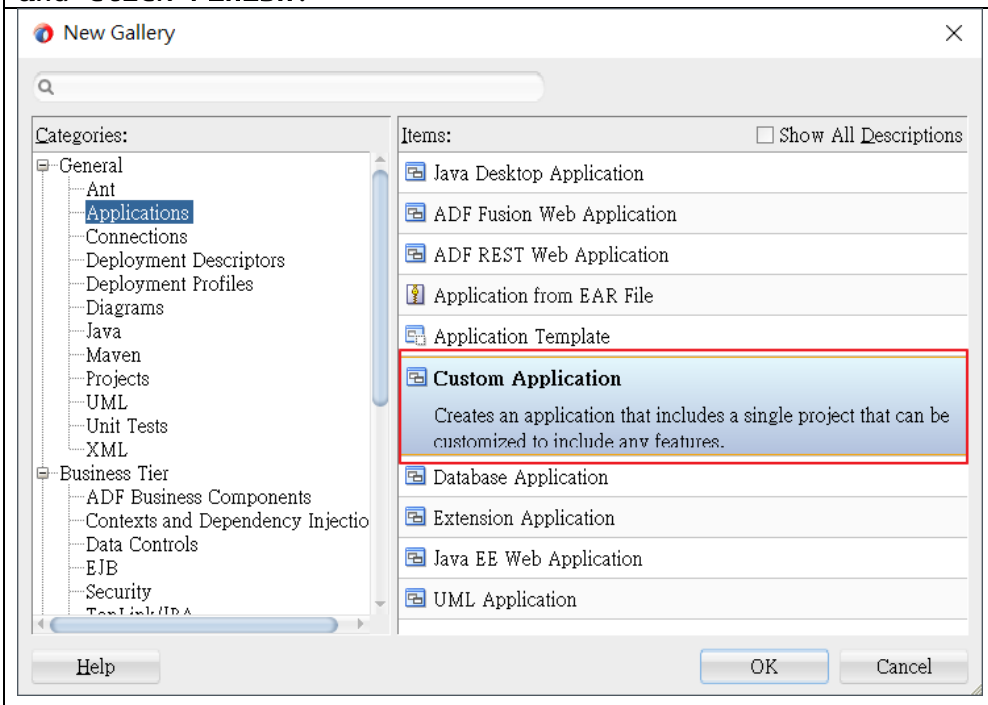
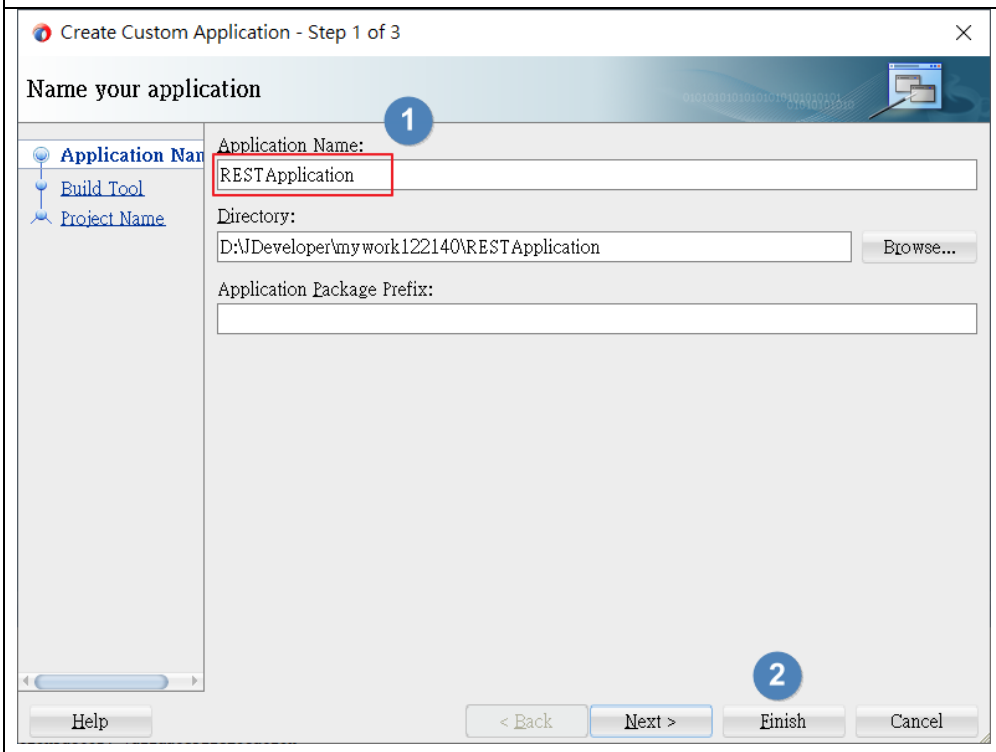
本實作練習請使用 **Jdeveloper 12.2.1.2.0** 以上版本，採用 JPA,EJB 3.1 及 JAX-RS 2.0

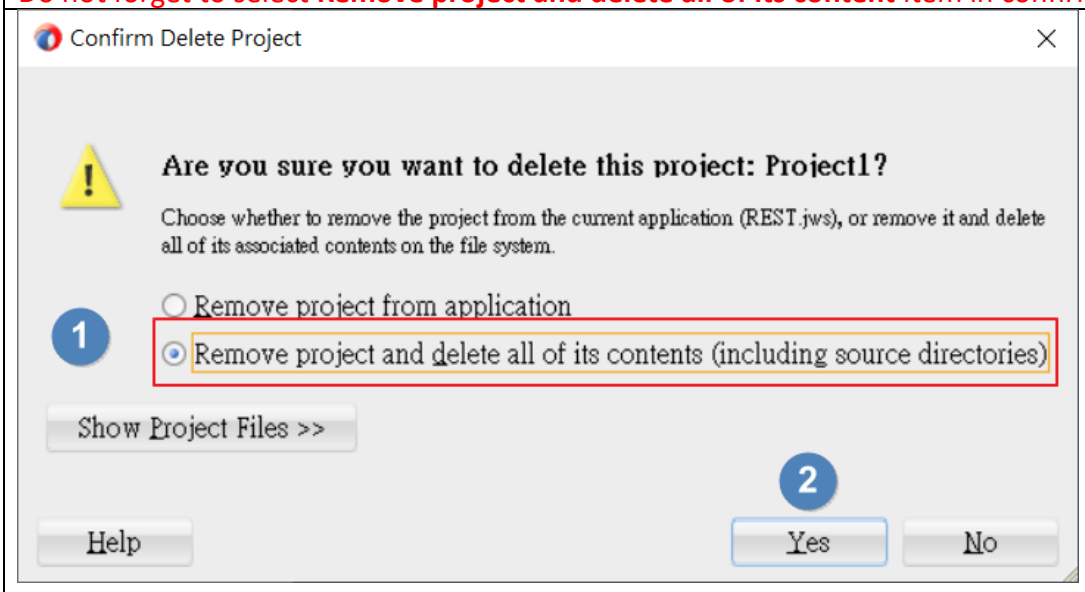
```
create table Person
(
  id   INTEGER,
  age  INTEGER,
  name VARCHAR2(20)
  constraint PK_PERSON primary key (id)
);
```

Navigate **File>New>Application** in JDeveloper and select **Custom Application** gallery item in New Gallery window.Specify application name as **RESTApplication** and click **Finish**.





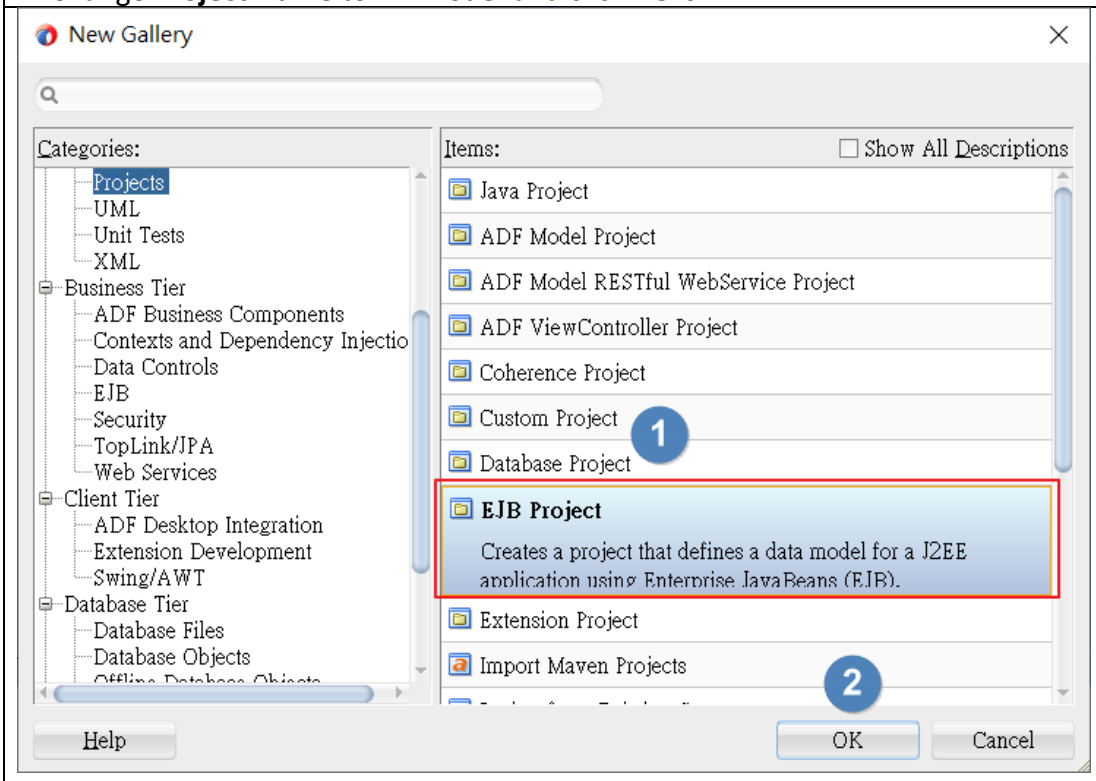
Delete the default ***Project1*** project in application workspace (right click the ***Project1*** and select **Delete Project** menu item).
Do not forget to select **Remove project and delete all of its content** item in confirmation window.

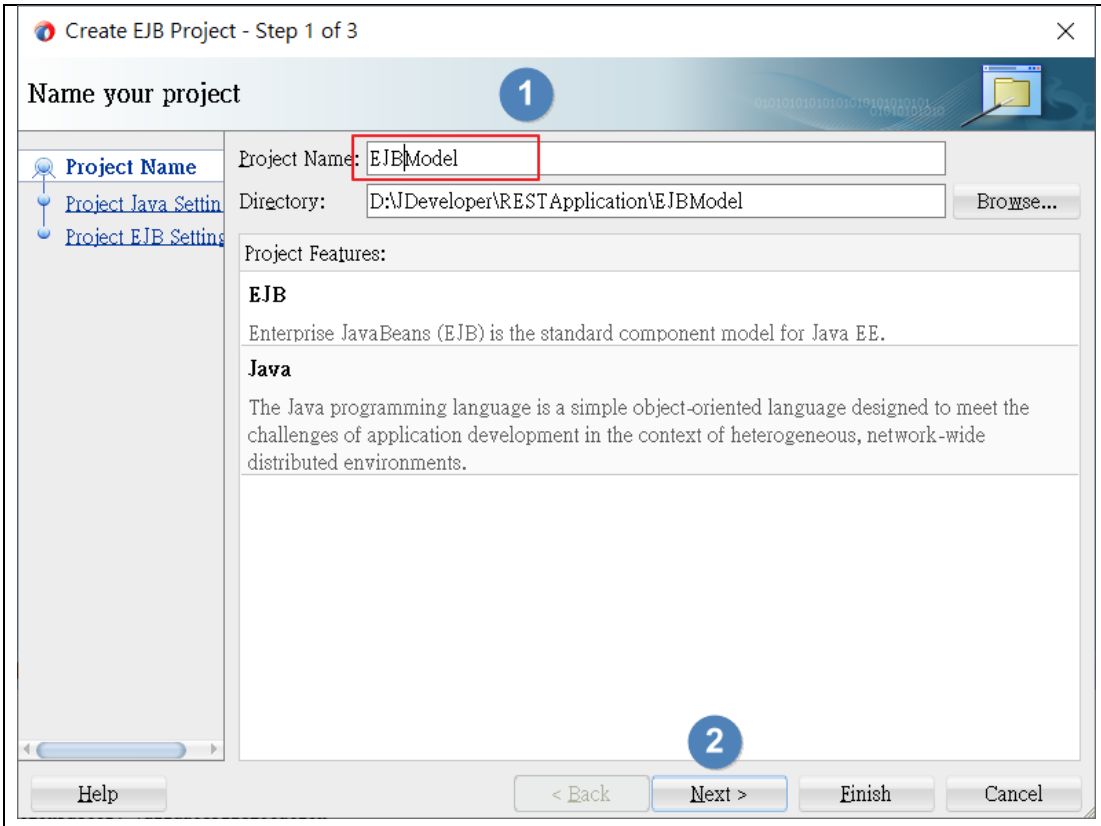


接下來我們要增設2個projects:EJBModel及RESTService
EJBModel:包含JPA/EJB元件當作bussiness service
RESTService:包含JAS-RS類別作為REST resource

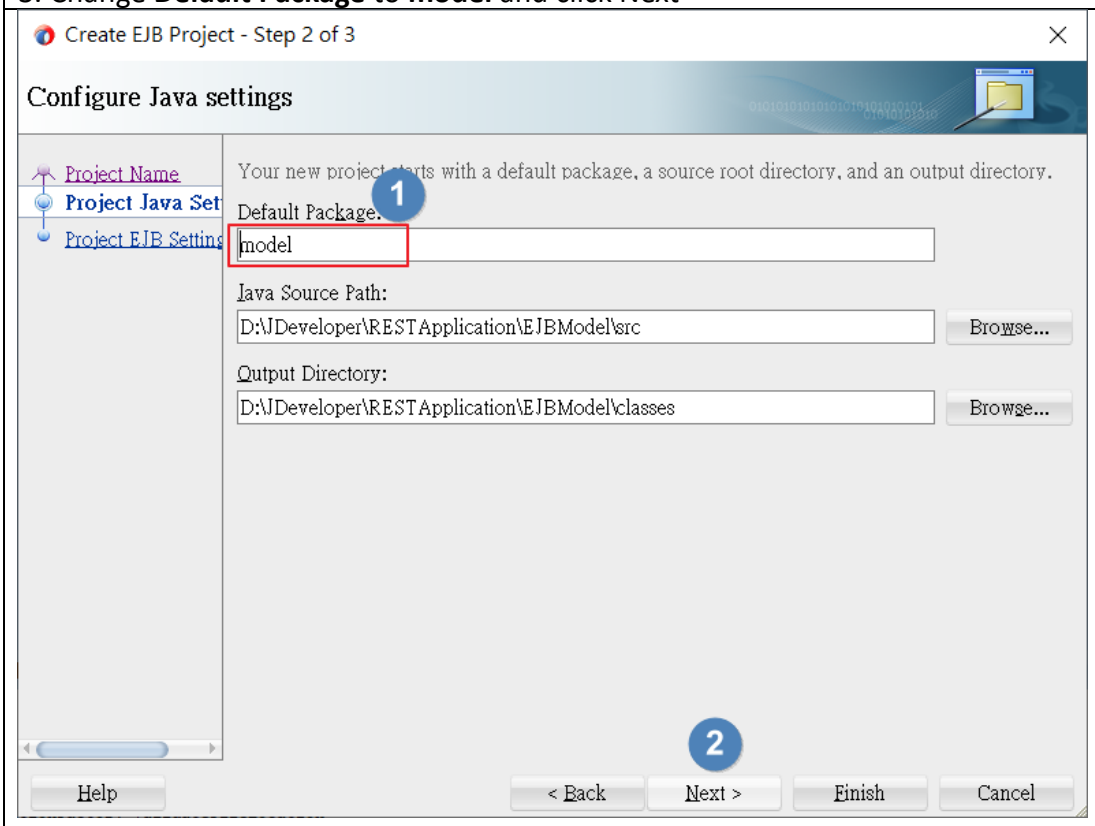
Creating Business Tier Project

1. Navigate **File>New>Project** and select **EJB Project** gallery item
2. Change **Project Name** to ***EJBModel*** and click **Next**

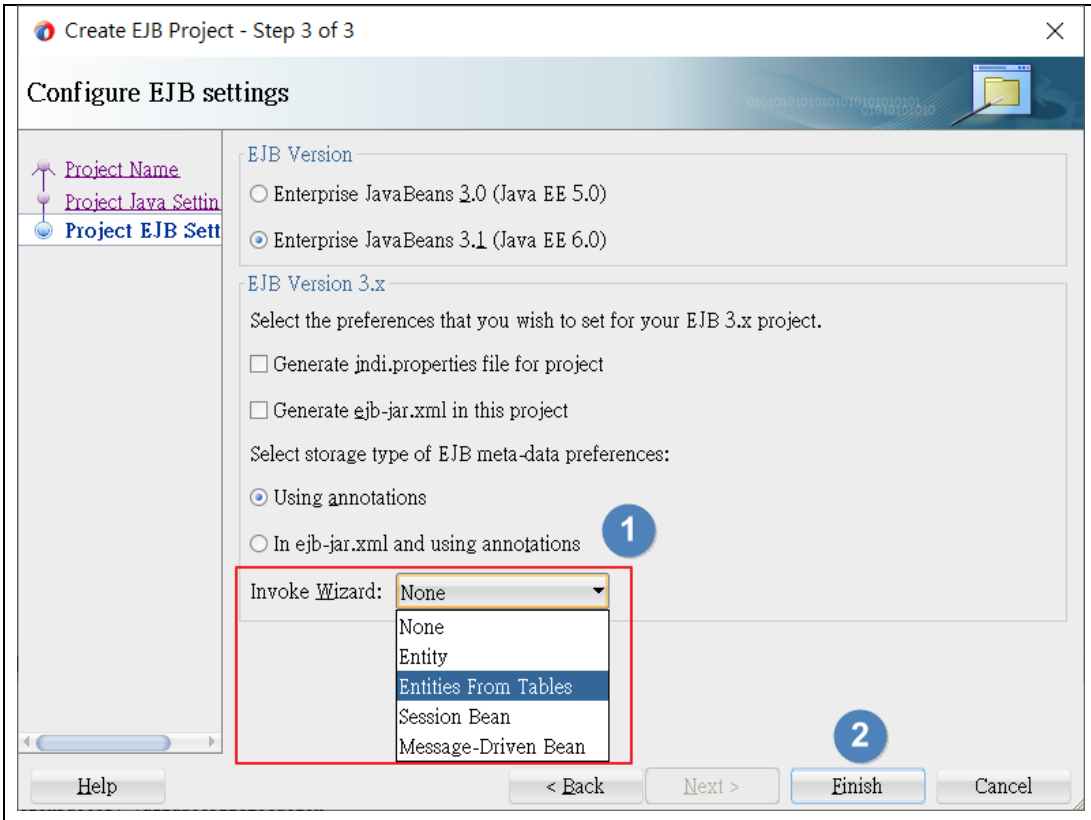




3. Change **Default Package** to **model** and click Next

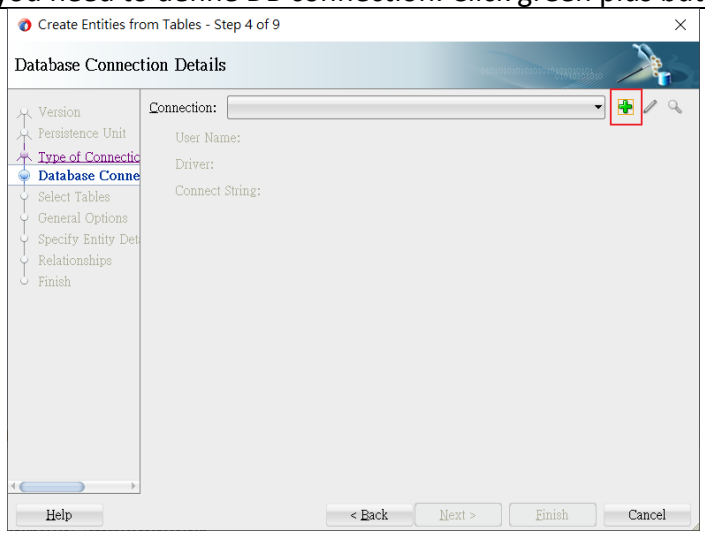


4. Change **Invoke Wizard** to **Entities From Tables**, leaving other setting by default and then click **Finish** button. Note that we are going to use **Enterprise JavaBeans 3.1** specification which corresponds to **Java EE 6** platform

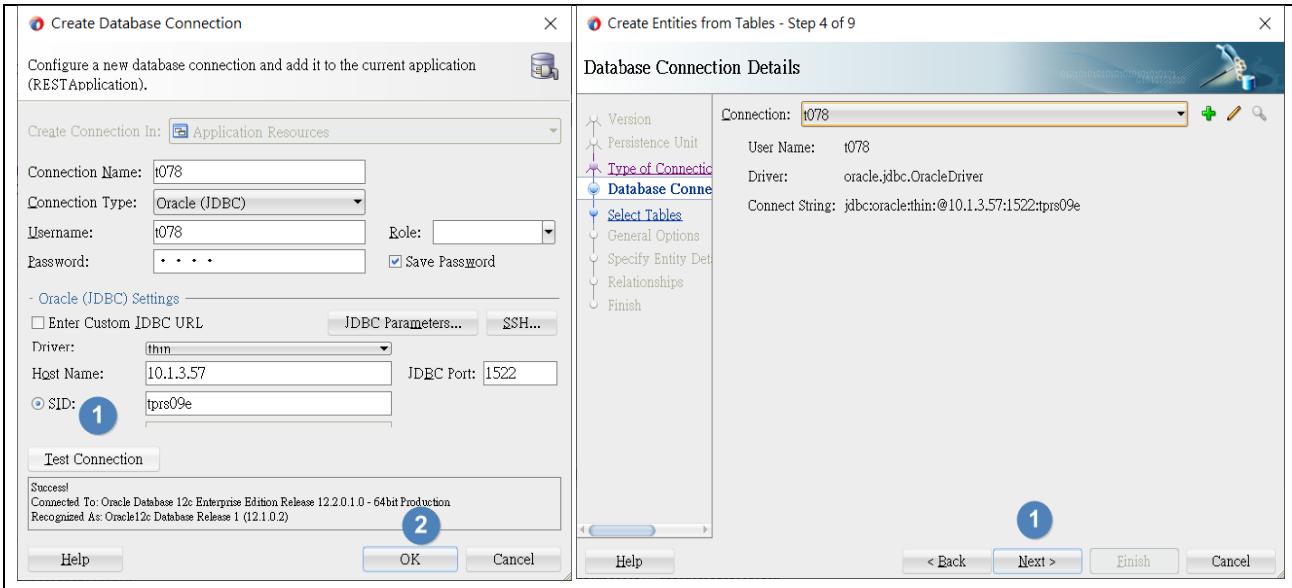


JDeveloper will create the project and automatically will start **Entities From Tables** Wizard

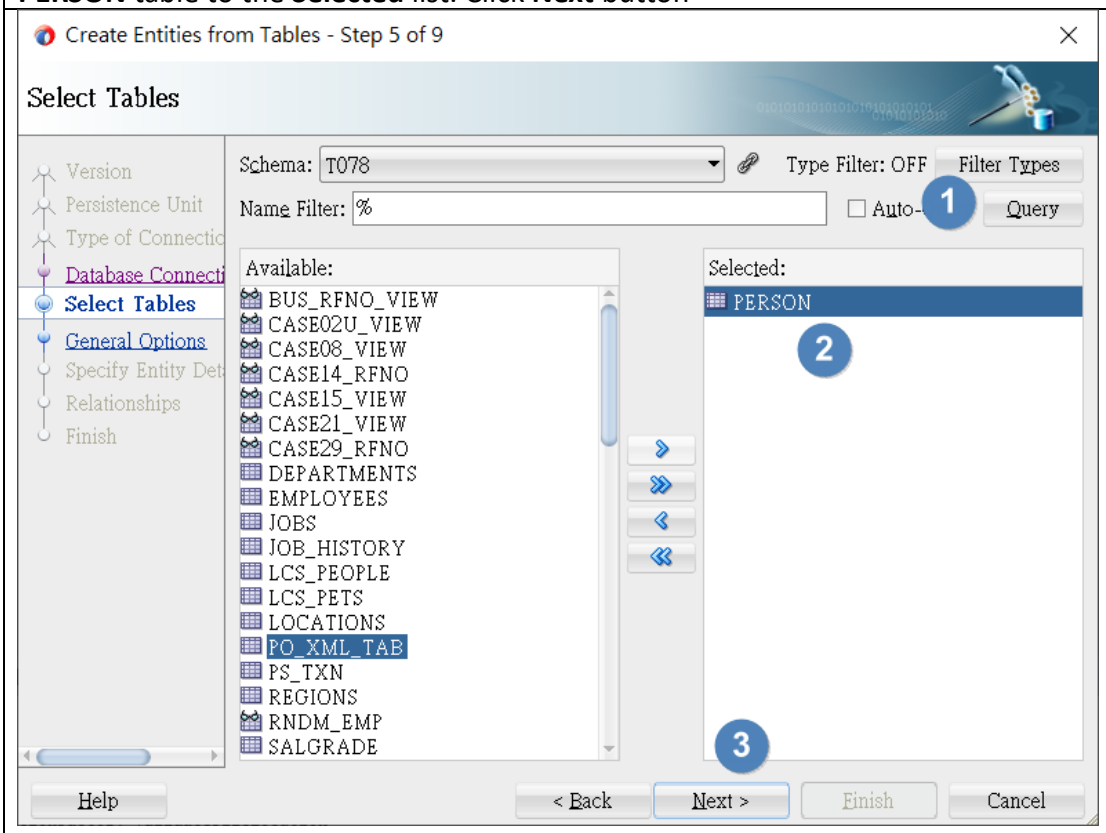
5. In **Entities From Tables** Wizard leave settings in steps **2** and **3** by default and move to the step **4** where you need to define DB connection. Click green plus button and define DB connection



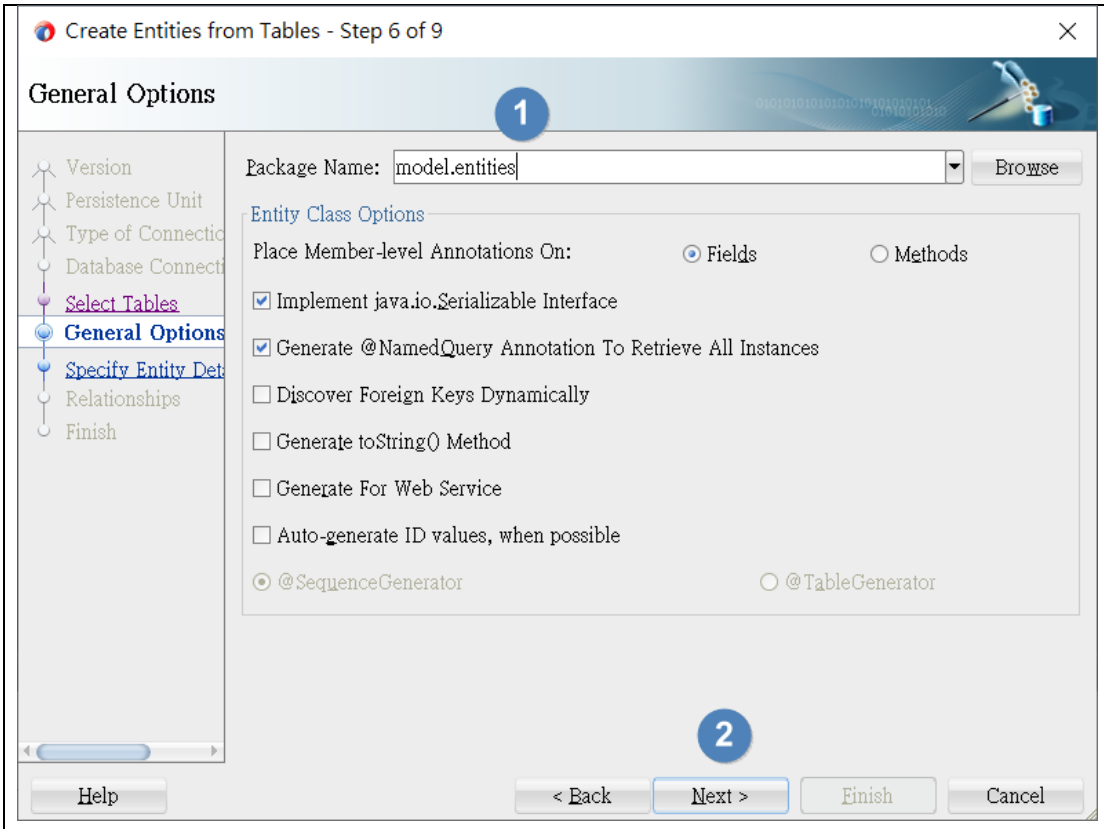
6. Enter DB Connection settings in order for <程式員帳號> schema connection and then click **OK** button in **Create Database Connection** window. Click **Next** and move to step **5** of the Wizard

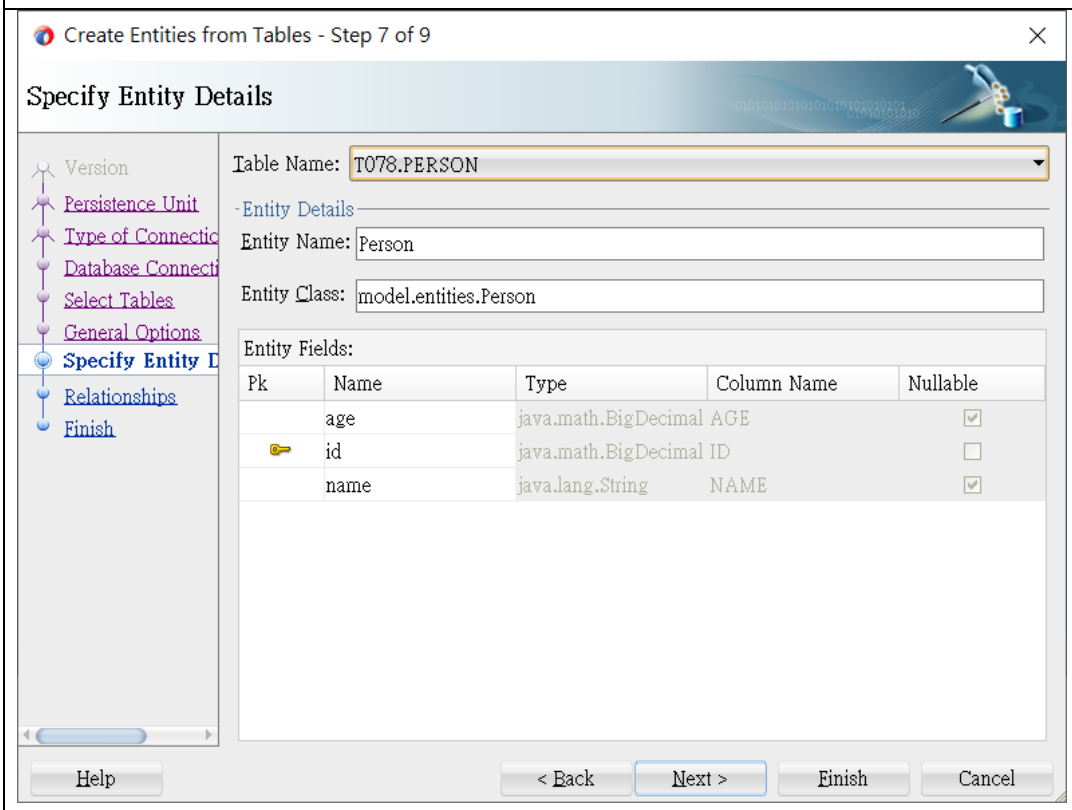


7. Now we need to define the DB tables which will be used for entities creation. In our simple case it will be only 1 table - **PERSON**. Click **Query** button to get the whole list of tables and views, then shuttle **PERSON** table to the **Selected** list. Click **Next** button

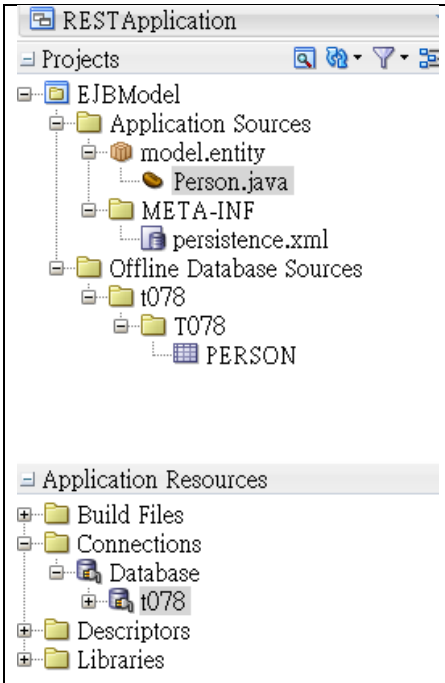


8. On the next step, change **Package Name** to **model.entities** and click **Next** button

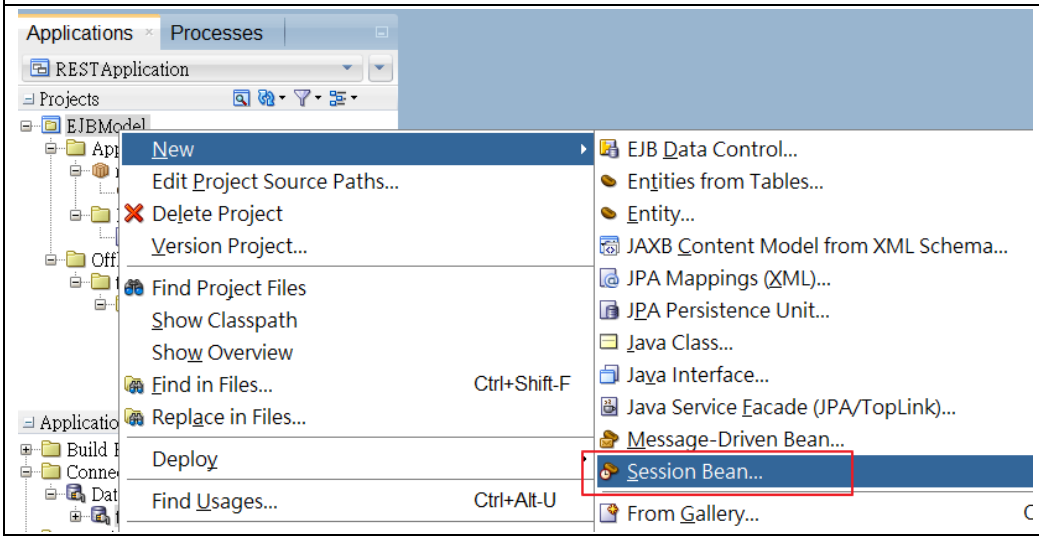


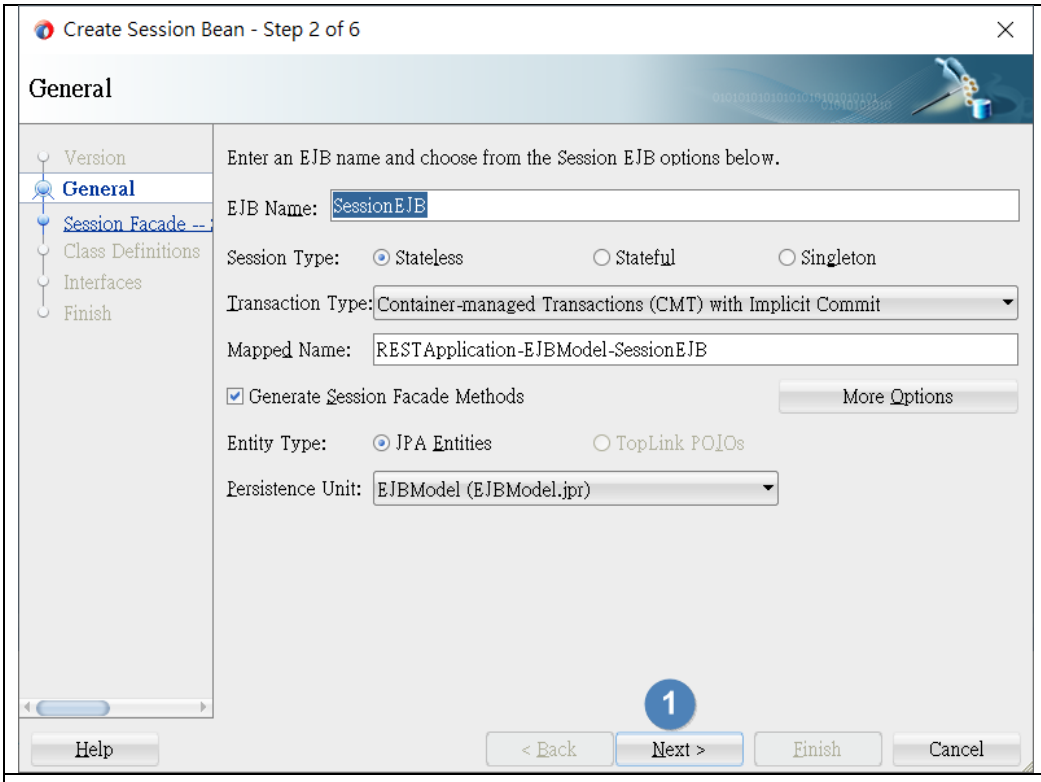


9. On the next steps of the wizard leave other settings by default and finally click **Finish** button to create our entity. After creation process projects tree should look like this:

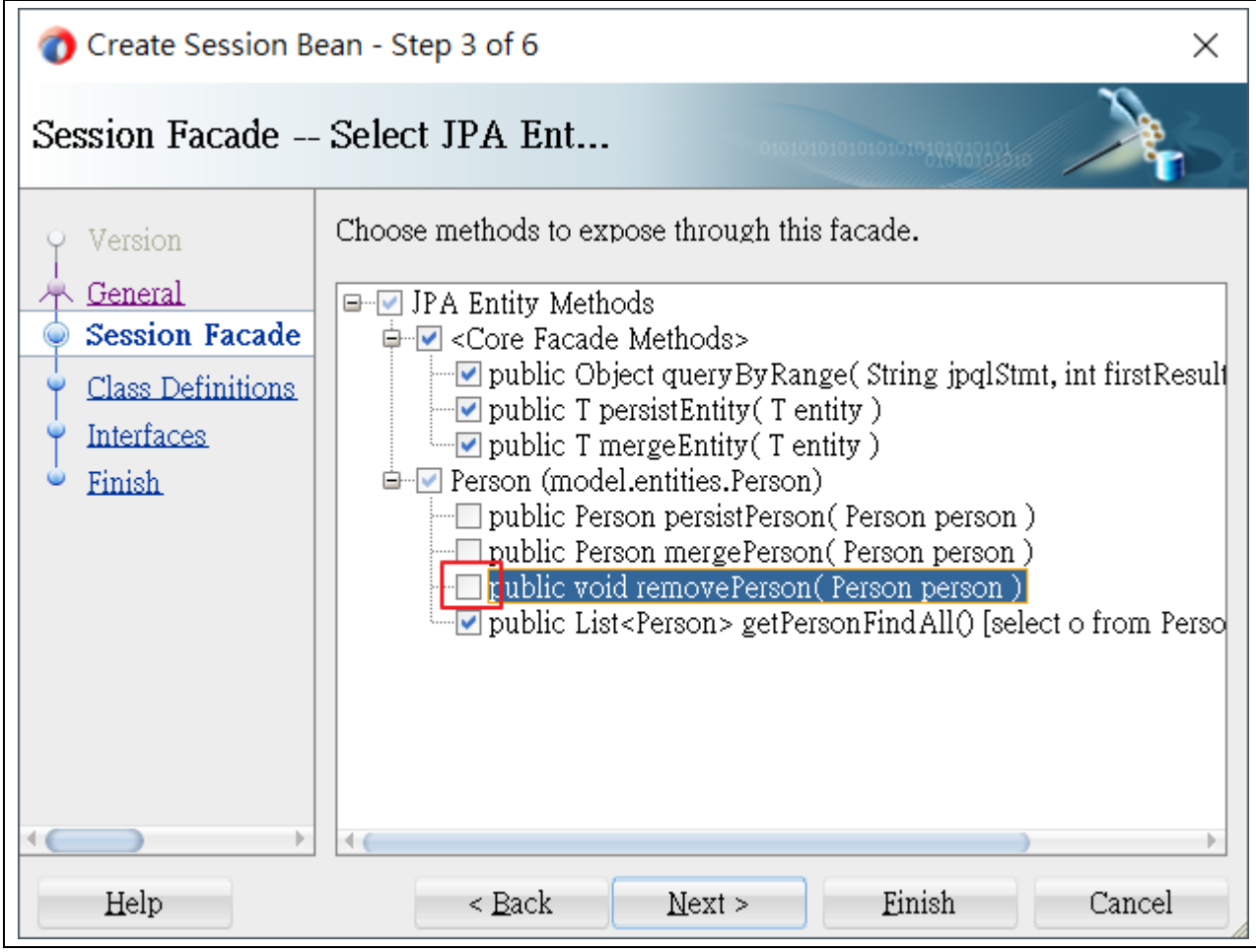


10. Now we need to create EJB Session Bean in our business tier. Right click on **EJBModel** Project **New>Session Bean**… and **Create Session Bean** Wizard will be opened. As we are going to use **Stateless** bean (a stateless session bean does not maintain a conversational state with the client), so leave all settings by default and click **Next** button



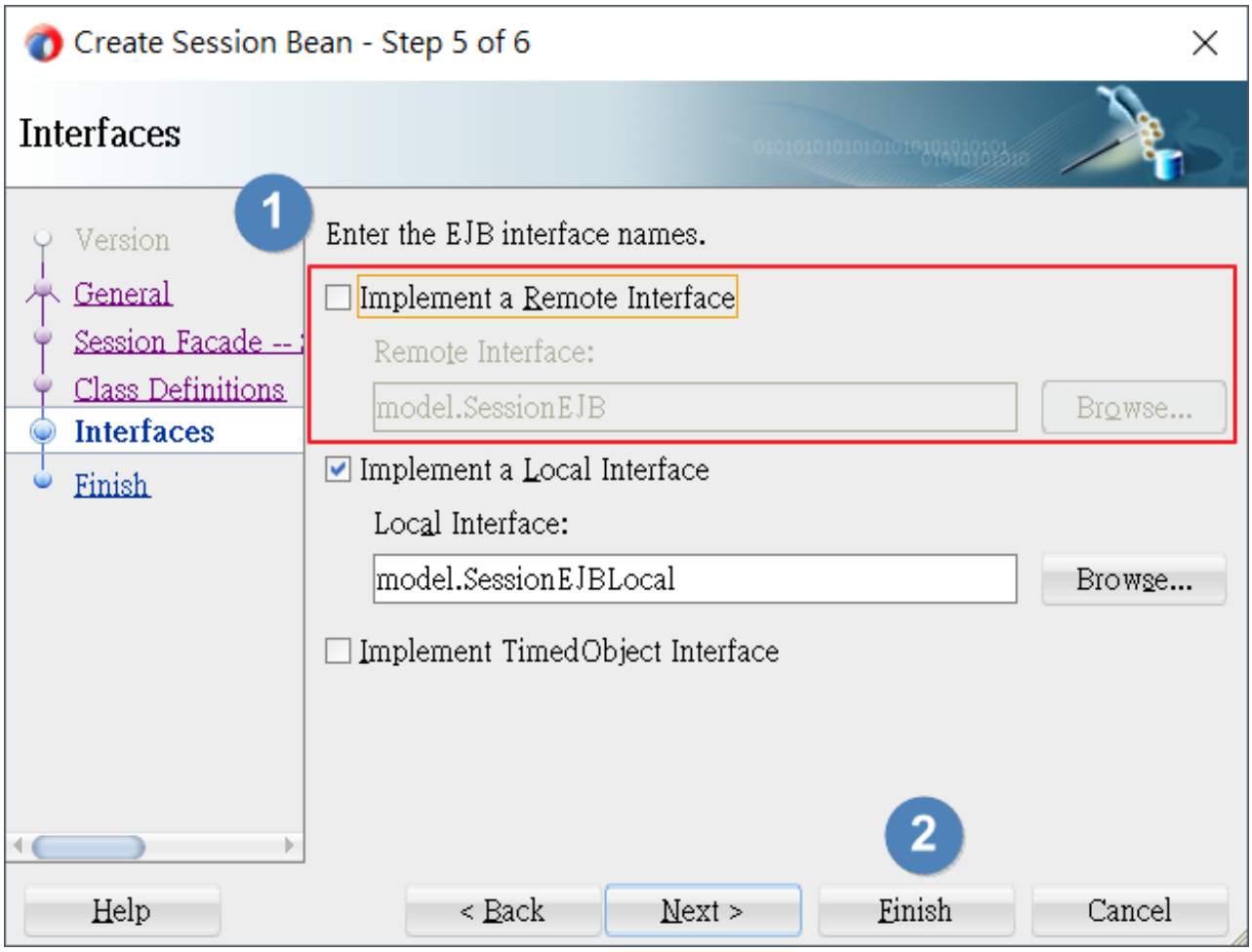


11. On the Step 3 uncheck **removePerson** method. We do not need to expose it through facade. Leave other items by default and click **Next** button

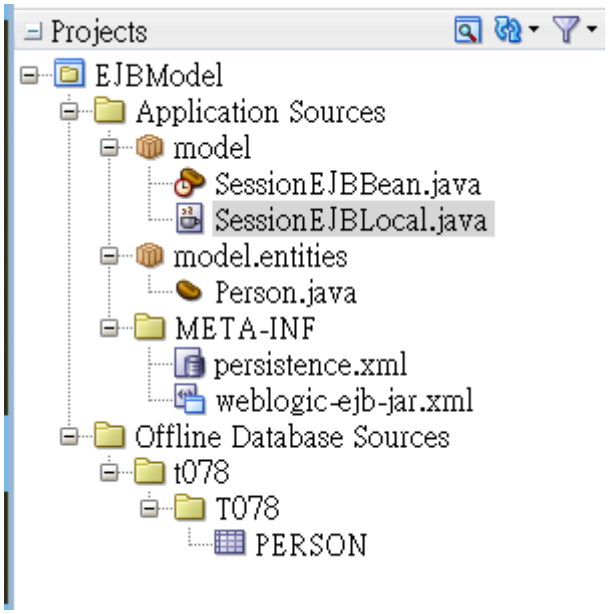


12. click **Next** button

13. On the next step uncheck **Implement Remote Interface**. In our case we are not going to use remote clients for our bean so we do not need remote interface. Click **Finish** to complete the Wizard



14. Projects tree will look like on figure below after creation process will be completed:

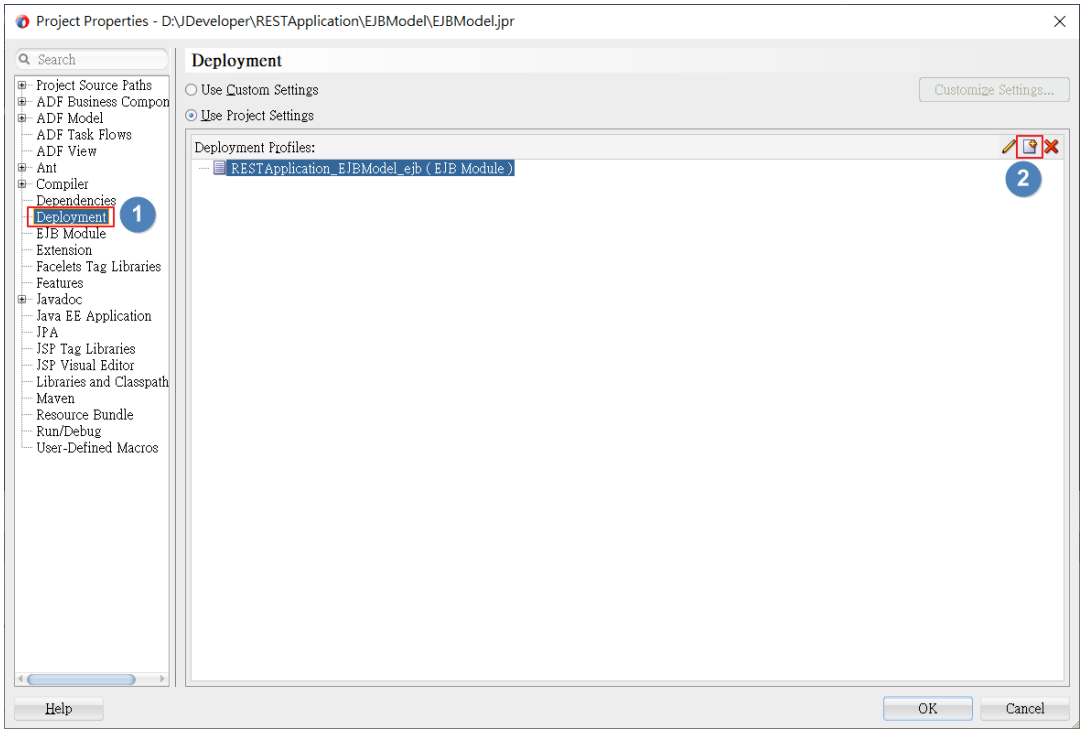


15. Open **Person.java** class in the code editor (double click on it in the projects tree). Now we need to add some changes in code. In order for list of type <Person> can be unmarshalled into a XML format, we need to add some JAXB annotations. Add certain annotations (@XmlRootElement, @XmlElement) in Person.java class according to the code snippet below:

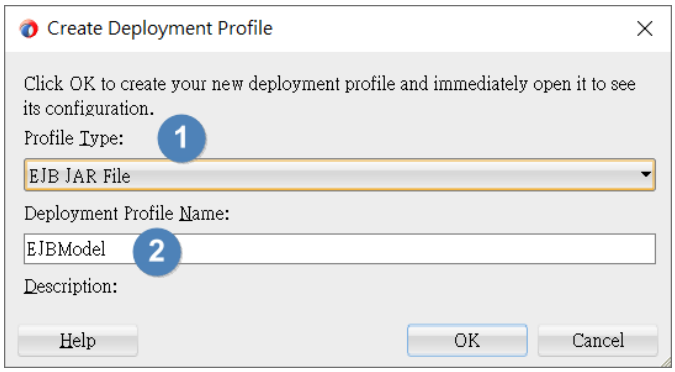
```
@Entity
@NamedQueries({ @NamedQuery(name = "Person.findAll", query = "select o from Person
o") })
public class Person implements Serializable {
    private static final long serialVersionUID = -3931802554027982556L;
    private BigDecimal age;
    @Id
    @Column(nullable = false)
    private BigDecimal id;
    @Column(length = 20)
    private String name;

    public Person() {}

    public Person(BigDecimal age, BigDecimal id, String name) {
        this.age = age;
        this.id = id;
        this.name = name;
    }
    ........
}
```

16. Now we need to define deployment profile for this project which will be used during packaging project. Right click on **EJBModel** project in projects tree and select **Project Properties**... menu item

17. In the **Project Properties** window navigate to **Deployment** section and click **New Profile** button



18. Select **Profile Type** as **EJB JAR File** and set Deployment Profile Name as **EJBModel**. Then click OK.



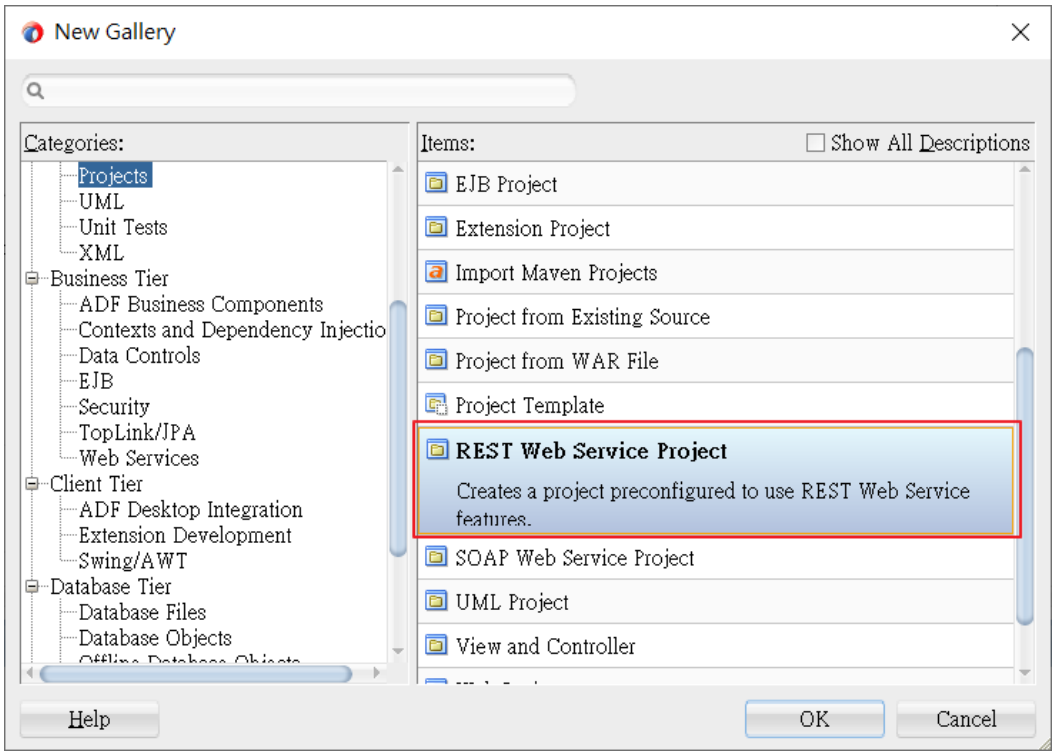
19. Click **OK** once again in **Deployment Profile Properties** window leaving all settings by default

20. Click **OK** in order to close **Project Properties** window

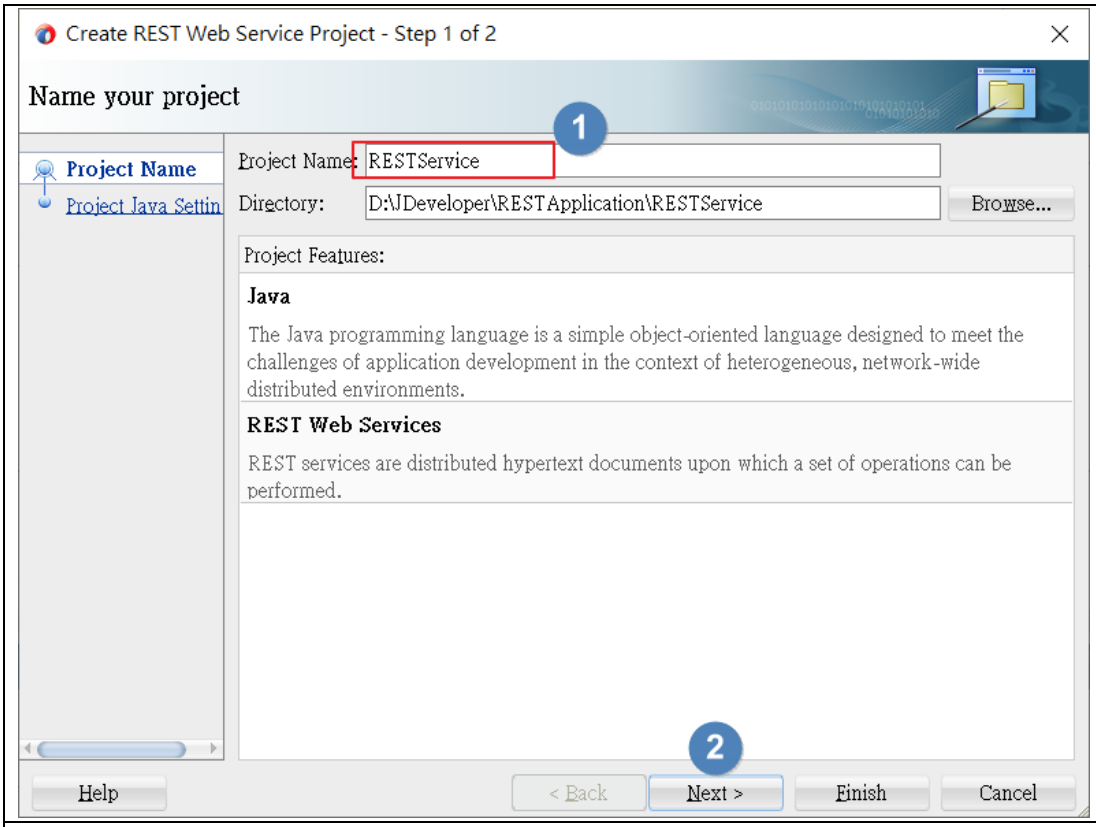
Now we are done with **EJBModel** project and business tier and it is time to move forward with REST service project.

Creating REST Service Project

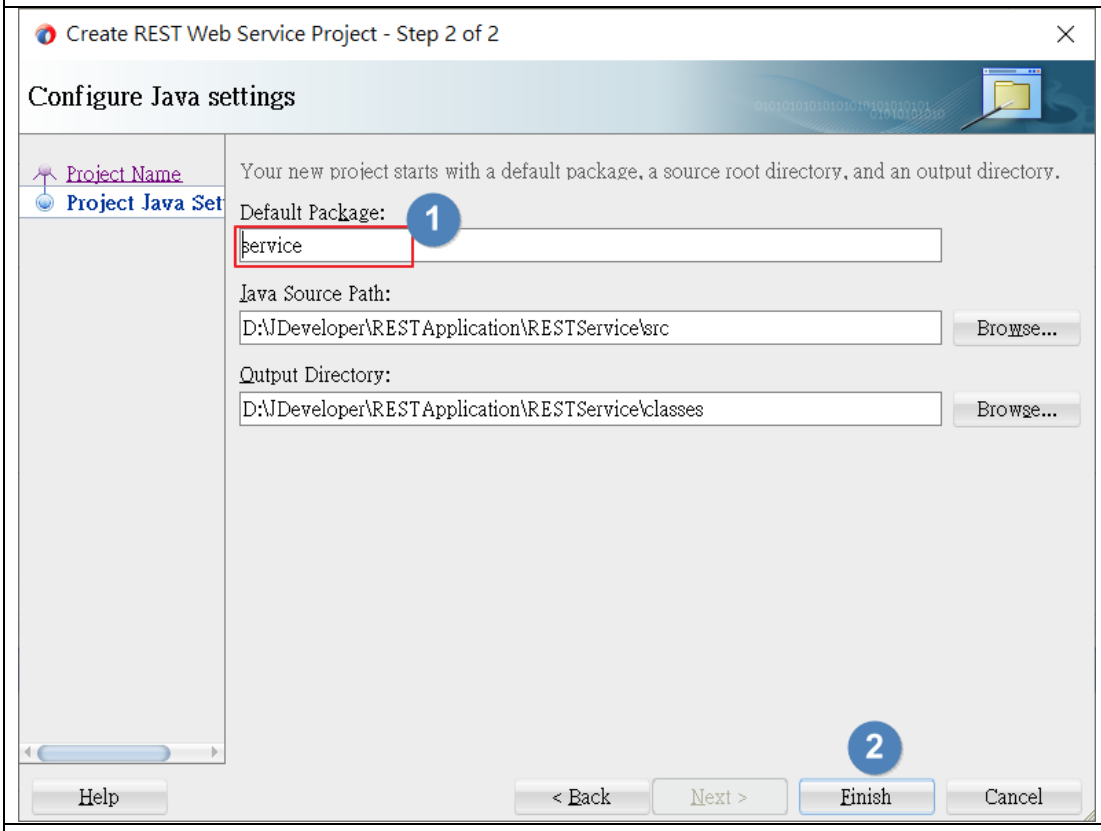
1. Navigate to **File>New>Project** item and then select **REST Web Service Project** gallery item in **New Gallery** window



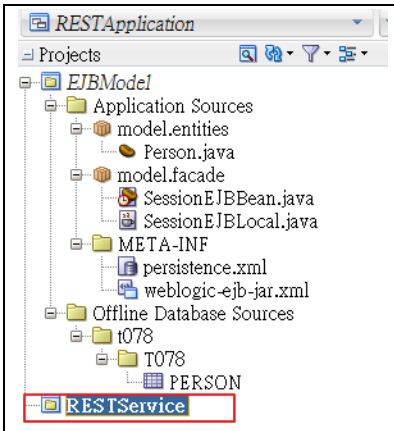
2. In **Create REST Web Service Project** Wizard set **Project Name** to **RESTService** and click **Next** button



3. Change **Default Package** to **service** and click **Finish**

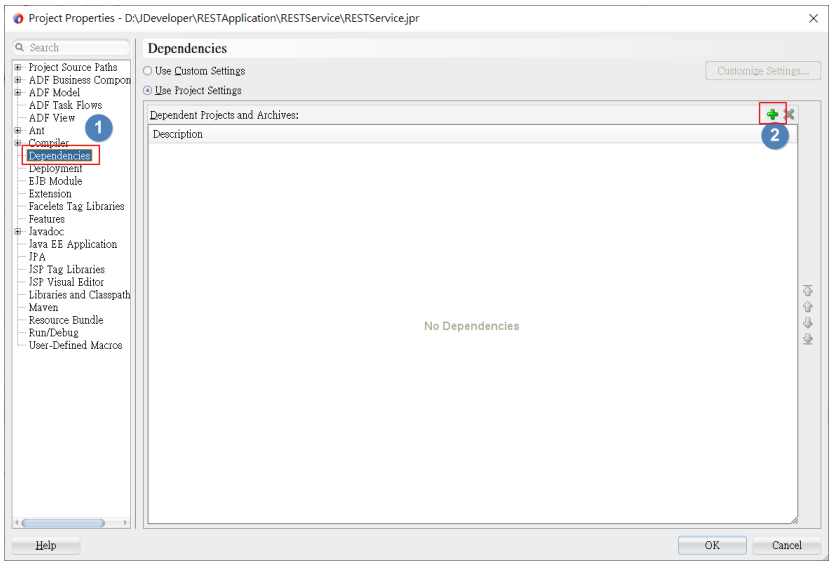


4. Empty **RESTService** project will be created and you will see it in the projects tree

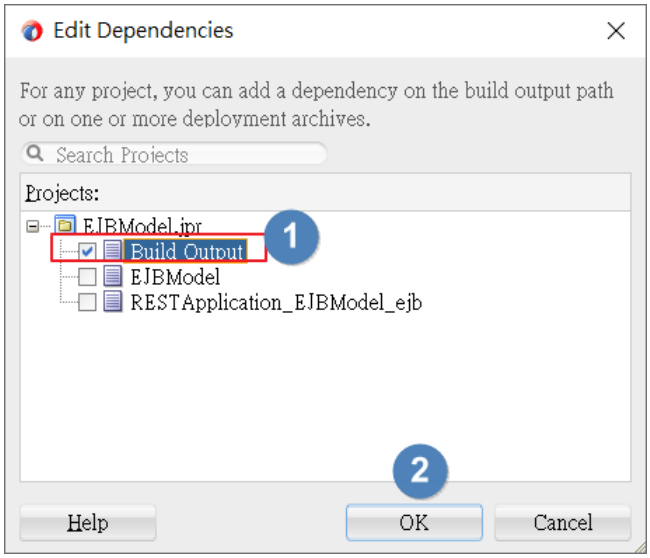


5. Now we need to define dependency for just created project. Right click on **RESTService** project in projects tree and select **Project Properties**… menu item.

6. In the **Project Properties** window navigate to **Dependencies** section and click green plus button to add dependency

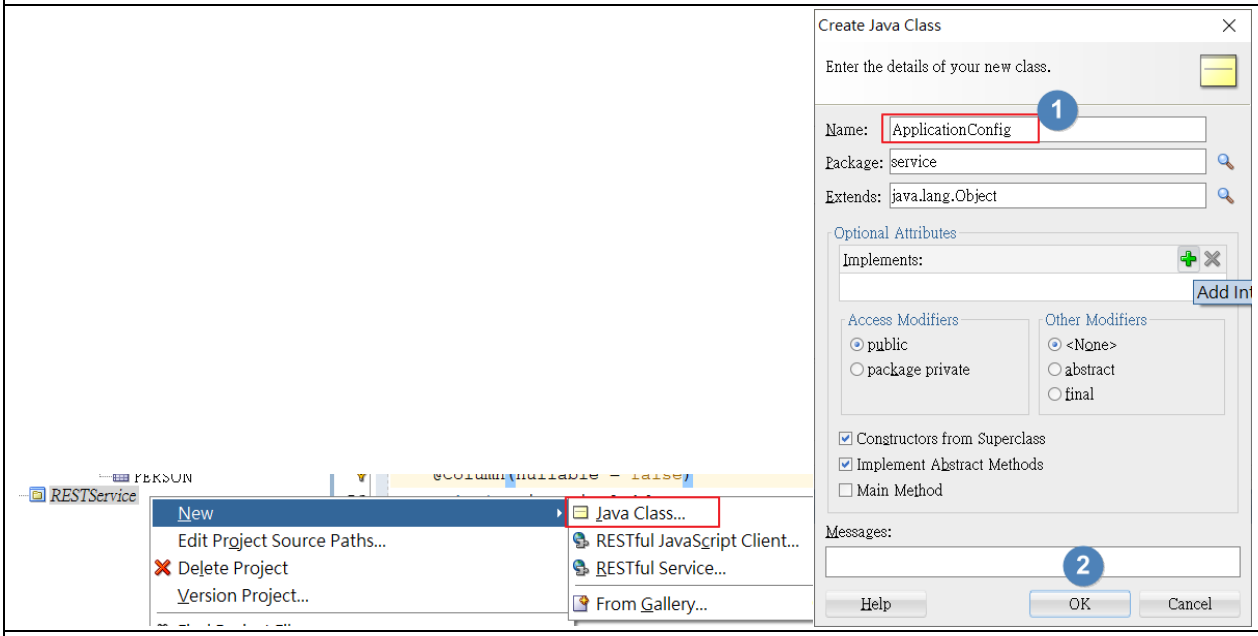


7. Expand **EJBModel.jpr** tree and select **Build Output** item. Then click **OK**.



8. Click **OK** in order to close **Project Properties** window

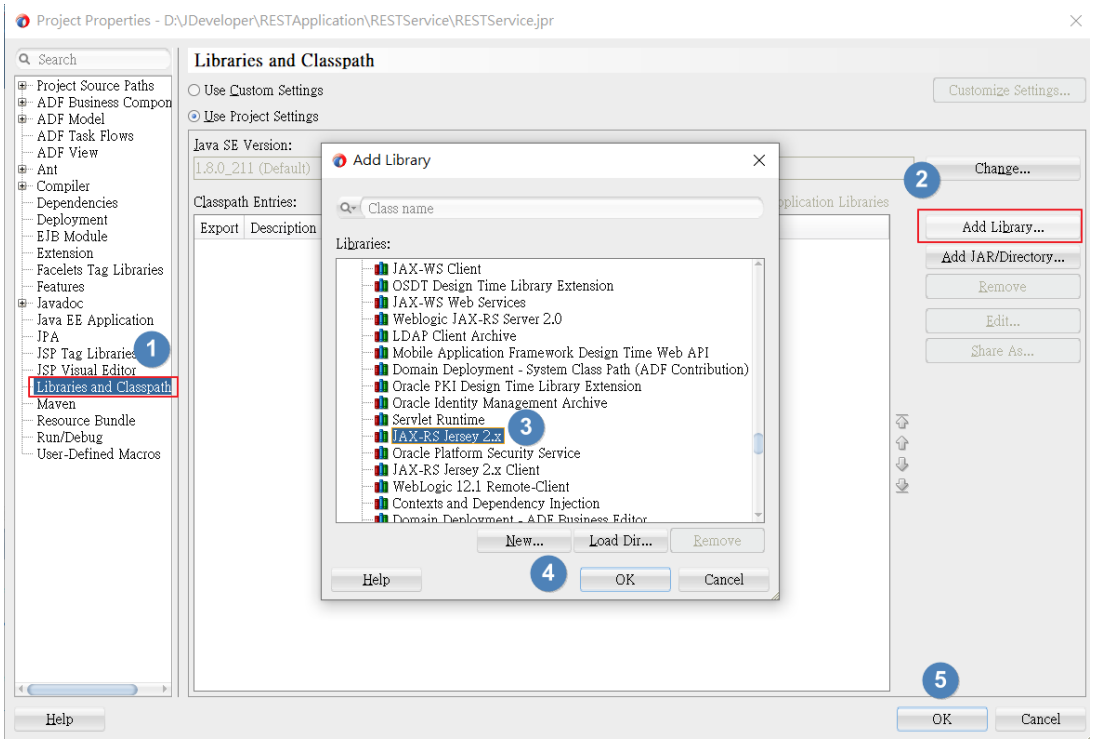
9. Let's create a Java Class named **ApplicationConfig**. Right click on the **RESTService** project and select **New>Java Class**… item. Change the name to **ApplicationConfig** and click **OK**.



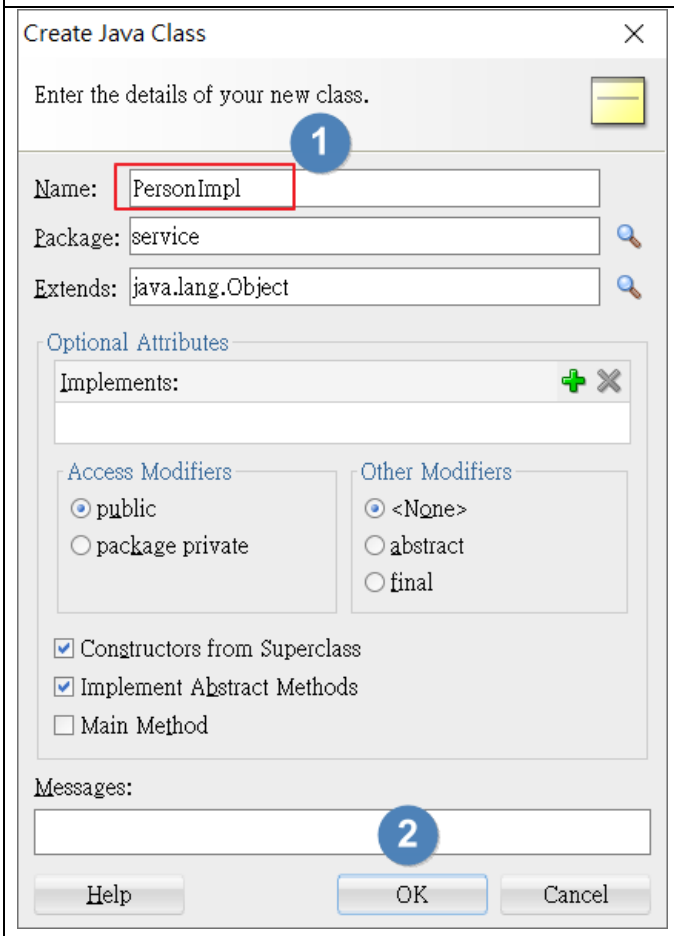
10. Replace the code in just created **ApplicationConfig** class with the code below:

```
package service;
import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;
@ApplicationPath("service")
public class ApplicationConfig extends Application {
	public ApplicationConfig() {
		super();
	}
}
```

這時候程式可能會報錯，是因為需要匯入 JAX-RS 2.X library，所以點擊 Project Properties，匯入 library



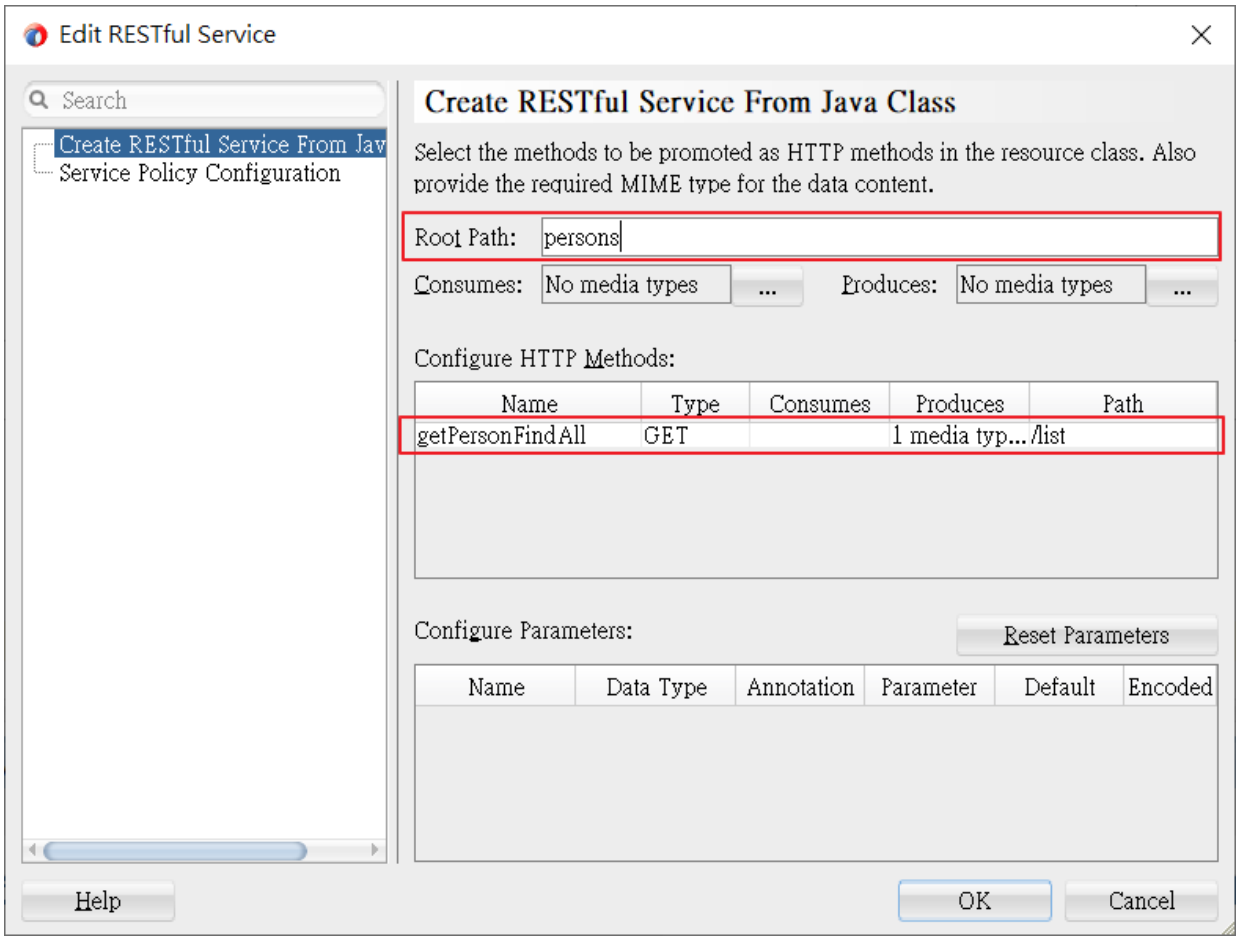
11. Create another Java class and name it as **PersonImpl**. Replace the code of just created class with the code below:



```
package service;
import java.util.List;
import javax.ejb.EJB;
import javax.ejb.Stateless;
import model.SessionEJBLocal;
import model.entities.Person;
@Stateless
public class PersonImpl {
    public PersonImpl() {
        super();
    }
    @EJB(beanName="SessionEJB")
    SessionEJBLocal mySessionBean;
    public List<Person> getPersonFindAll(){
        return mySessionBean.getPersonFindAll();
    }
}
```

12. Right click on ***PersonImpl.java*** class in the projects tree and select **Create RESTful Service**… item in the context menu

13. Set **Root Path** as **persons**, for ***getPersonFindAll*** method select **Type** as ***GET***, set **Produces** as **application/json** MIME types, set **Path** as ***/list*** in **Create REST Service** Wizard. Then click **Finish**

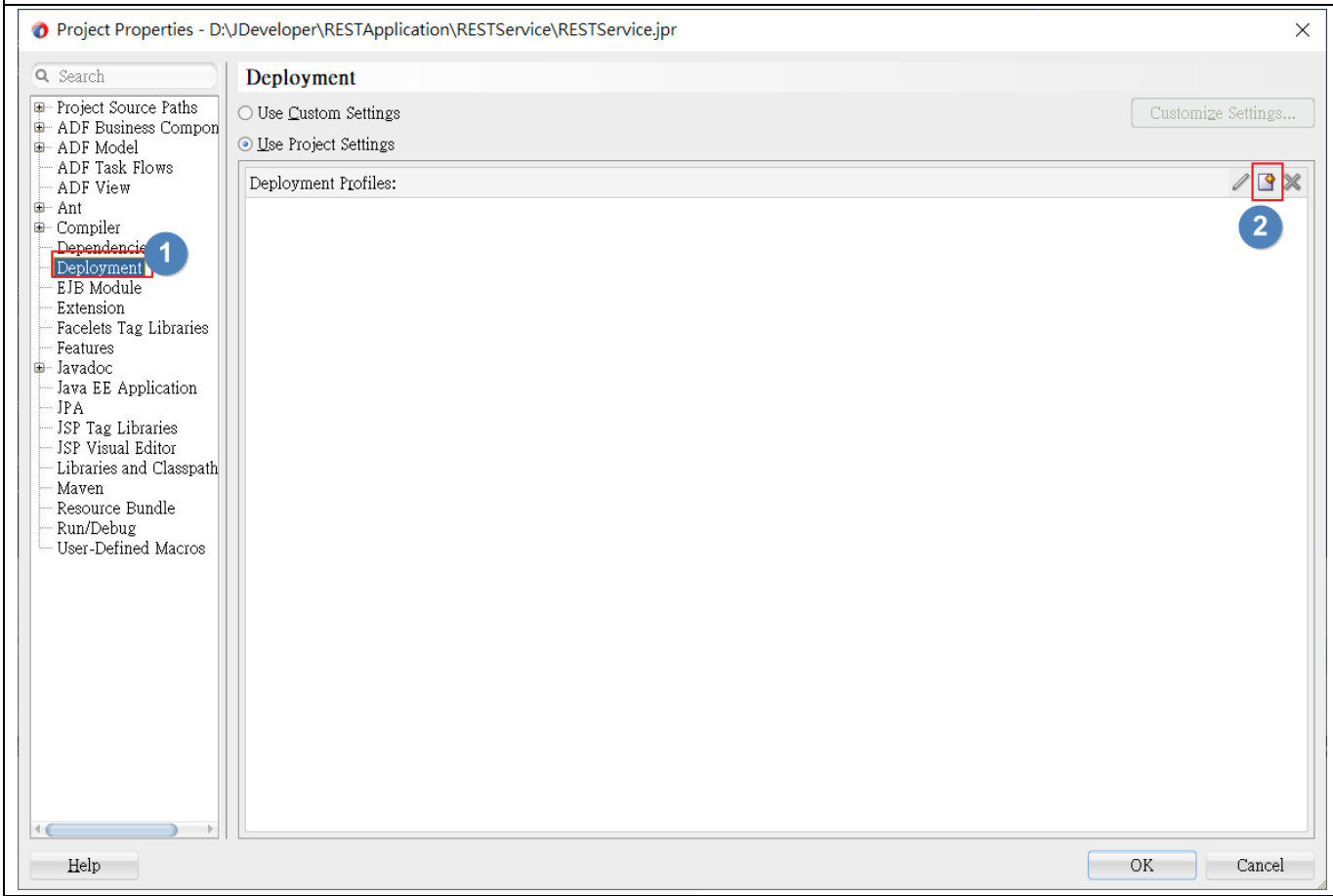


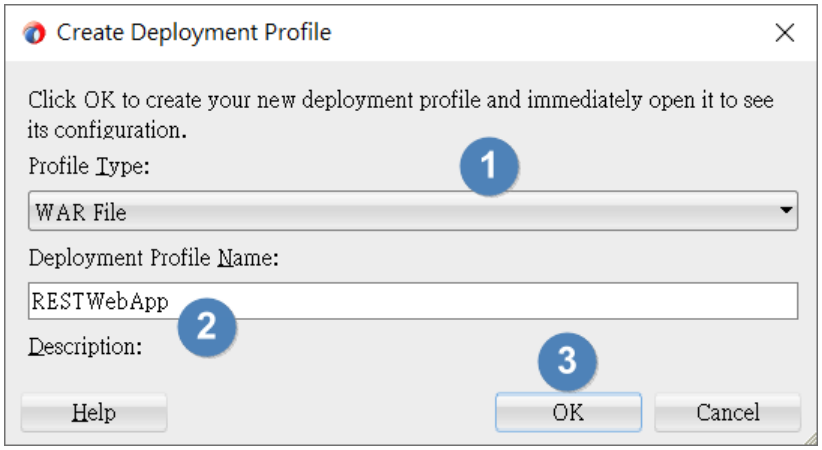
程式碼會變成:

```
package service;
import java.util.List;
import javax.ejb.EJB;
```

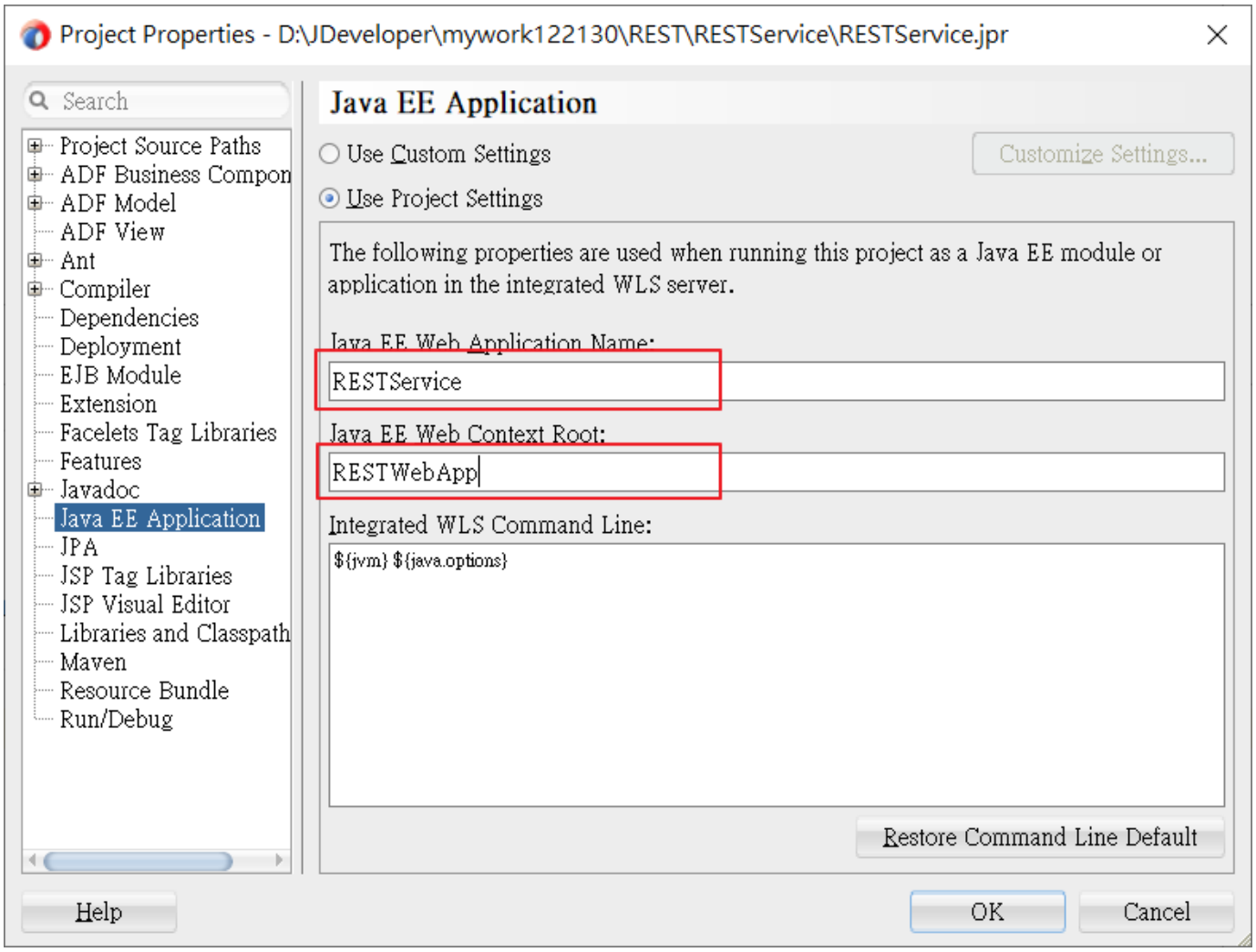
```
import javax.ejb.Stateless;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import model.SessionEJBLocal;
import model.entities.Person;
@Stateless
@Path("persons")
public class PersonImpl {
    public PersonImpl() {
        super();
    }
    @EJB(beanName="SessionEJB")
    SessionEJBLocal mySessionBean;

    @GET
    @Produces("application/json")
    @Path("/list")
    public List<Person> getPersonFindAll(){
        return mySessionBean.getPersonFindAll();
    }
}
```

14. Now we need to define deployment profile for **RESTService** project which will be used during packaging project within application archive. Right click on project in projects tree and select **Project Properties**… menu item. In the **Project Properties** window navigate to **Deployment** section and create new deployment profile with **WAR File** profile type. Set profile name as **RESTWebApp**





15. Now we need to define some specific **Java EE Application** properties for the project. Navigate to the ***RESTService*** project properties and set **Java EE Web Application Name** as ***RESTWebApp*** and **Java EE Web Context Root** as person

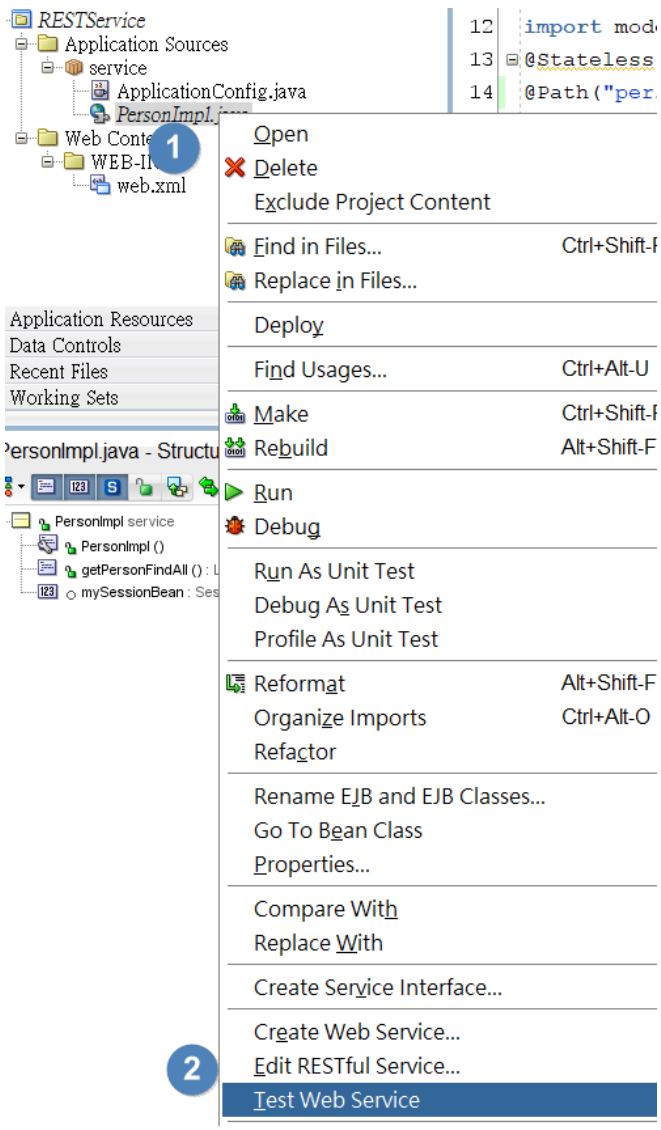


16. Our REST service project is ready

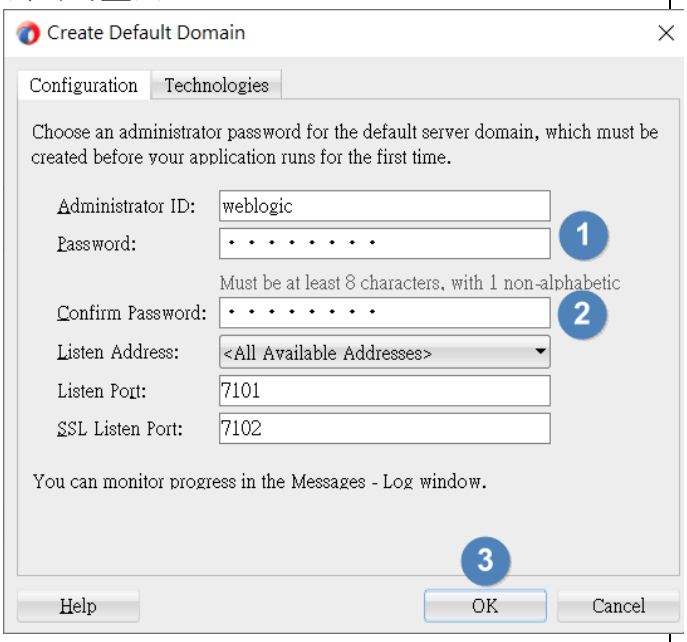
程式佈署(Deploy)

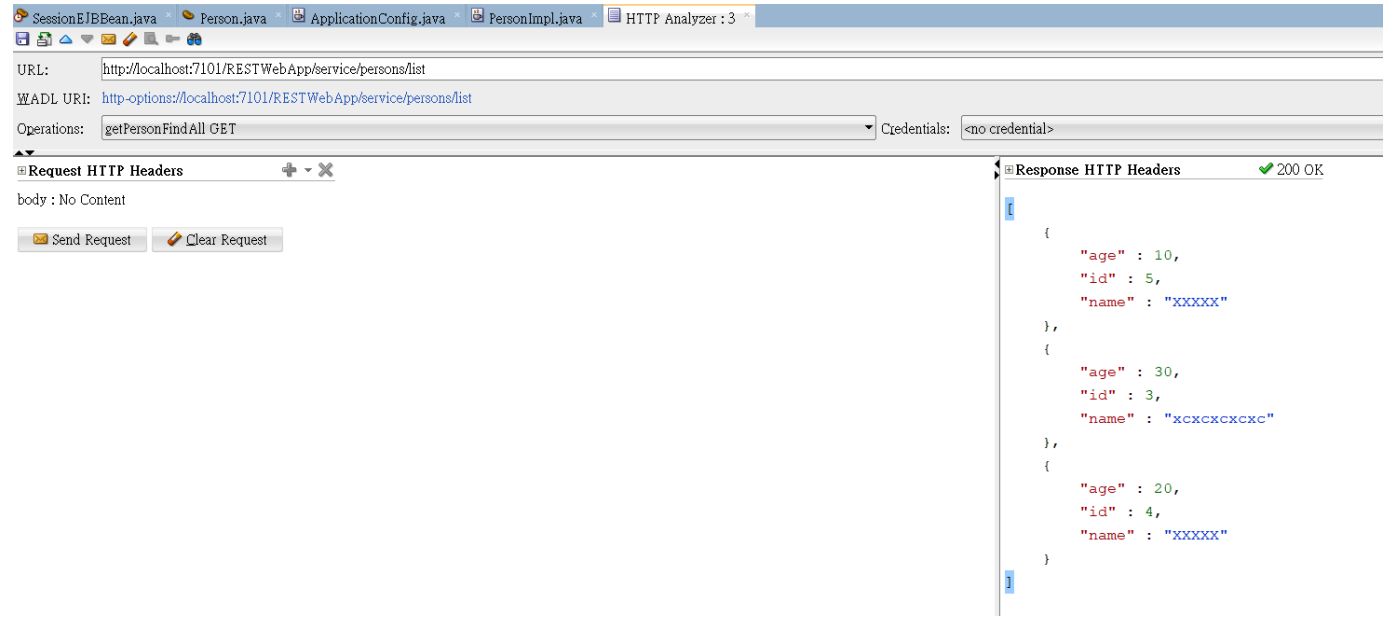
In order to deploy our application we need to create application deployment profile and add there our 2 projects (**EJBModel** and **RESTService**) created earlier. Navigate to application properties in JDeveloper (**Application>Application Properties**) and create a new deployment profile. Set **Profile Type** as **EAR** File and name as **RESTApplication**.

執行測試



第一次執行可能會出現啟動 Weblogic 需要設定密碼的畫面。

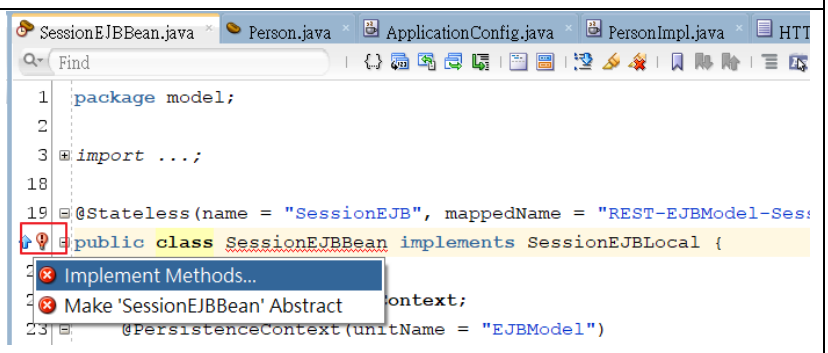




進階功能實作:

1.條件查詢(select * from person where id=?)

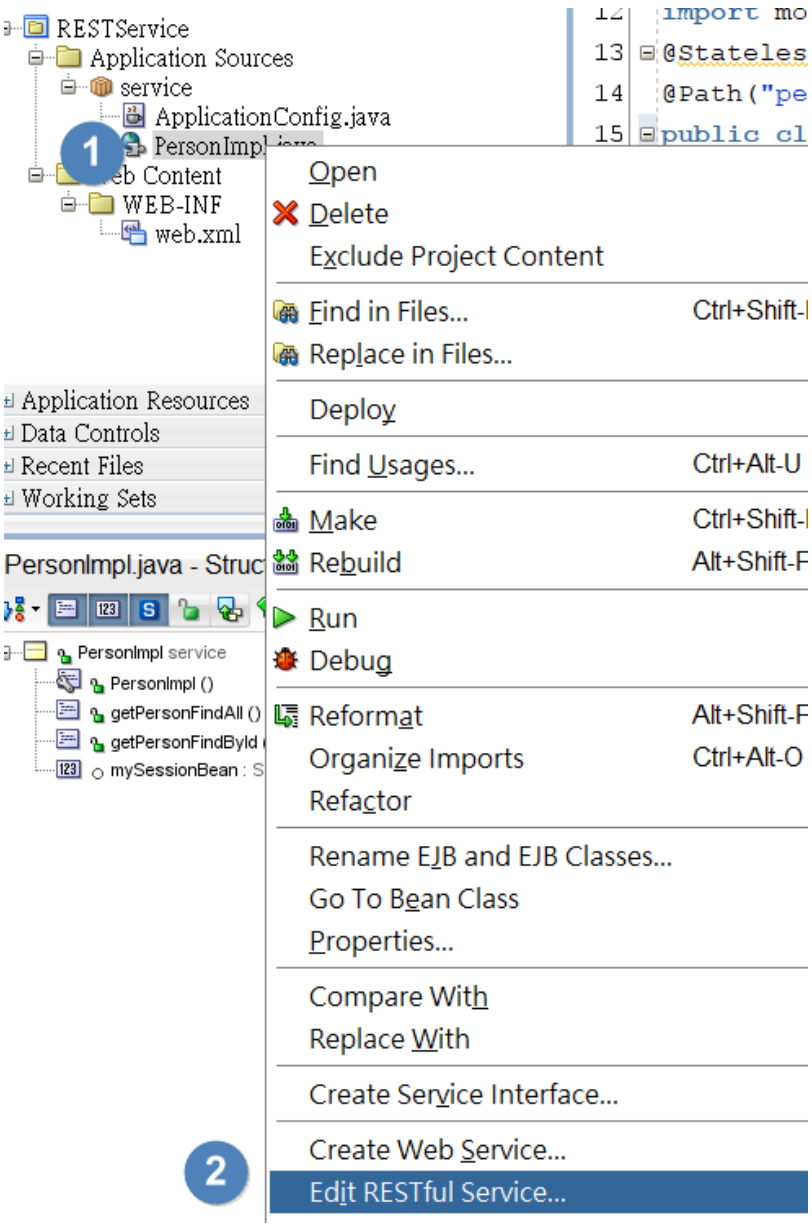
在 Person.java 增設一個 @NameQuery， @NameQueries 內容如右邊程式片段。	@NamedQueries({ @NamedQuery(name = "Person.findAll", query = "select o from Person o"), @NamedQuery(name = "Person.findById", query = "select o from Person o where o.id = :id") })
在 SessionEJBLocal.java 宣告一個 getPersonFindById 的方法	Person getPersonFindById(BigDecimal id);

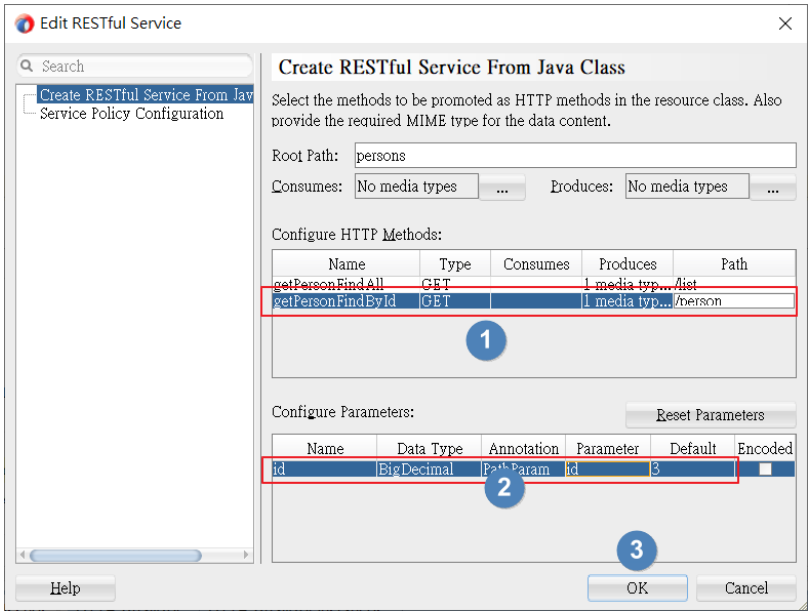
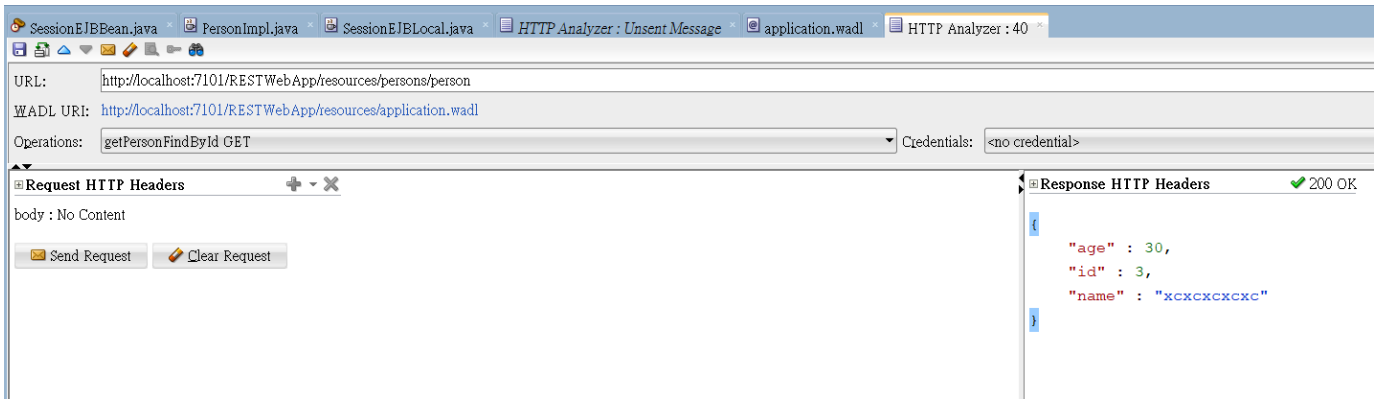
在 SessionEJBBean.java 實作上面宣告的方法	
	@Override
	public Person getPersonFindById(BigDecimal id) {
	// TODO Implement this method
	return null;
	}

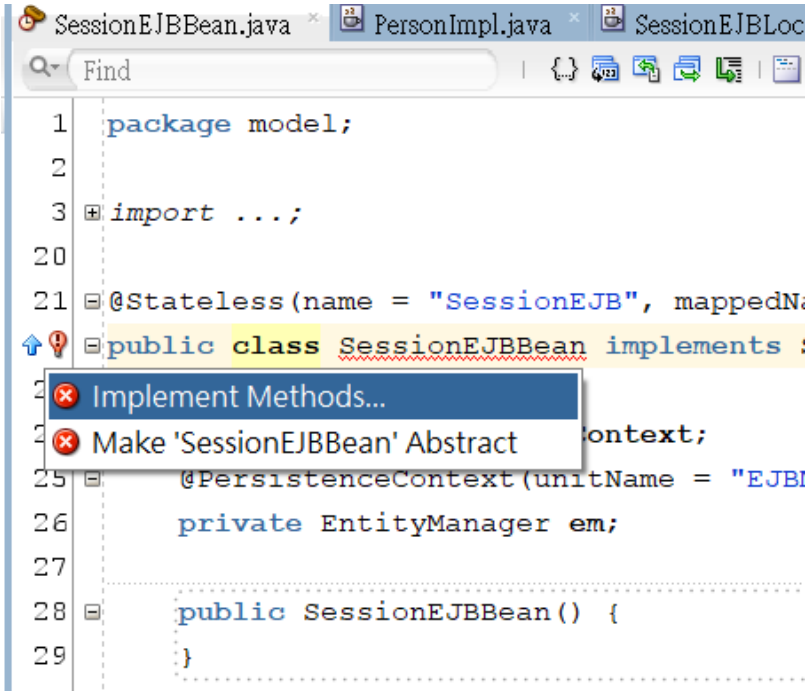
實作後的程式碼如右	@Override
	public Person getPersonFindById(BigDecimal id) {
	// TODO Implement this method

	return em.createNamedQuery("Person.findById", Person.class).setParameter("id", id).getSingleResult();
	}
	}

在 PersonImpl.java 宣告 Service 使用的方法	public Person getPersonFindById(BigDecimal id){
	return mySessionBean.getPersonFindById(id);
	}

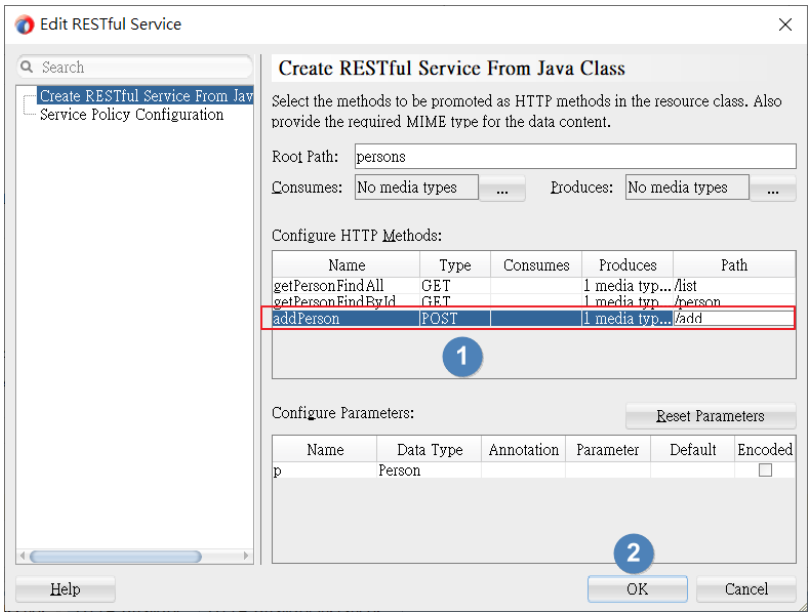
	修改屬性 

Type:GET Produces:Application/json Path:/person Annotation:Query Param Parameter:id Default:3	
執行測試	
	
2.新增資料(insert into person values(?,?,?))	
在 SessionEJBLocal.java 宣告一個 addPerson 的方法	Response addPerson(Person p);

<table>
<tr>
<td>在 SessionEJBBean.java 實作上面宣告的方法</td>
<td>


@Override
public Response addPerson(Person p) {
// TODO Implement this method
return null;
}
</td>
</tr>
<tr>
<td>實作後的程式碼</td>
<td>
@Override
public Response addPerson(Person p) {
em.createNativeQuery("INSERT INTO person (id, age, name) VALUES (?,?,?)")
.setParameter(1, p.getId())
.setParameter(2, p.getAge())
.setParameter(3, p.getName())
.executeUpdate();
String resp = "{\"message\":\"資料已新增\",\"status\":\"" + Response.Status.CREATED + "\"}";
return Response.status(Response.Status.CREATED)
.entity(resp)
.build();
}
</td>
</tr>
<tr>
<td>因為要 Response Payload 給呼叫端，須 import javax.ws.rs.core.Response，所以須將 JAX-RS 2.X library 納入 EJBModel project</td>
<td></td>
</tr>
</table>

在 PersonImpl.java 宣告 addPerson 方法，return Response Payload	public Response addPerson(Person p){
	return mySessionBean.addPerson(p);
	}

設定 RestFul Service Type:POST Produces:application/json Path:/add	
執行測試	

POST http://localhost:7101/RESTWebApp/resources/persons/add

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "age" : 77,
3     "id" : 7,
4     "name" : "李超人"
5 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2     "message": "資料已新增",
3     "status": "Created"
4 }
```

URL: http://localhost:7101/RESTWebApp/resources/persons/person?id=7

WADL URI: http://localhost:7101/RESTWebApp/resources/application.wadl

Operations: getPersonFindById GET Credentials: <no credential>

Request HTTP Headers

Parameters

id : string 7 Include

body : No Content

Send Request Clear Request

Response HTTP Headers 200 OK

```
{
    "age" : 77,
    "id" : 7,
    "name" : "李超人"
}
```