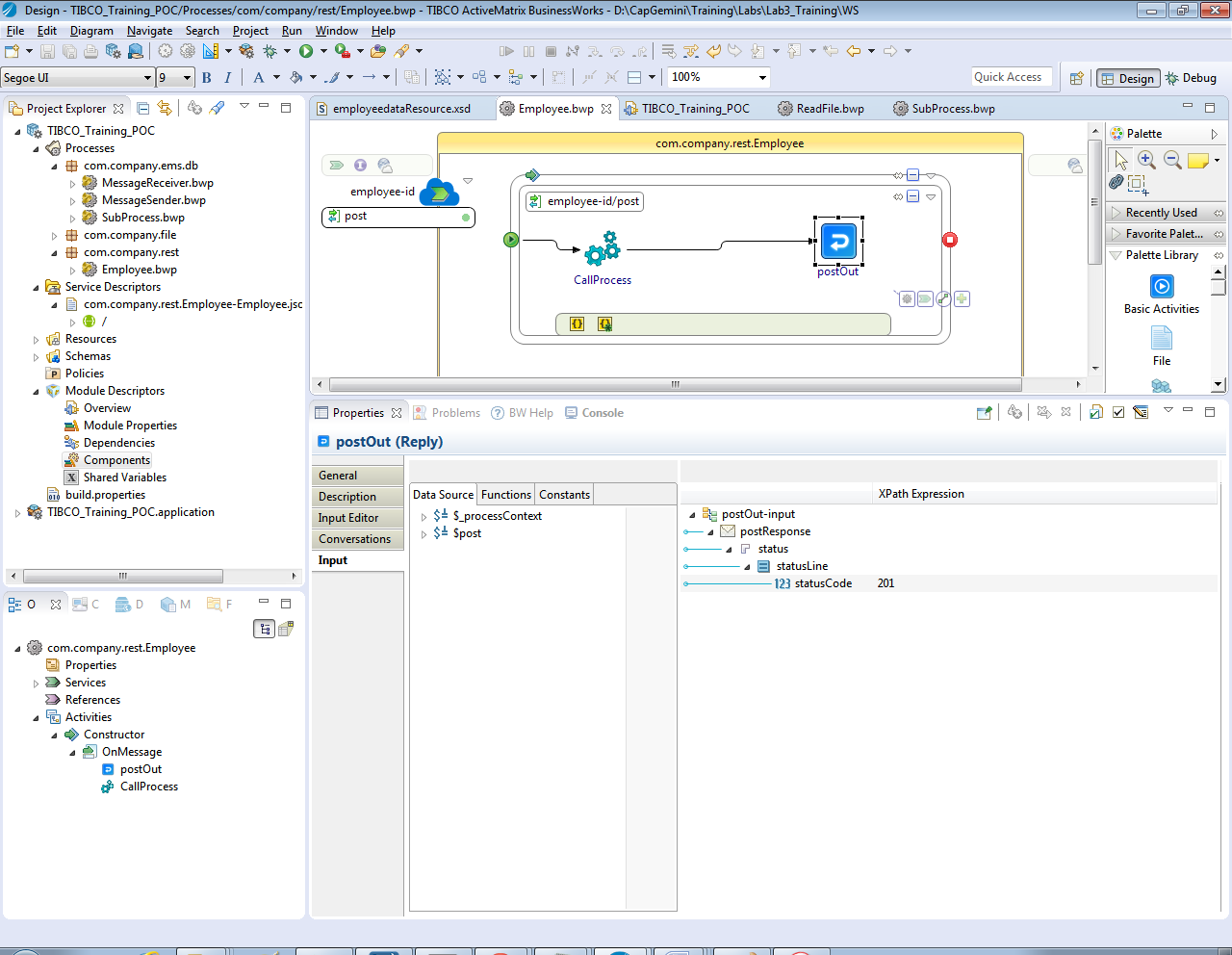
* Go to Components under Module Descriptors, select REST Binding and then select Response tab as shown below and select Response with status code only.



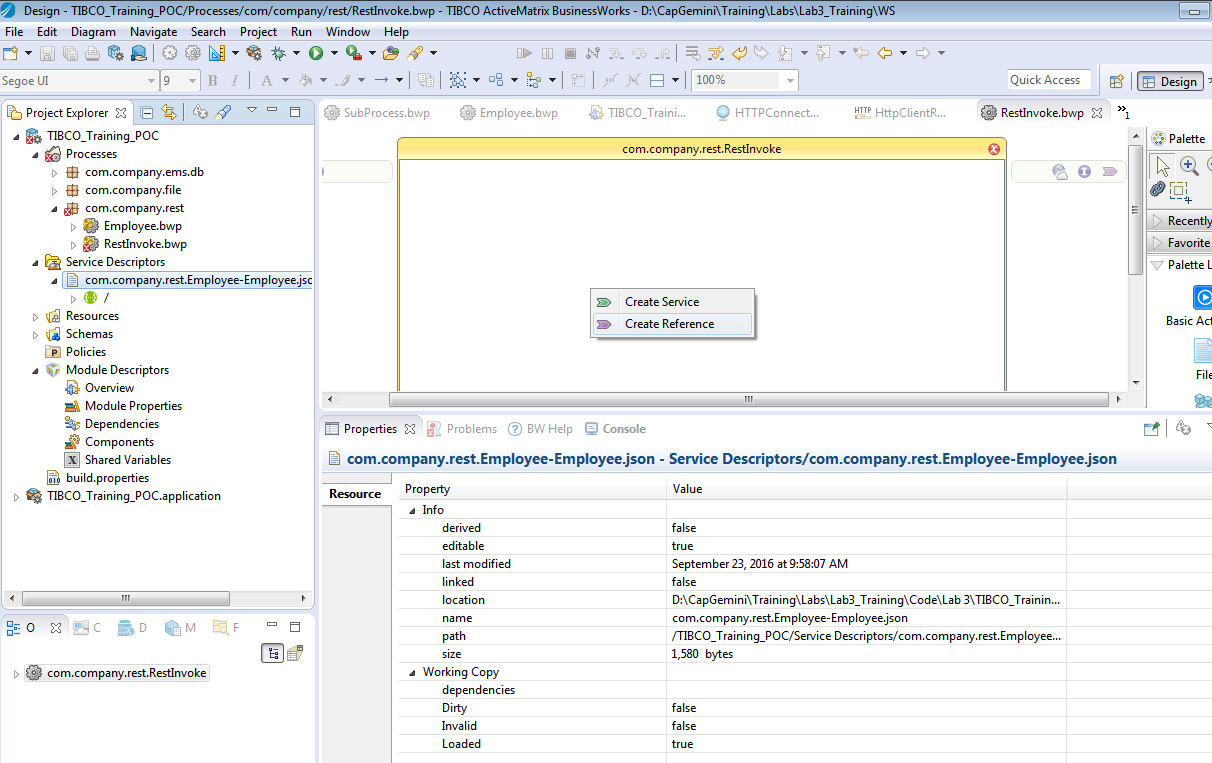
* Add Response codes which we will be using:



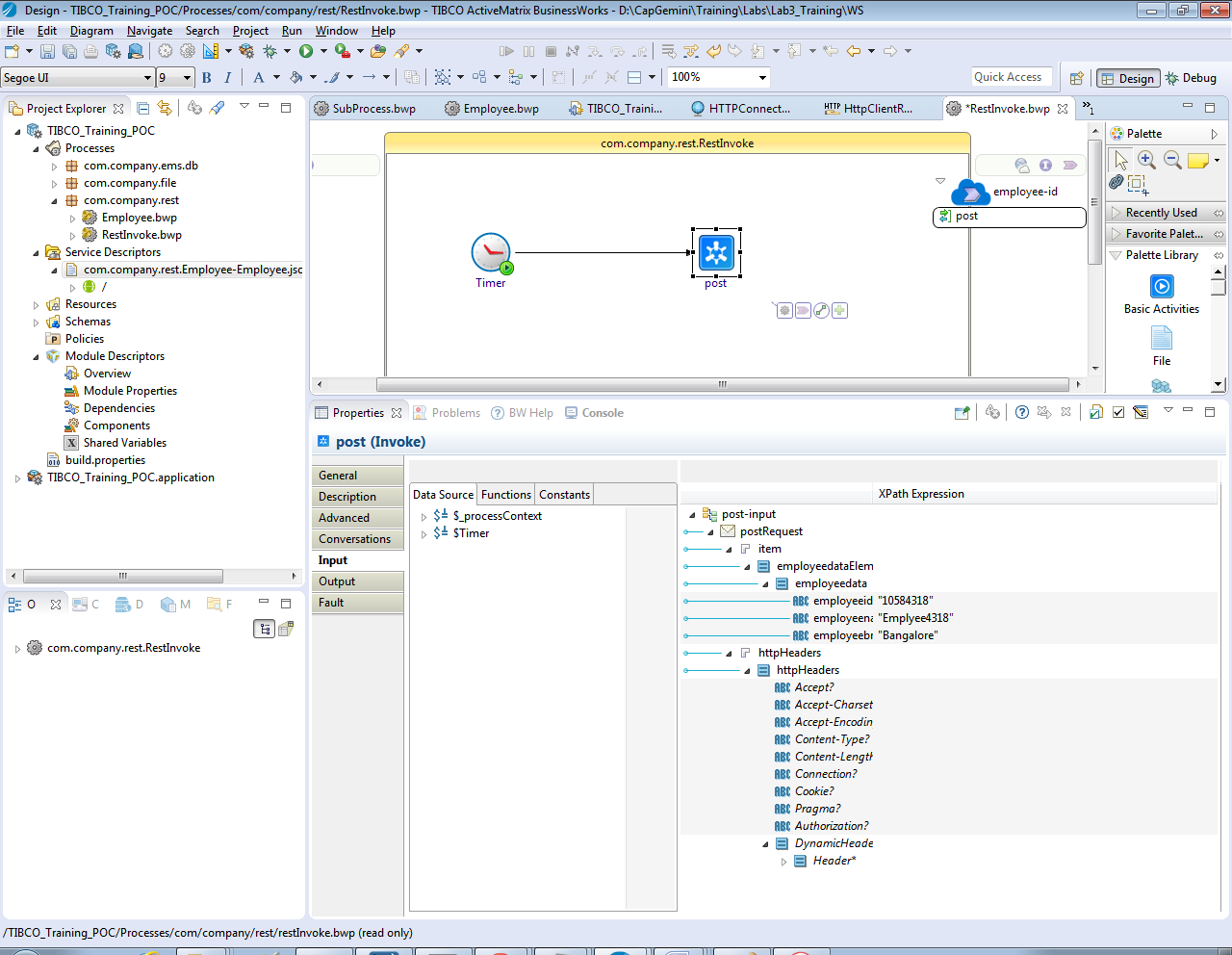
* Populate fields in postOut activity as shown below:



* Creating a REST reference to invoke REST service we created now.
  + To create a rest reference, create a new BuisnessWorks process, drag and drop swagger file created from service descriptors folder on the process and select Create reference.

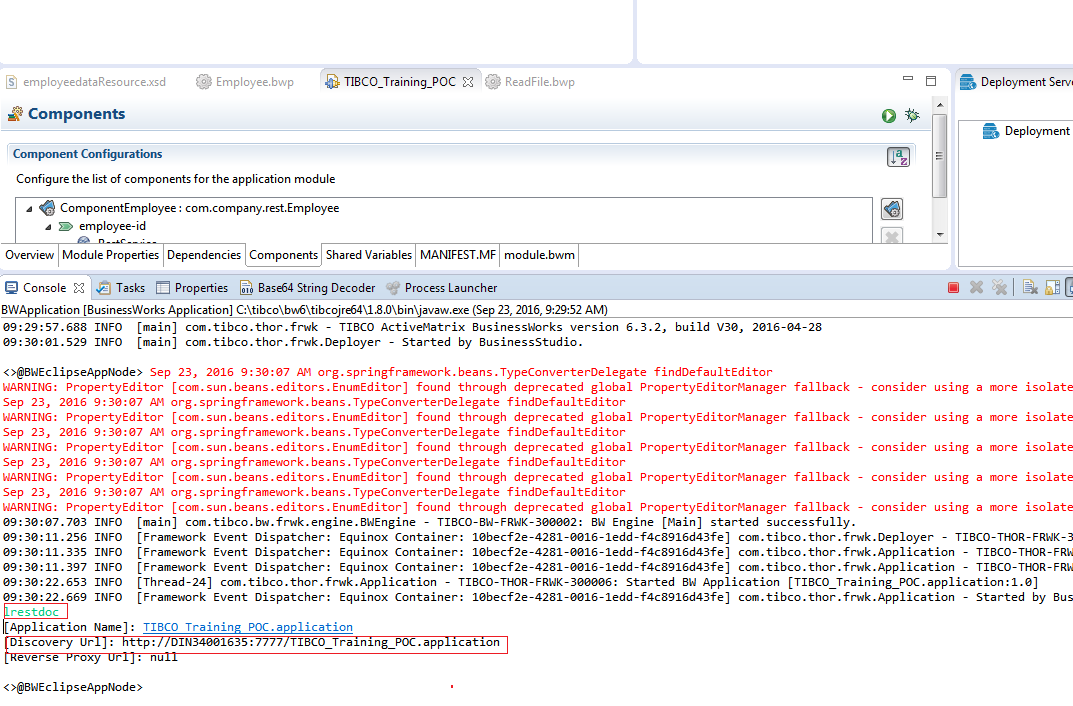


* + Add a timer starter activity and populate the fields in invoke activity required to be posted

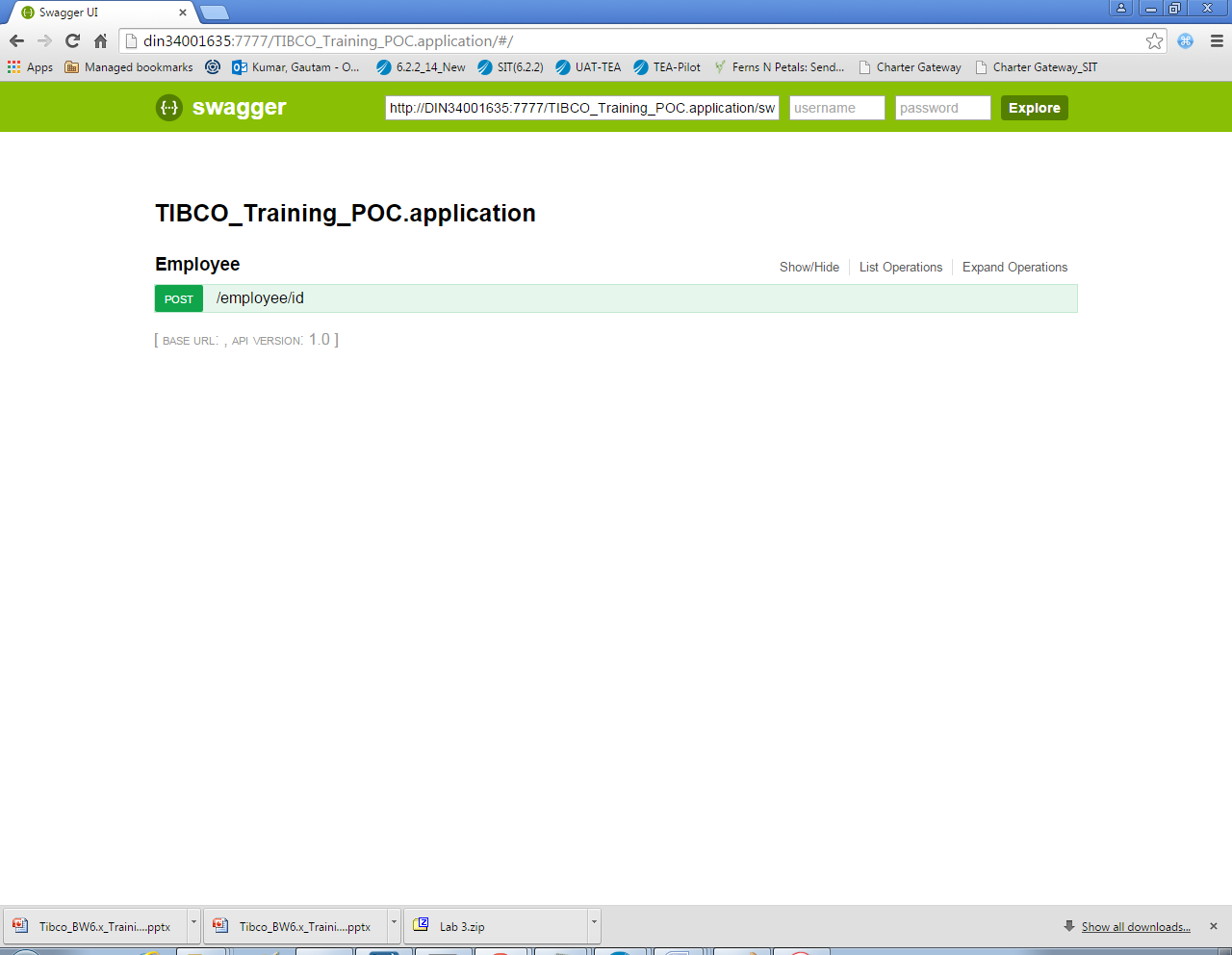


### Testing

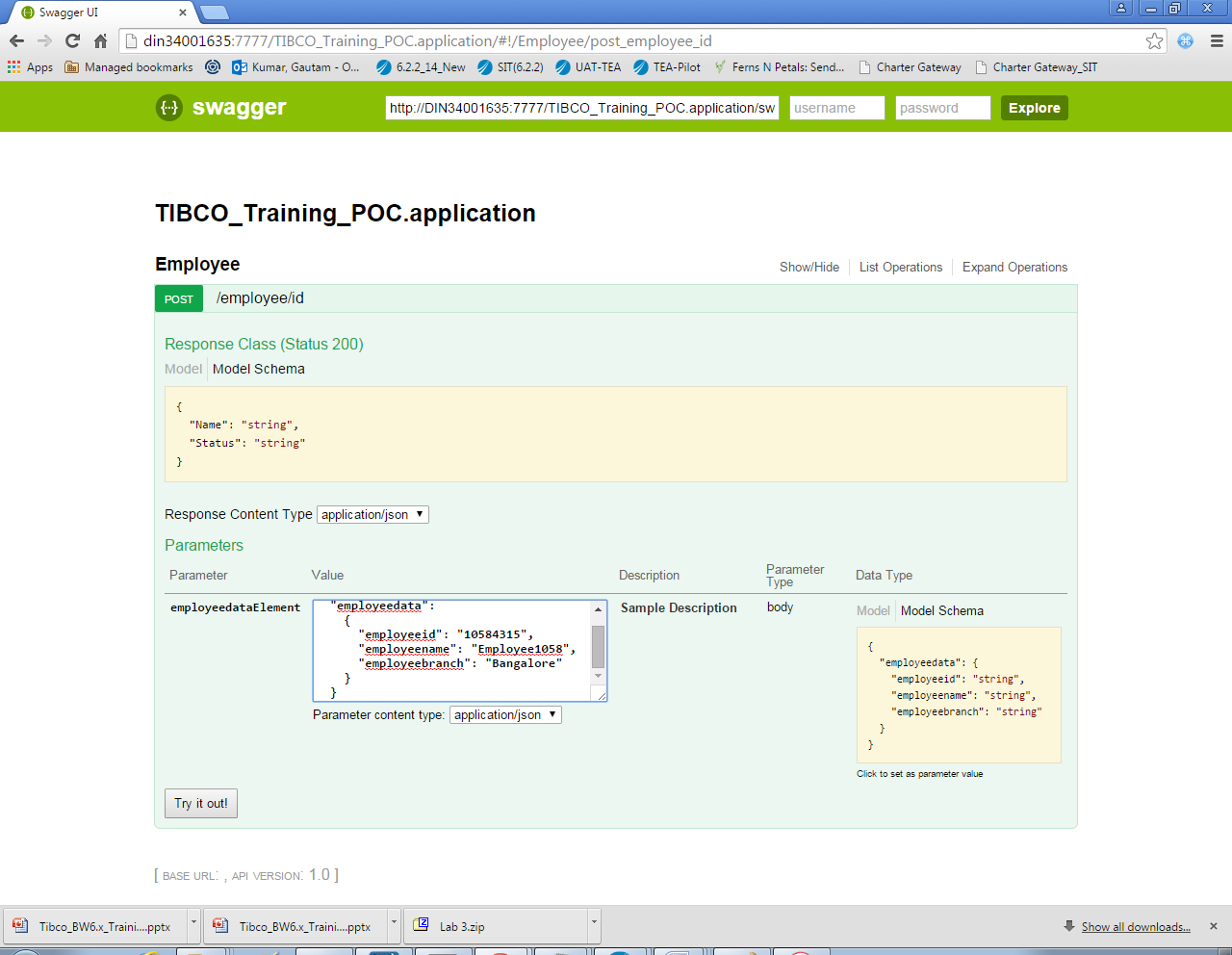
* Debug the application as we did in previous labs.
* Once BW Application is started , type **lrestdoc** and enter in console which will provide the swagger url to test rest service.
* Copy the discovery url from the console and open in chrome as shown in below screenshots:



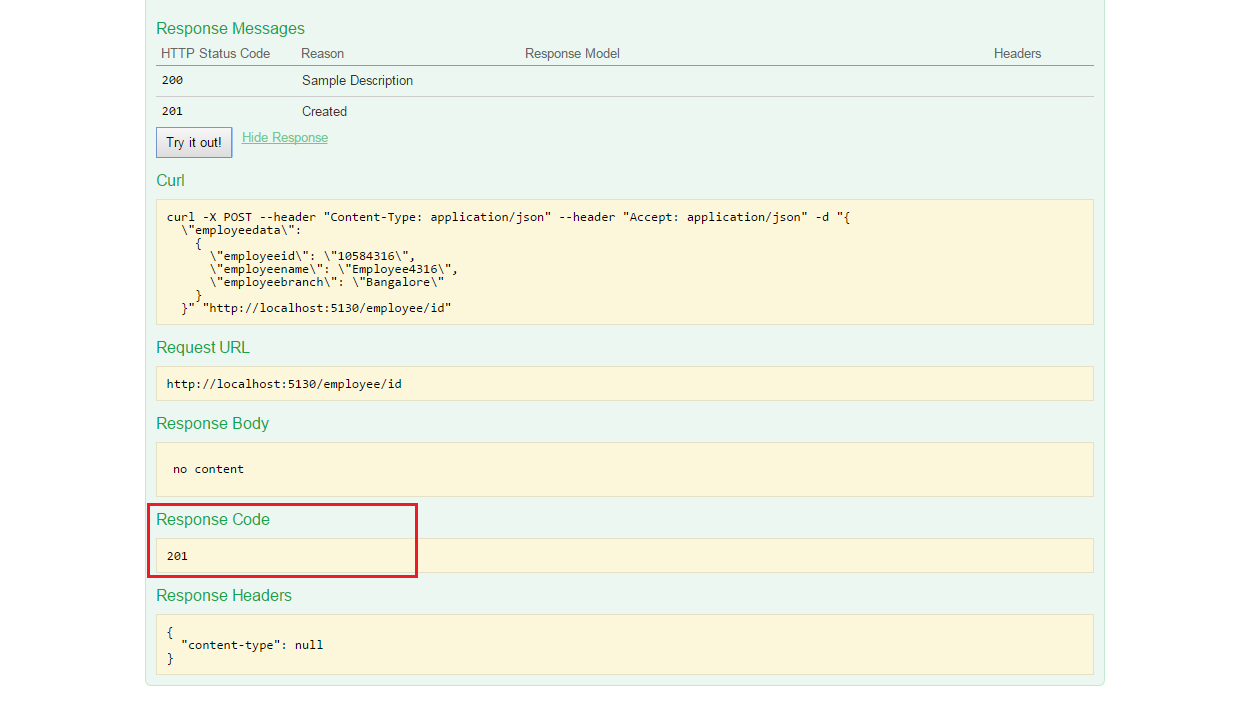
Swagger UI page is displayed as below:

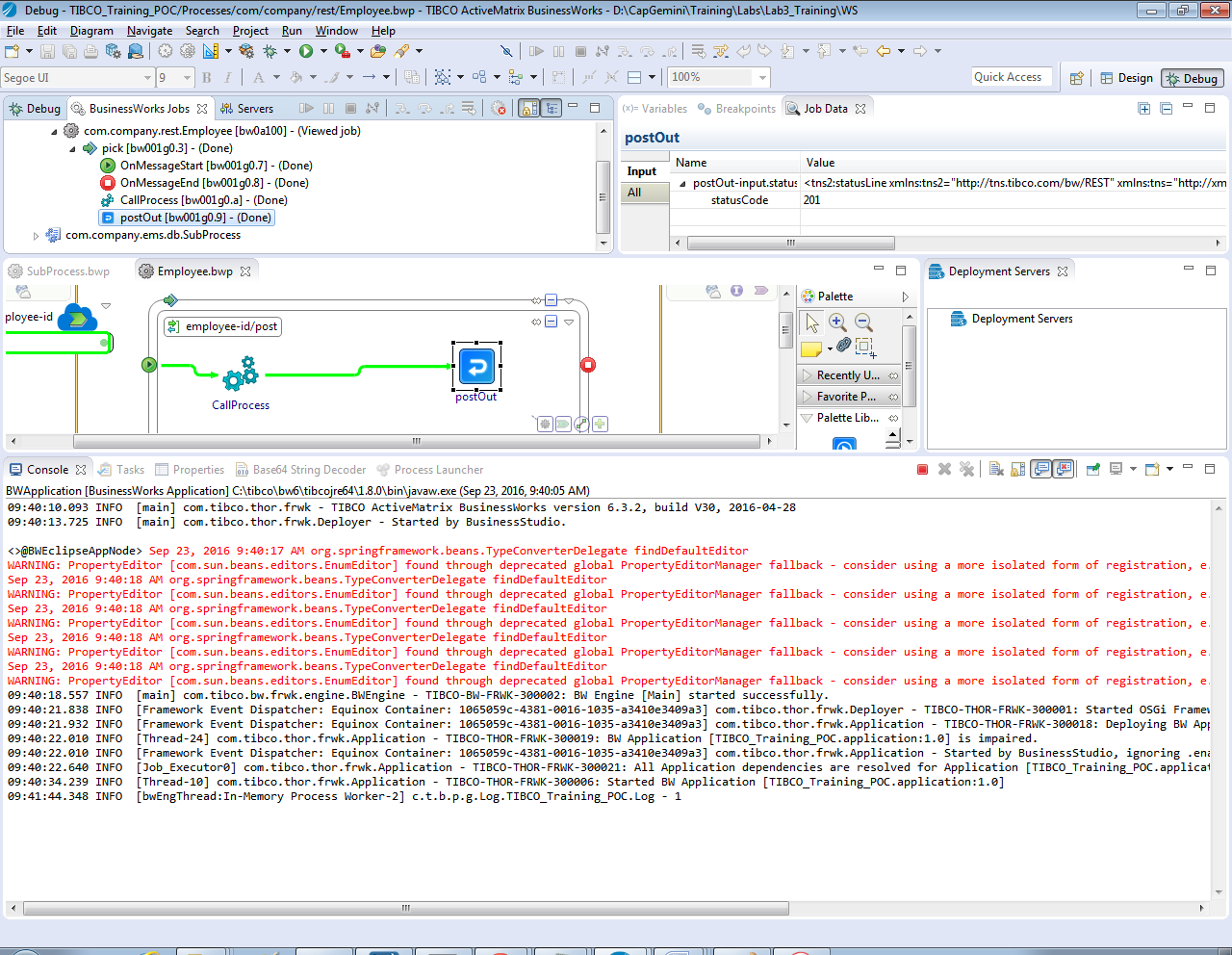


Click on POST on swagger page and enter the input JSON:



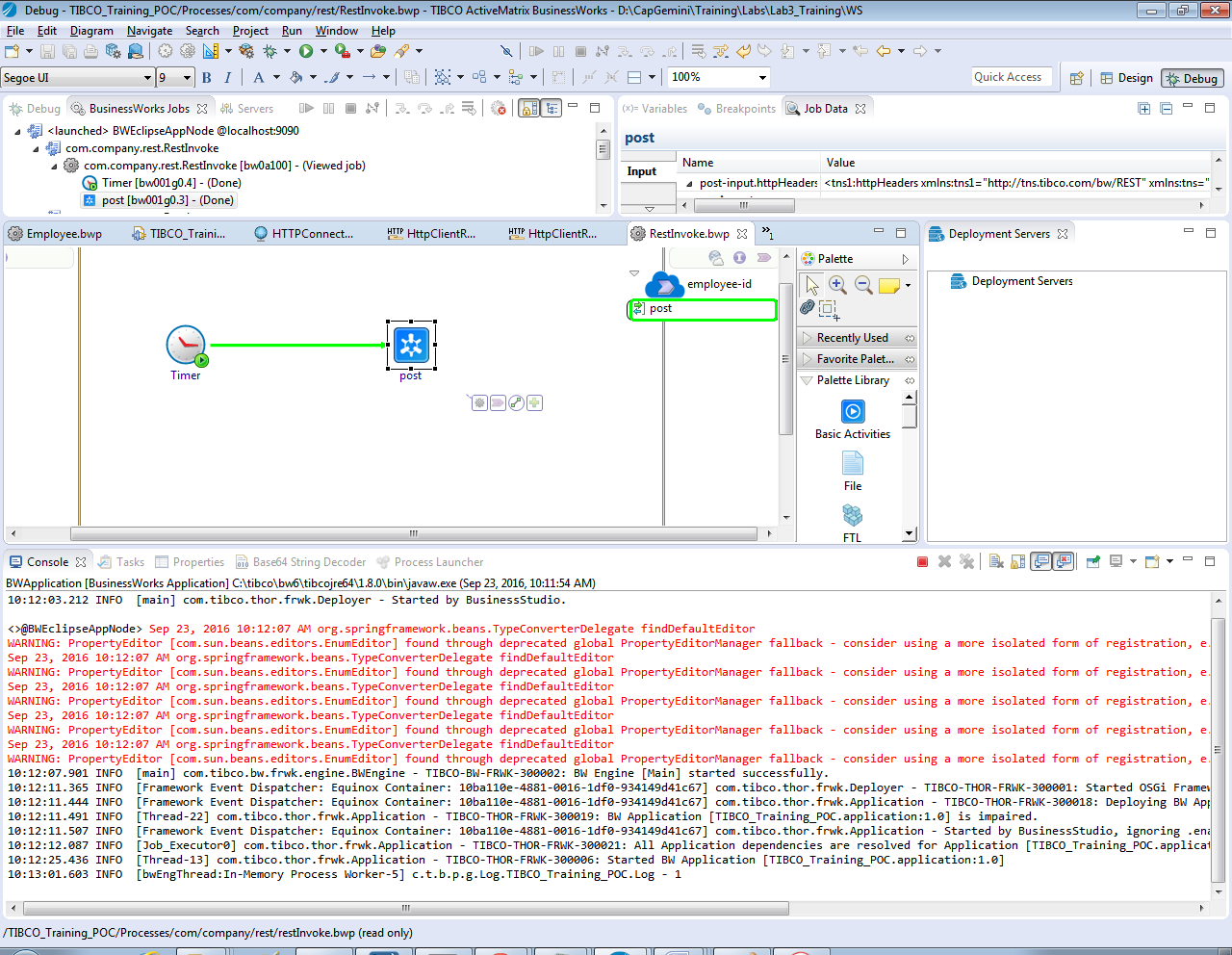
Click on Try it out button and below is the response, Response Code as 201:





* Testing Rest Reference:

Once BW Application is started, timer gets triggered at as mentioned in Start Time and rest service created will be invoked which will post the employee data to the service created as shown below:



# Homework

## Homework-1

Extend Exercise-1 and create solution which receives a message with employee id from a queue Capgemini.TIBCOTraining.HomeWork1.Queue and delete corresponding records from database.

Extend Exercise-1 and create solution which receives a message with employee id from a queue Capgemini.TIBCOTraining.HomeWork2.Queue, search the employee based on the employee id and publish on another queue Capgemini.TIBCOTraining.HomeWork3.Queue.

## Homework-2

Extend Exercise-2 to create operation to delete employee from the database.

Extend Exercise-2 to create operation to search employee from the database.

## Homework-3

Extend Exercise-3 to use REST Get method to get employee details from the database.

Extend Exercise-3 to use REST Delete method to delete employee details from the database.