**SOAP-WS (SOAP over HTTP & SOAP over JMS with Active MQ /IBM Web Sphere MQ) WITH -JBOSS-DB2**

POC Project Setup Guide

[Authors]

* Ramu Choudaboyina
* Venu Vinjamoori

Table of Contents

[Overview: 3](#_Toc2791503)

[1. Eclipse and JBOSS installation 4](#_Toc2791504)

[2. DB2 Setup and JNDI creation on JBoss 8](#_Toc2791505)

[3. Create JPA using Eclipse-Link implementation then entity class generation and Soap Over HTTP WS 9](#_Toc2791506)

[4. MQ (both ActiveMQ and WebSphereMQ) Configuration and Setup & Credentials with Two security Checks 18](#_Toc2791507)

[5. Add binding file of MQ in standalone folder of JBoss. 20](#_Toc2791514)

[copy & paste .binding file from MQ folder to Jboss folder as shown in following diagram and update the fully qualified path in destination property configured in wsdl 20](#_Toc2791515)

[6. Create a Soap Over JMS WS Service configurations 21](#_Toc2791516)

[7. start Queue Manager and test using rfhutil tool using Request.XML 21](#_Toc2791517)

[8. Add required Jars 23](#_Toc2791518)

[9. add Security to Soap Request 24](#_Toc2791519)

[10. Test Client class generation 24](#_Toc2791520)

[Final Touchups & pending Items: 25](#_Toc2791521)

# Overview:

As part of Eidiko POC project we have created Soap HTTP/JMS with JPA project with IBM Web Sphere MQ on JBoss Server.

this Document is being created to help Developers to understand and create necessary Configurations and adding required Jars for SOAP WS (HTTP/JMS with Active MQ & IBM Web Sphere MQ) to run / execute /enhance Project.

Topics:

1. Eclipse and JBOSS installation
2. DB2 Setup and JNDI creation on JBoss
3. Create JPA using Eclipse-Link implementation then entity class generation and Soap Over HTTP WS
4. MQ (both ActiveMQ and WebSphereMQ) Configuration and Setup & Credentials with Two security Checks

-ConnAuth & Auth disabled

1. Add binding file of MQ in standalone folder of JBoss.
2. Create a Soap Over JMS WS Service configurations

-Soap over JMS (Active MQ/Web sphere MQ), same as above but changes required in adding Jars for MQ

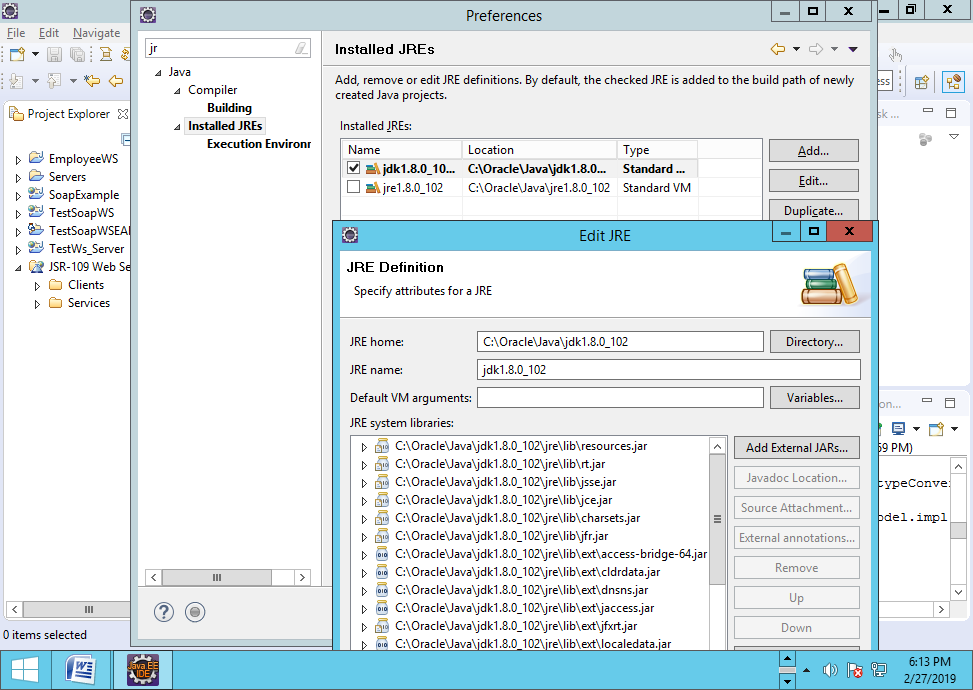
1. start Queue Manager and test using **rfhutil** tool using Request.XML
2. Add required Jars
3. add Security to Soap Request
4. Test Client class generation

**Details:**

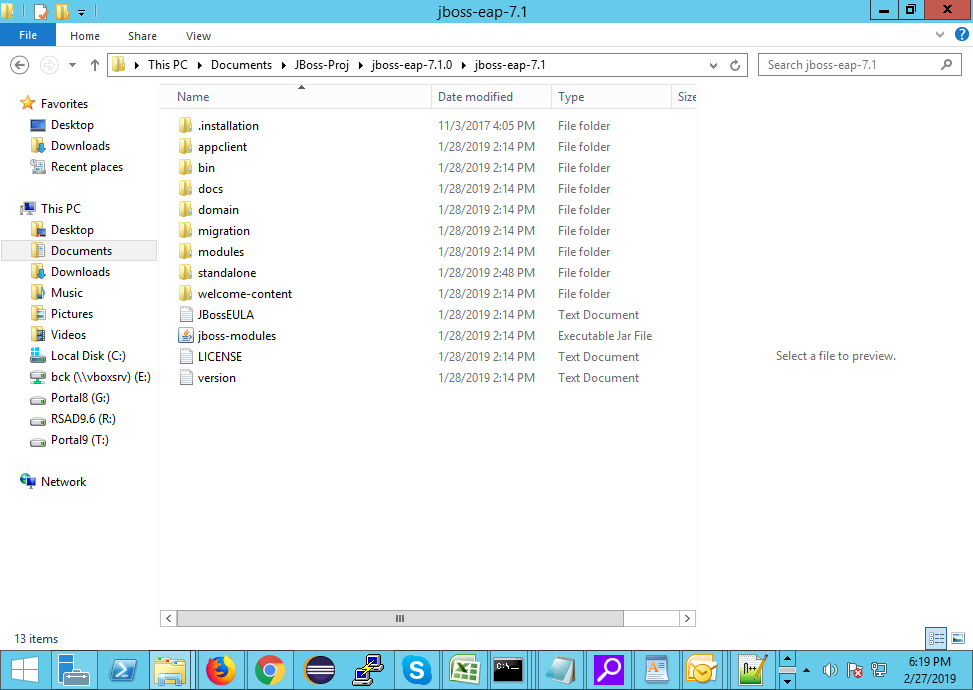
1. **Eclipse and JBOSS installation**

Install Eclipse IDE for Java EE ( Neon.3 or latest Version )

preferences ->Installed JREs -> set to JDK latest Version as shown below

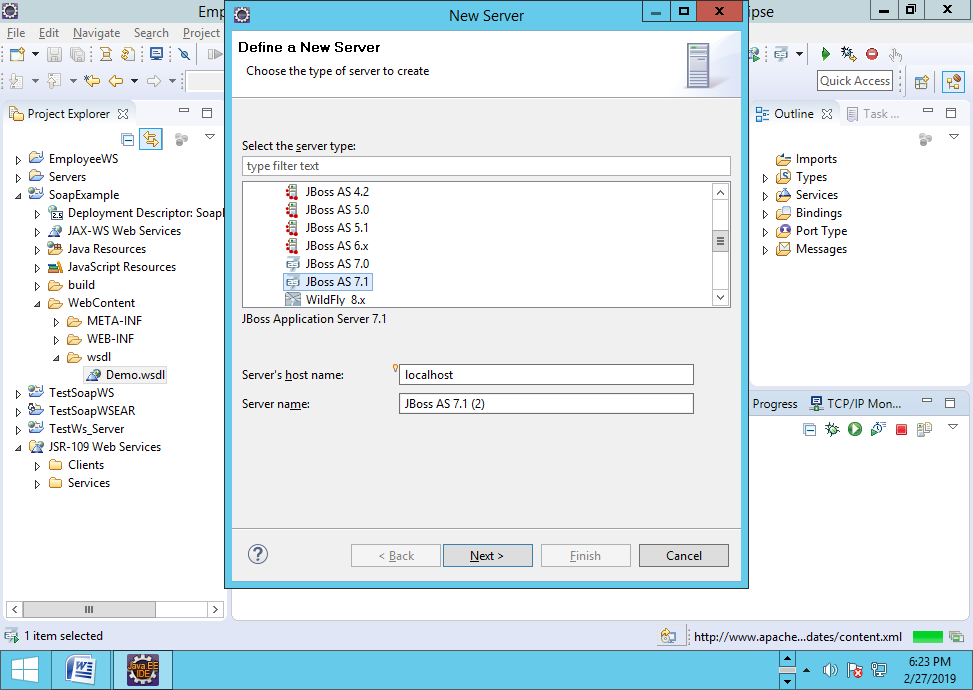


down load JBoss Server Software Compressed folder jboss-eap-7.1.0.zip and extract contents in local directory (Example: C:\Users\Administrator\Documents\JBoss-Proj\jboss-eap-7.1.0 )

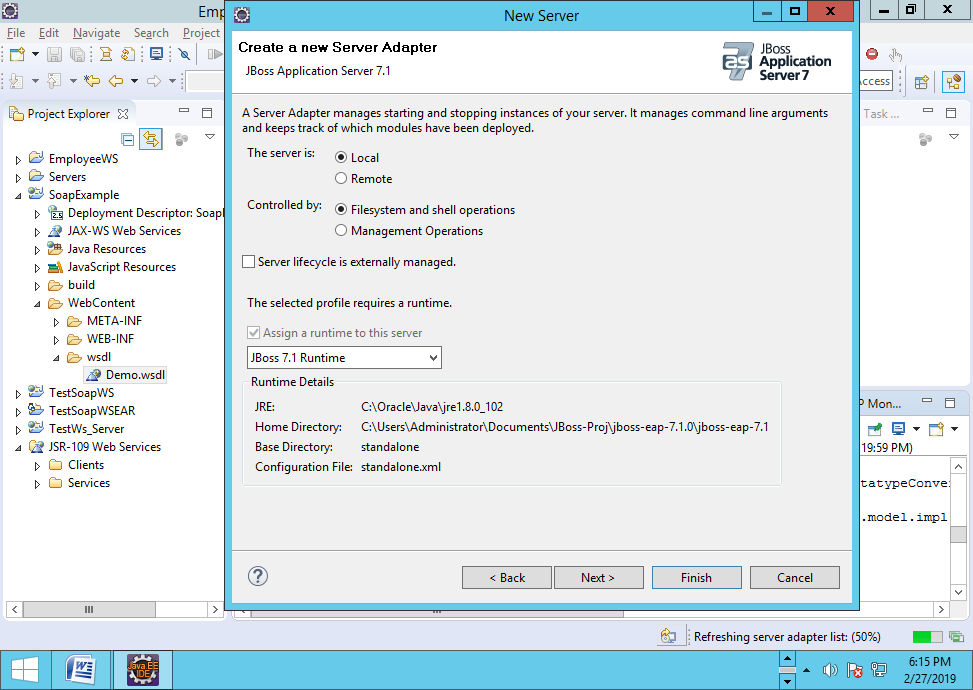


goto Eclipse IDE and Windows -show view ->Servers

right click on Servers view -> new -Server -> select Option as JBOSS



click next-> select Options as below



and click Finish.

now Eclipse is configured with JBOSS Application Server. now we can right click on JBoss Server and Start it. after some time, eclipse console shows that JBOSS server started.

then we can have access to Admin Console at **http://localhost:9990/console**

for the first Time we dont have any users created on JBOSS Admin Console, so we need to create it first to login.

go to JBOSS bin folder on local Disk (Ex: C:\Users\Administrator\Documents\JBoss-Proj\jboss-eap-7.1.0\jboss-eap-7.1\bin) double click on add-user.bat windows batch file. follow the instructions create username and password. and finish the process.

then restart the JBoss Server from Eclipse and login to admin console using credentials that we created.

1. **DB2 Setup and JNDI creation on JBoss**

download or open Organization common sharing folder and get the copy of db2 software (Example: from Location \\eswn6415\K\Software\DB2-10.1-Win64) into local system.

unzip the setup files and note the setup file path.

click on setup file to install ( click on Install a product->Install New).

click next... give path where it is required

for username: wpsadmin ,password: Sarasu10

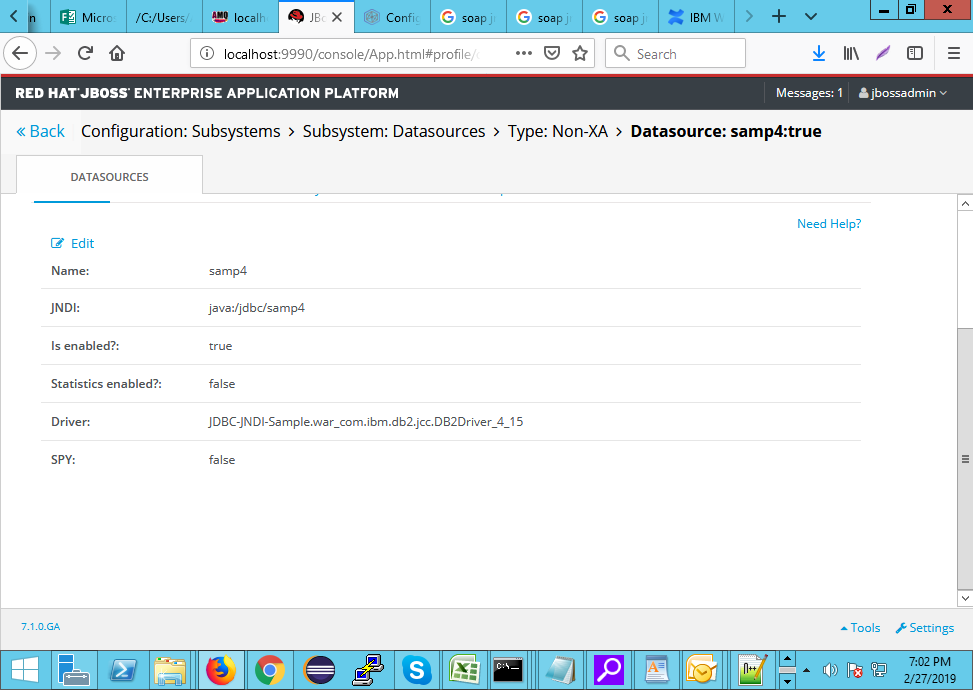
and Finish.

now DB2 got installed on your Machine.

open CommandLine Processor and type connect db2. try logining in see availble schema and tables. if required we can create our own schema and table data.

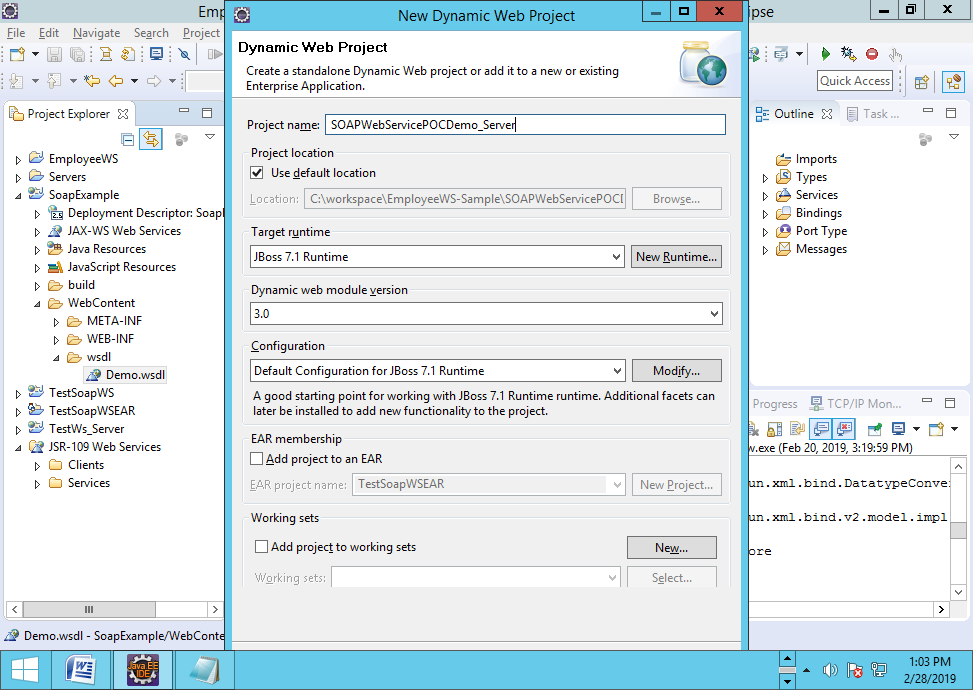
**JNDI creation:** <this step we are not using instead we are directly giving JNDI details in Code itself, but keeping for future reference>

login to JBoss Console ->Configuration -> SubSystems -> DataSources ->Type: Non-XA and give following info to create JNDI



1. **Create JPA using Eclipse-Link implementation then entity class generation and Soap Over HTTP WS**

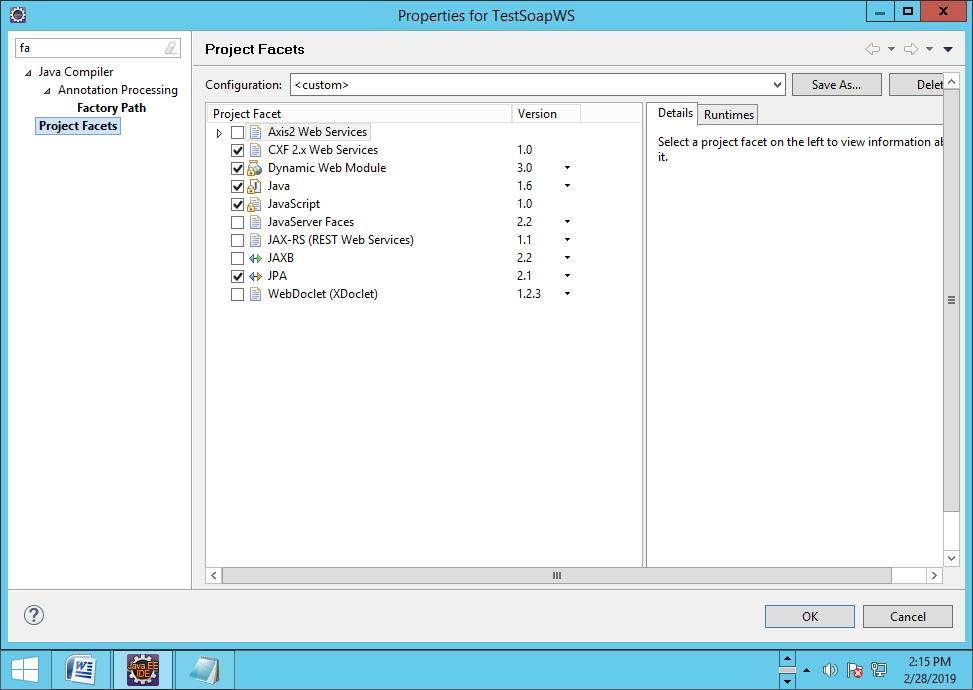
go to eclipse -> right click on Project explorer ->new -Project -> select Dynamic Web Project under Web -> give Project Name (Example: SOAPWebServicePOCDemo\_Server) **-> target Run Time** as JBoss7.1 Run Time and **Dynaimic web module version** as 3.0 ...ect.



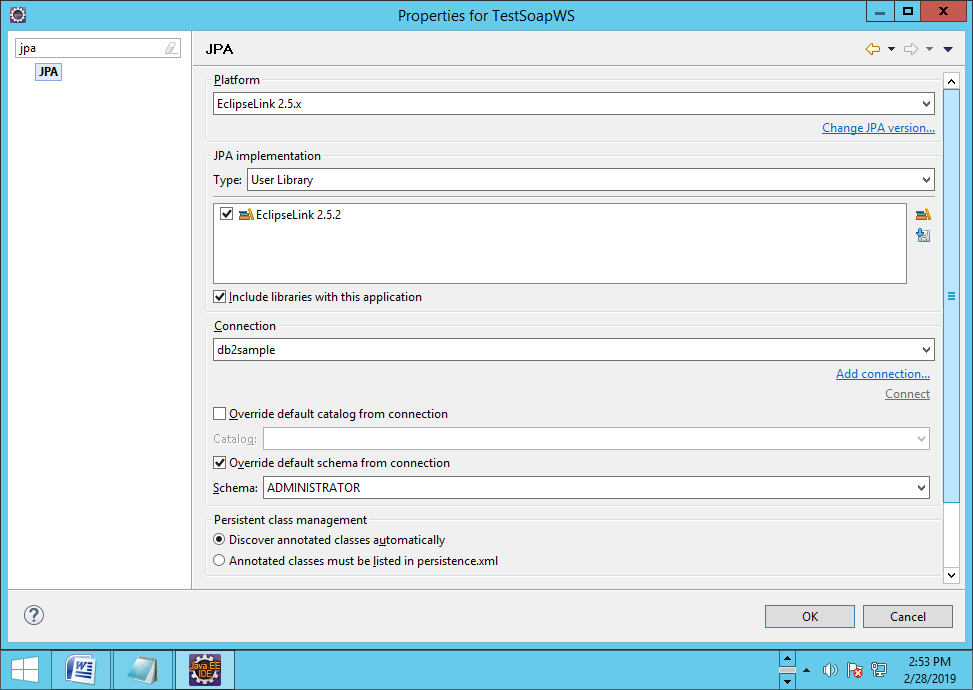
click Finish.

(convert Project to JPA capable )

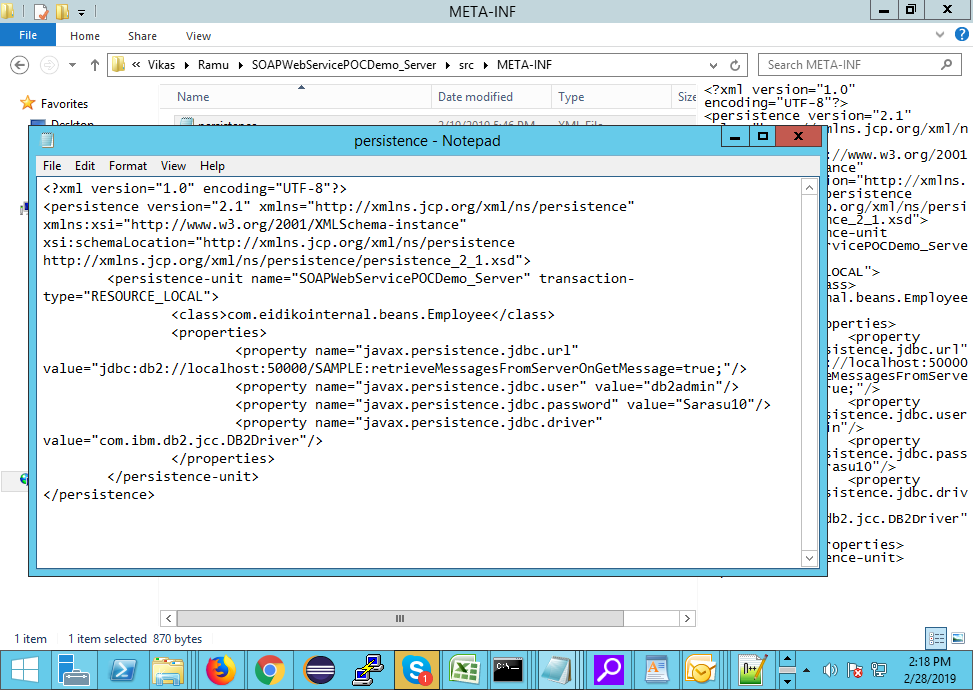
right click on Project->properties ->project facet -> select JPA



and provide runtime configurations for eclipse link , we need to add jars and do following



with this above step , eclipse will generate persistance.xml file for JPA configuration.

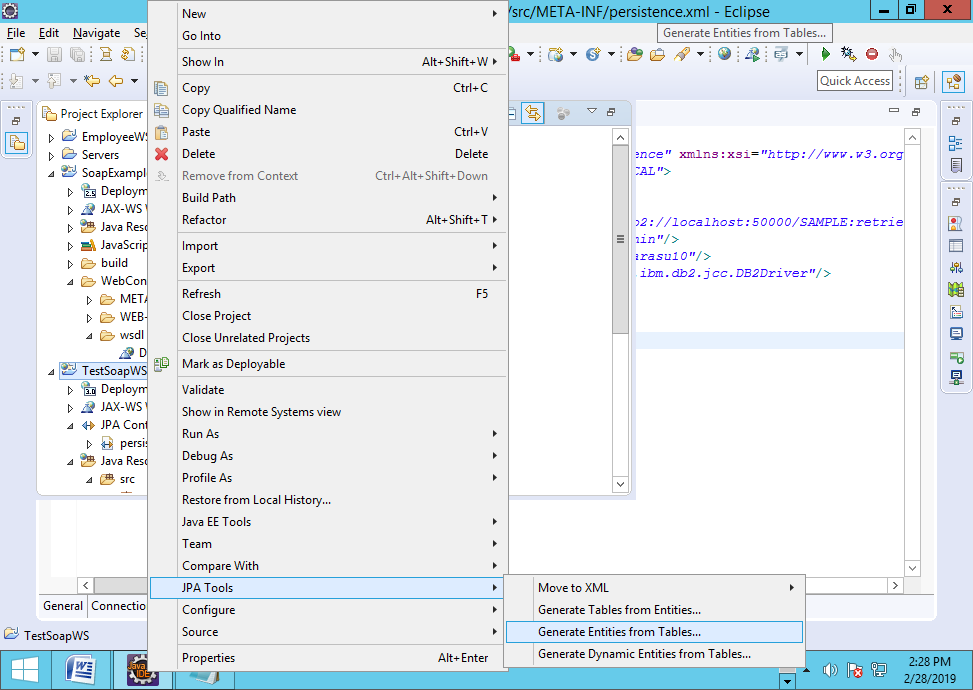
here we can mention database details (like driver name,User/PASSWORD ...etc) to connect db from application. 

* Create **EntityManagerFactory** class :

this class has getEntityManager() method which returns EntityManager obj

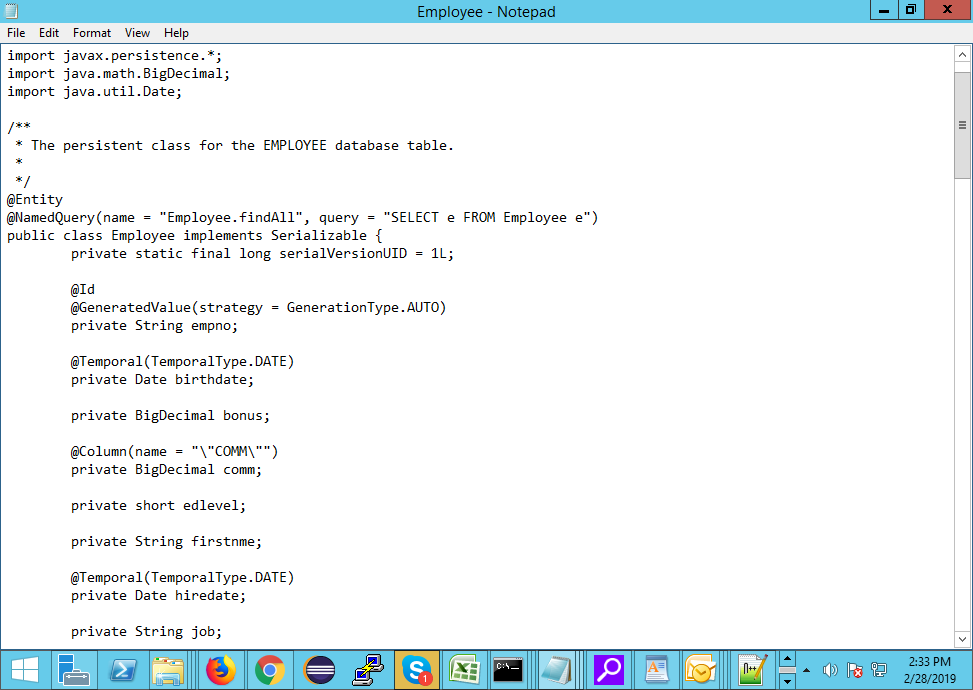
* create **Entity bean** Class :

right click on eclipse package explorer and JPA Tools ->generate entities from tables

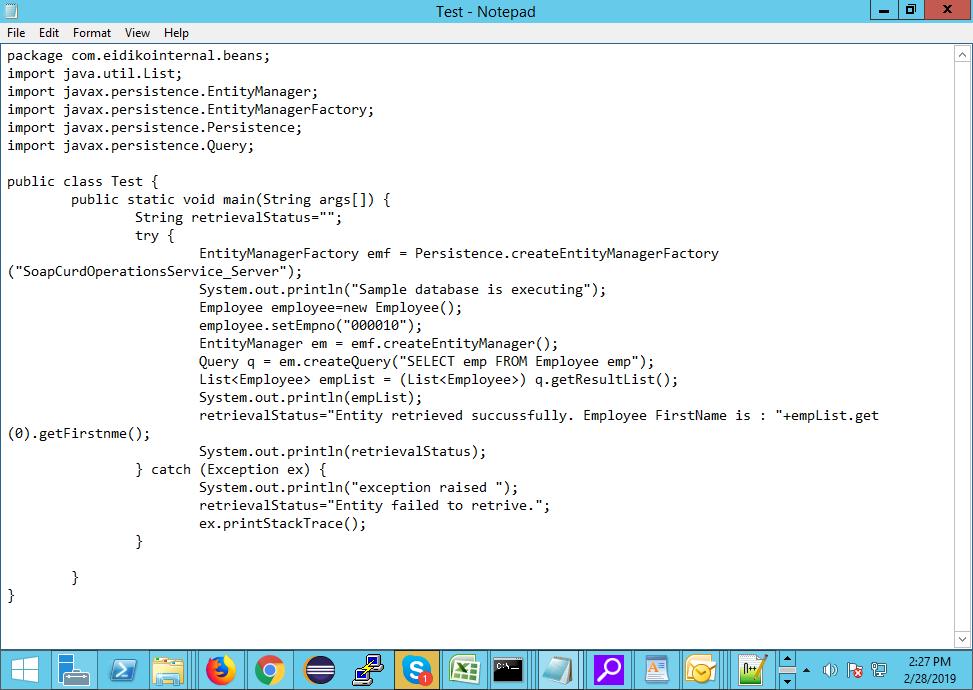


here we need to select tables for which we need to generate Entity classes

(example: we have selected Employee table, so eclipse generated Employee class based on table structure and constraints.



* to test this JPA connection, we can write a sample test client class and confirm.



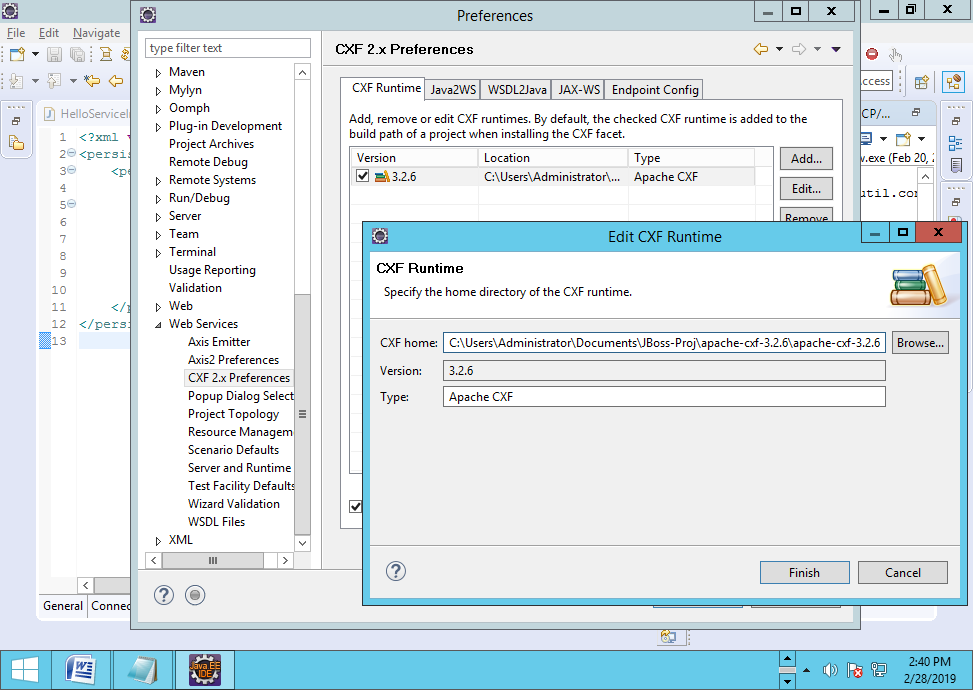
* **create Service class for CURD operations**

create a class with retrieve/delete... for employee and CURD Operations Implementation class

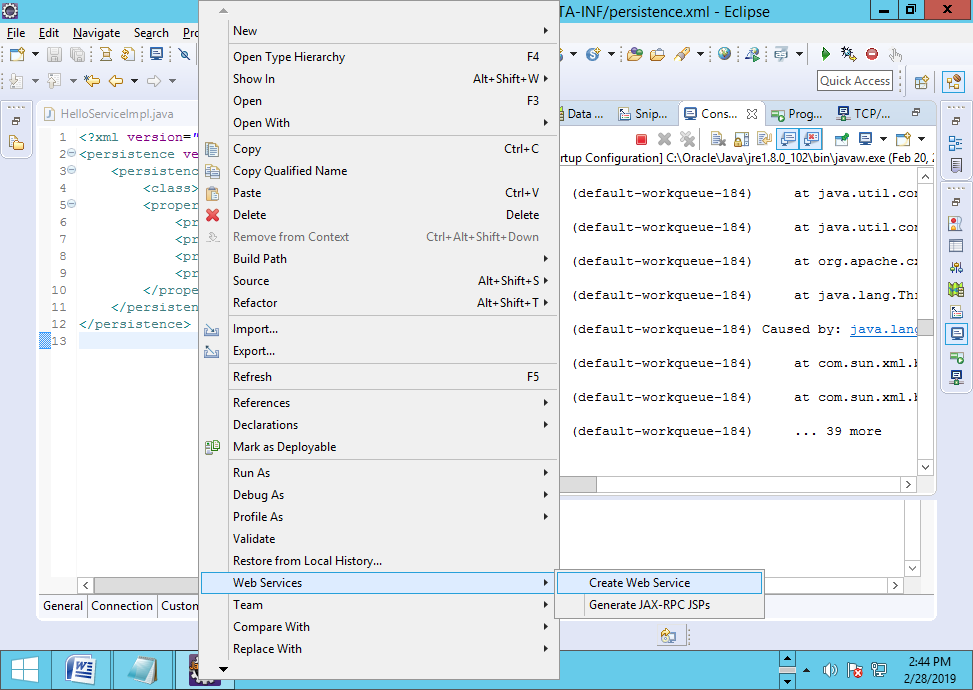
* **create Web Service for service class methods**:

to do this operation we need apache cxf run time environment

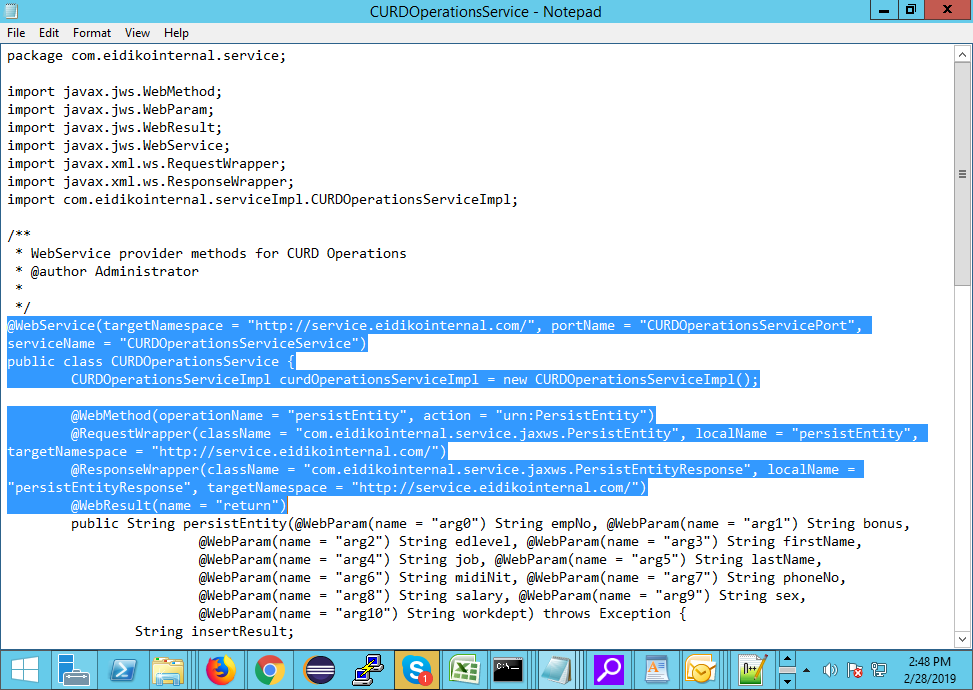
like this



then right click on service class to create a WebService for that Service Class



this will append service class methods with relevant annotations like following



also above step will generate WSDL file based on our Services. so now we can test SOAP Over HTTP using test client.

1. **MQ (both ActiveMQ and WebSphereMQ) Configuration and Setup & Credentials with Two security Checks**

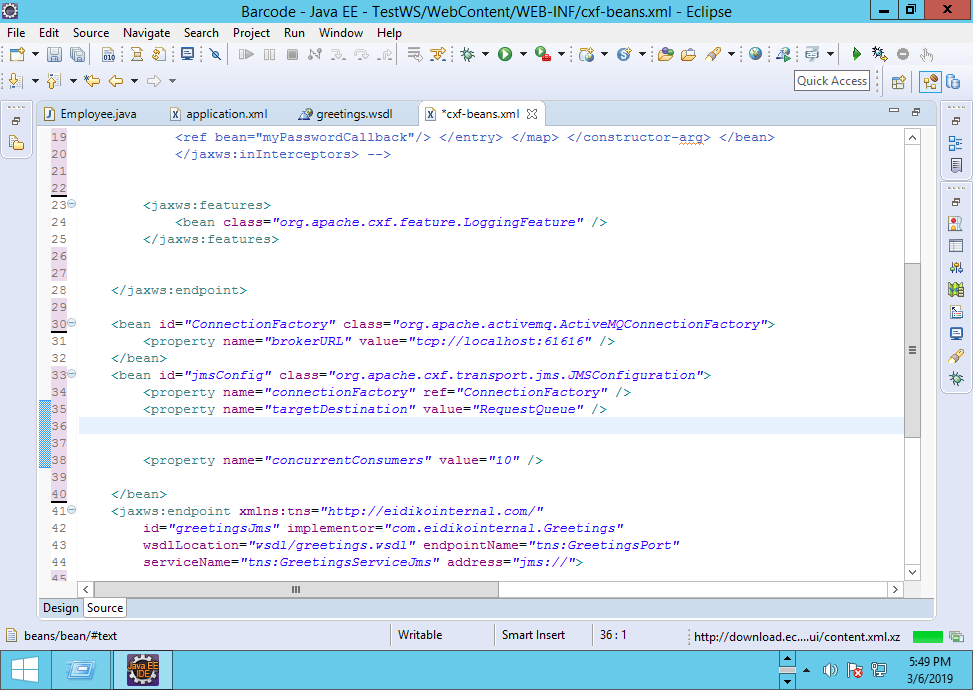
**Note: [if required take the help of Tibco / Websphere MQ Admin POC for this step]**

install WebSphere MQ software in local machine and create Queue manager as attached Document below

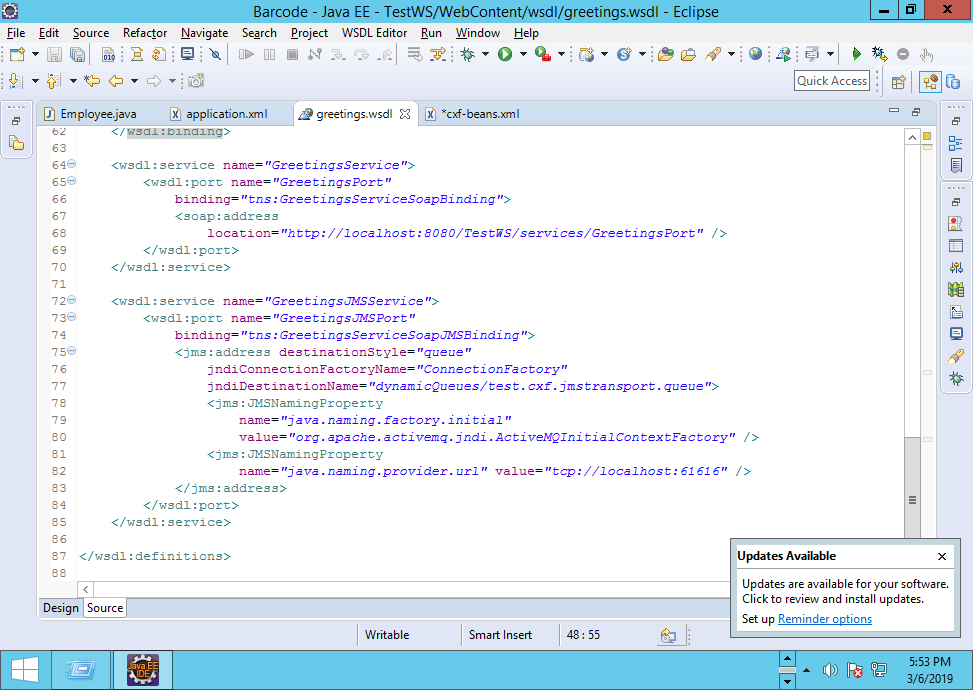
****

**-----------------**

if it is ActiveMQ, we need to install apache-activemq-5.14.5 software and start server by going to bin path from command propmpt and run start activemq.batch and give following configuration in cxf-bean.xml

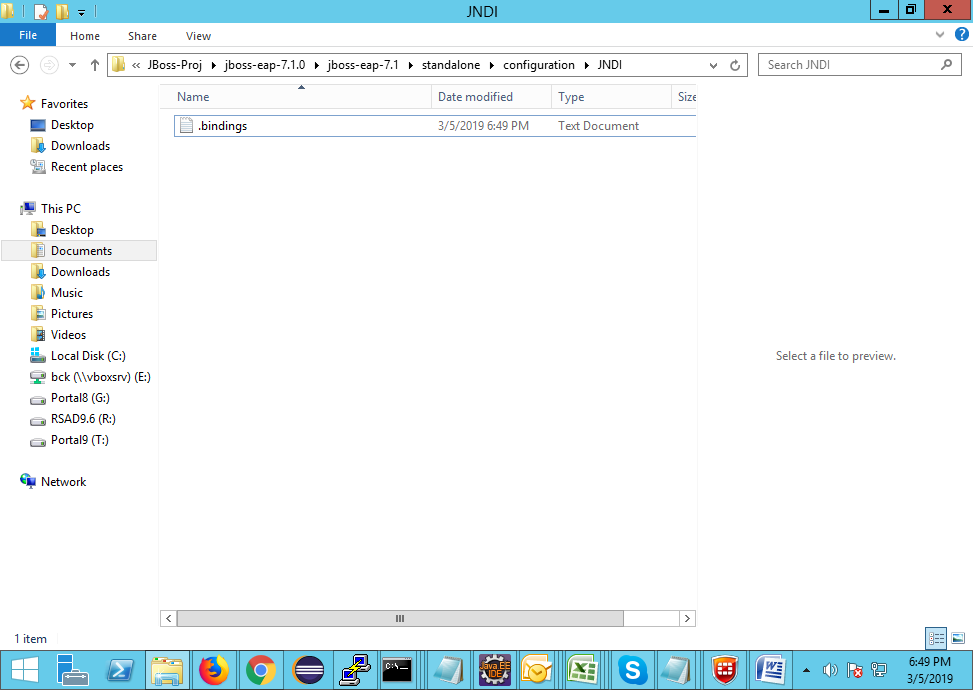
****

we can provide queue information in WSDL file

****

1. **Add binding file of MQ in standalone folder of JBoss.**

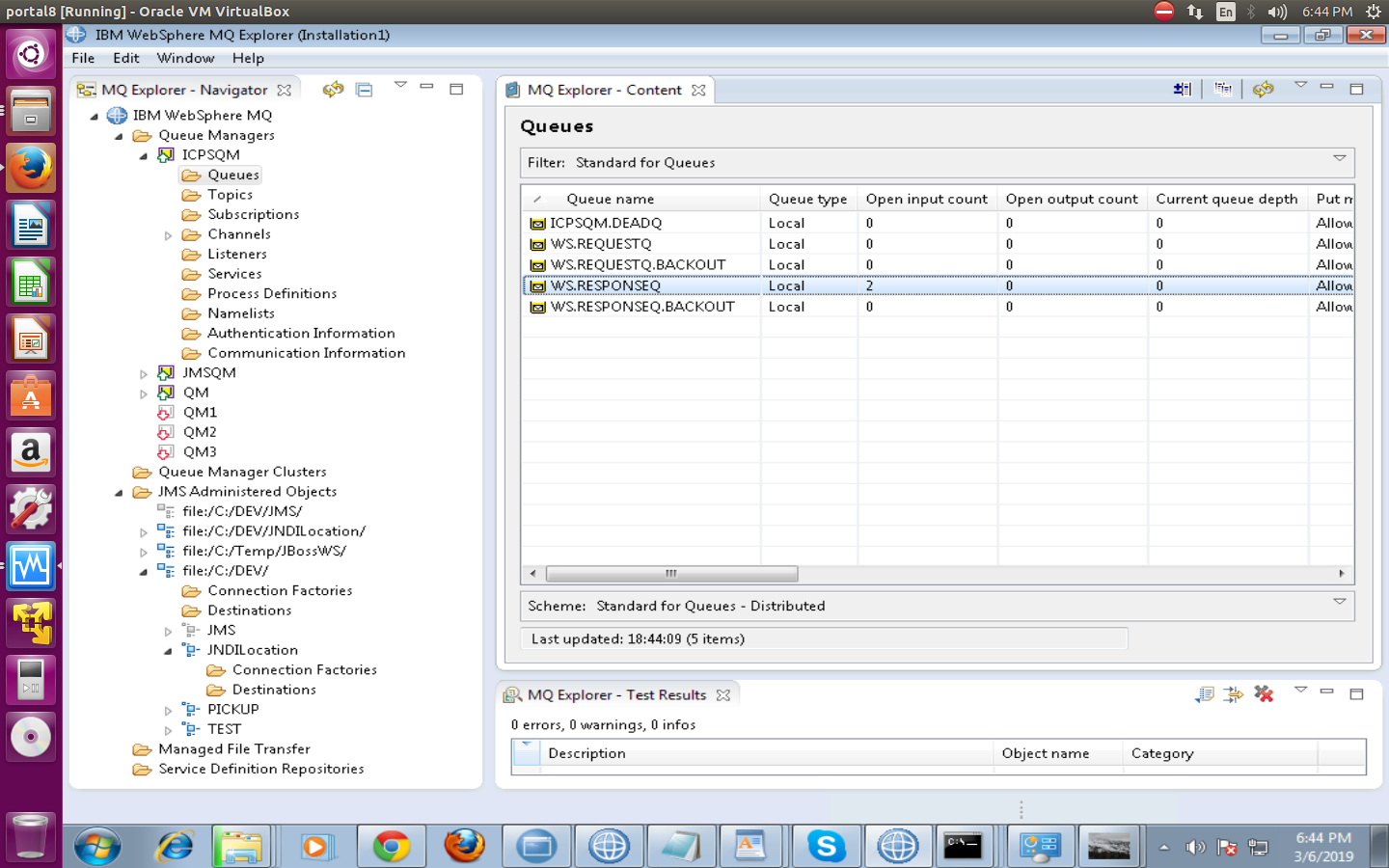
**copy & paste .binding file from MQ folder to Jboss folder as shown in following diagram and update the fully qualified path in destination property configured in wsdl**

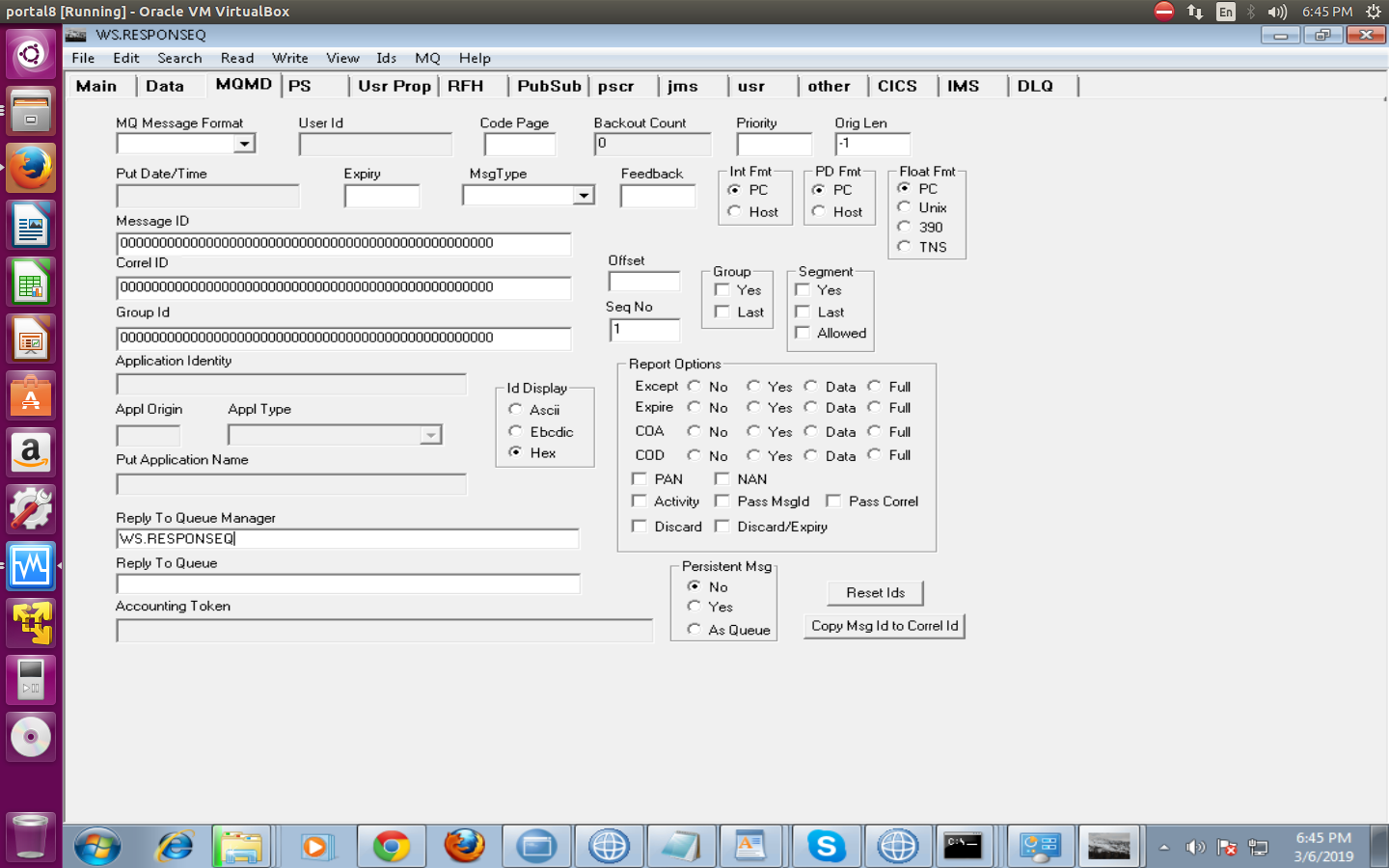
****

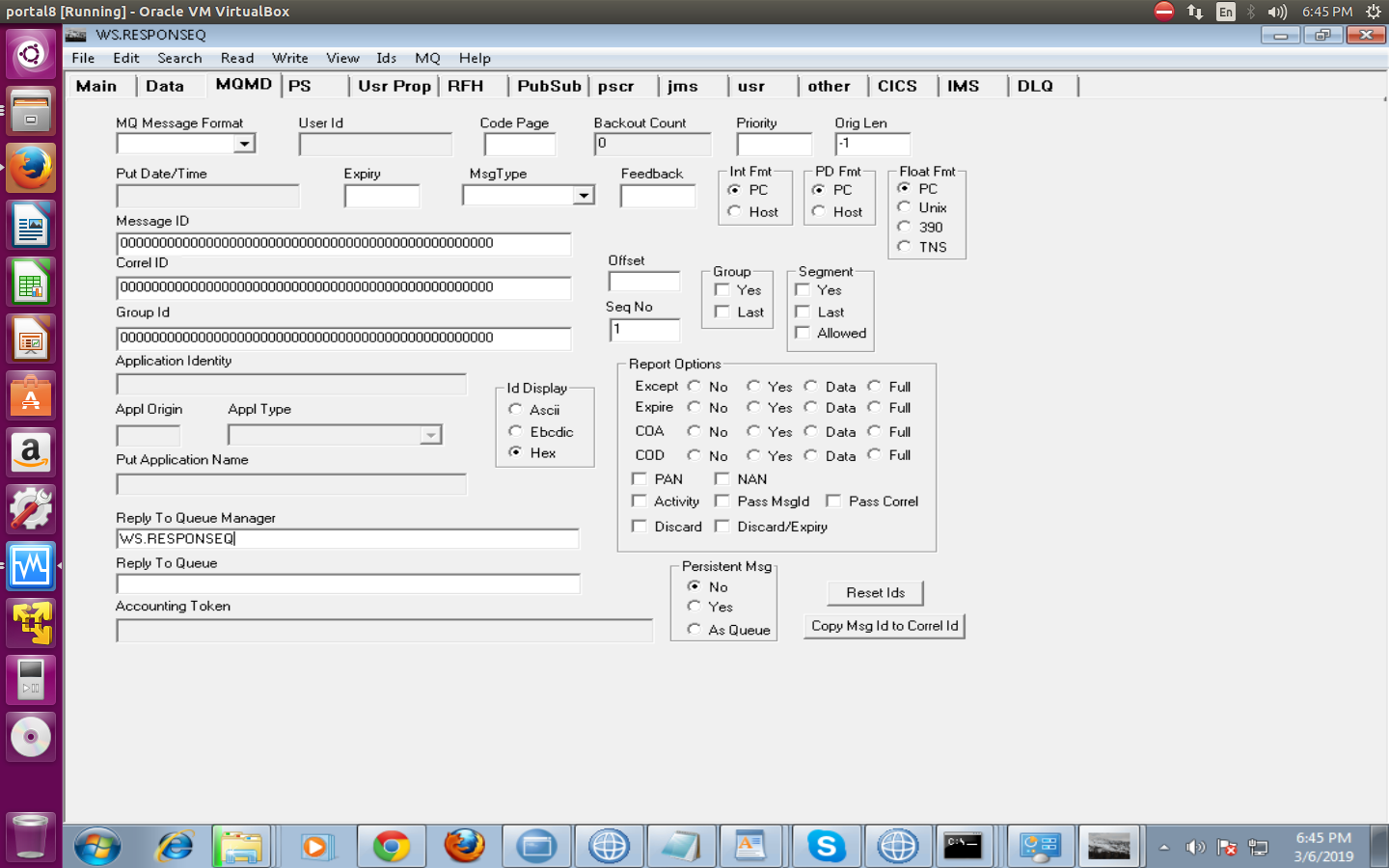
1. **Create a Soap Over JMS WS Service configurations**

****

1. **start Queue Manager and test using rfhutil tool using Request.XML**

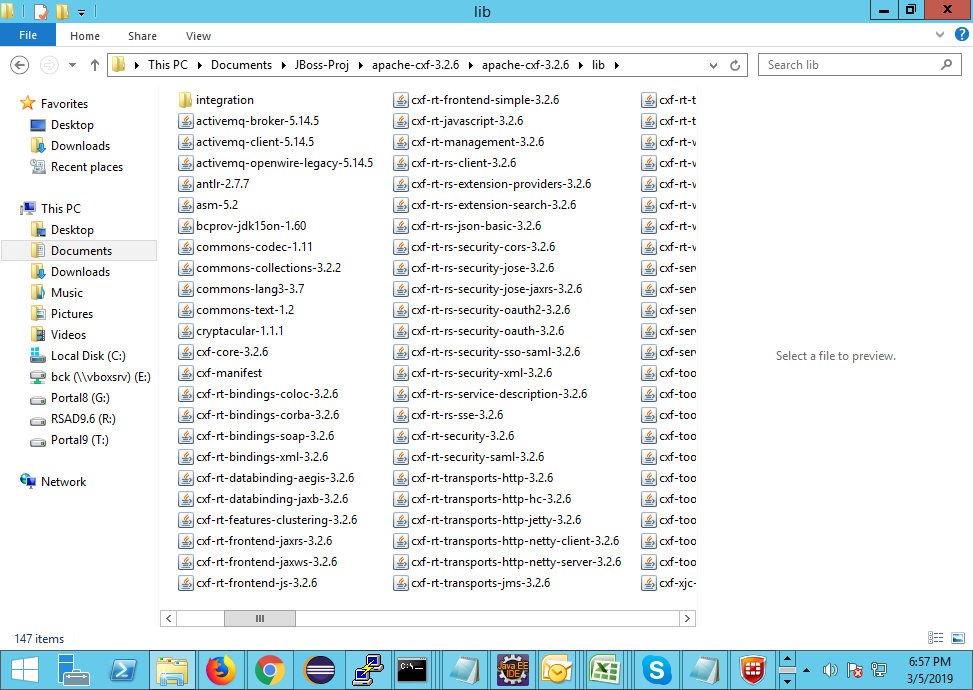
****

****

****

1. **Add required Jars**

Apache CXF related Jars

****

also add MQ or Active MQ jars

Also JPA Dependencies for DB2JCC4.jar, DB2Licence.jar and Eclipselink2.5.jar,persistance2.5.jar

1. **add Security to Soap Request**

something similar to this attachment



1. **Test Client class generation**

something similar to this attachment



# Final Touchups & pending Items:

Mention all Required Jars Information category wise and path where we need to keep

Resolve all Build Issues in eclipse

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***The End**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*