Tutorial for simple JSF example

Inhaltsverzeichnis

[1. Setup Projects 2](#_Toc475621589)

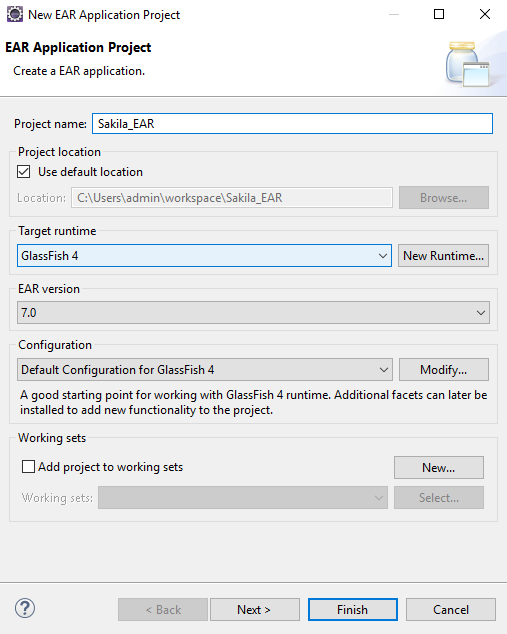
[2. Connecting the projects 6](#_Toc475621590)

[3. Code 8](#_Toc475621591)

# Setup Projects

First let’s setup a new EAR project

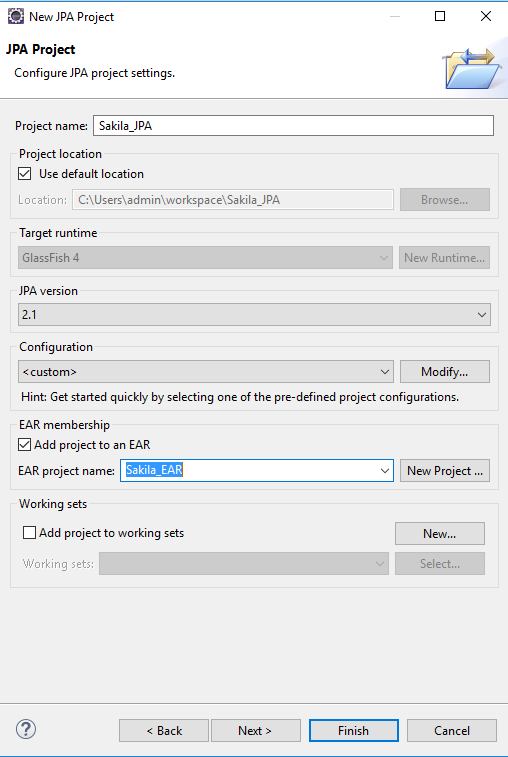
New-> Enterprise Application Project



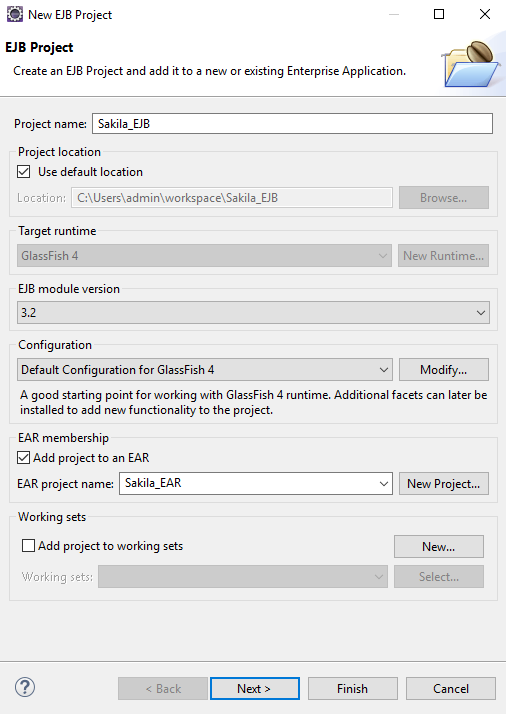
Next a JPA project and make sure you add it to the EAR

If there is any Pop-Up press “No”

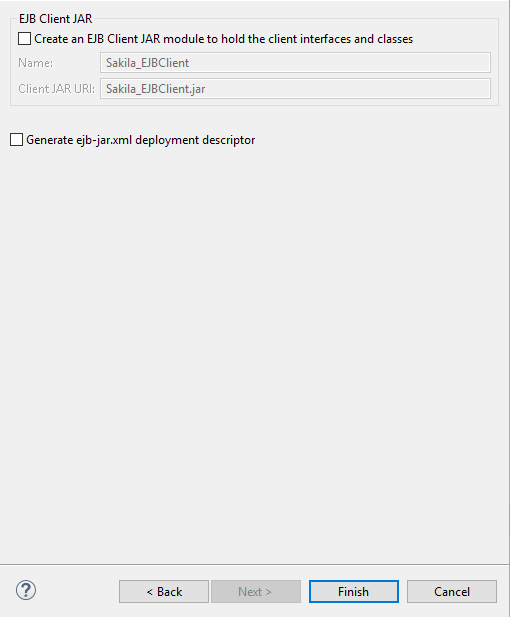
New -> Other -> JPA -> JPA Project



Next an EJB project and make sure you add it to the EAR

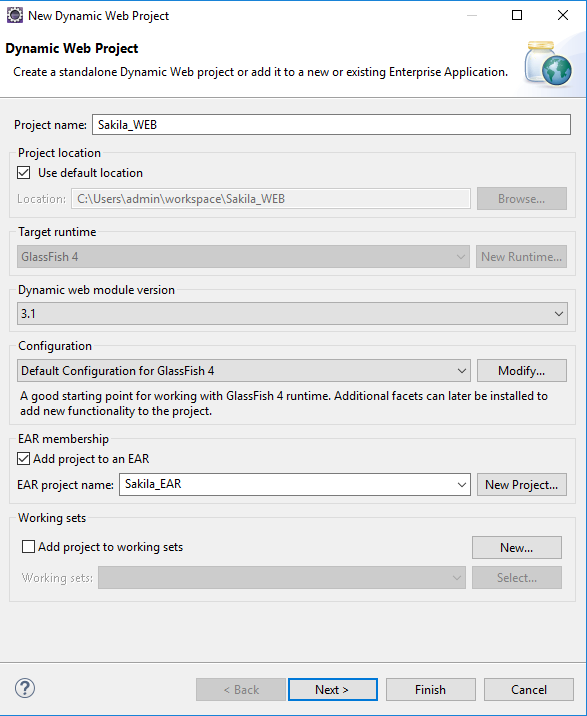


Click on the next button for two times and disable the first checkbox and then click on finish. You don’t need to create an EJB Client project because the WEB project will be your EJB Client.

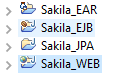


Last but definitely not least the WEB project and make sure to add it to the EAR

New-> Dynamic Web Project



Now you have 4 new projects

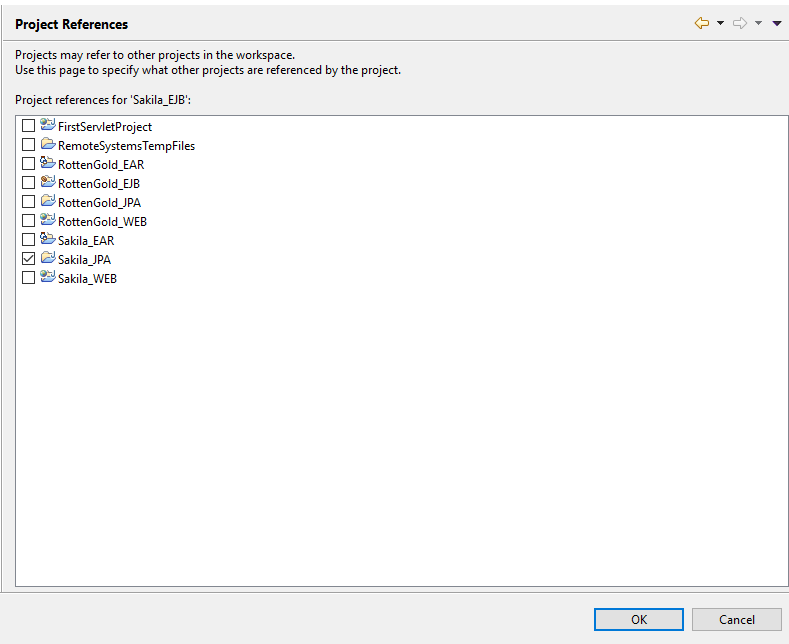


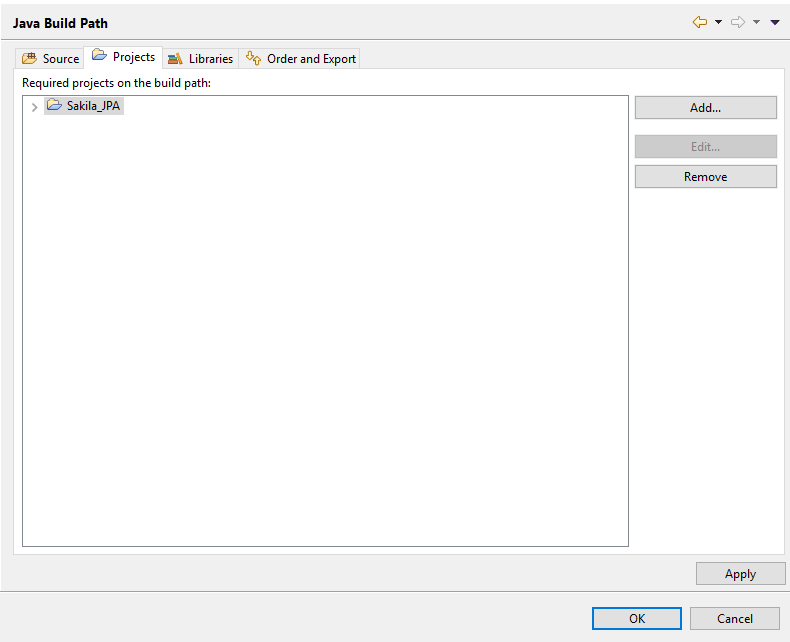
# Connecting the projects

Now you have to go to the project properties and add some project references

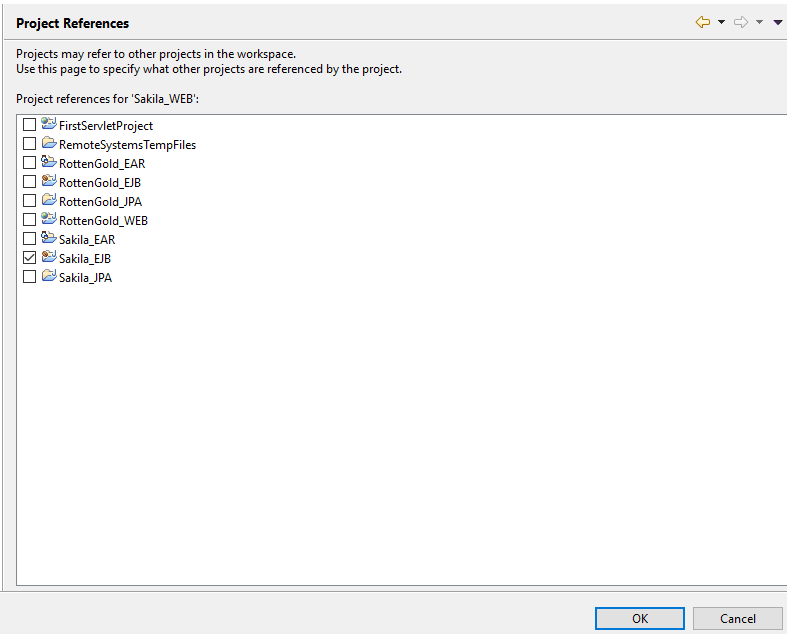
The JPA has to know NO other project

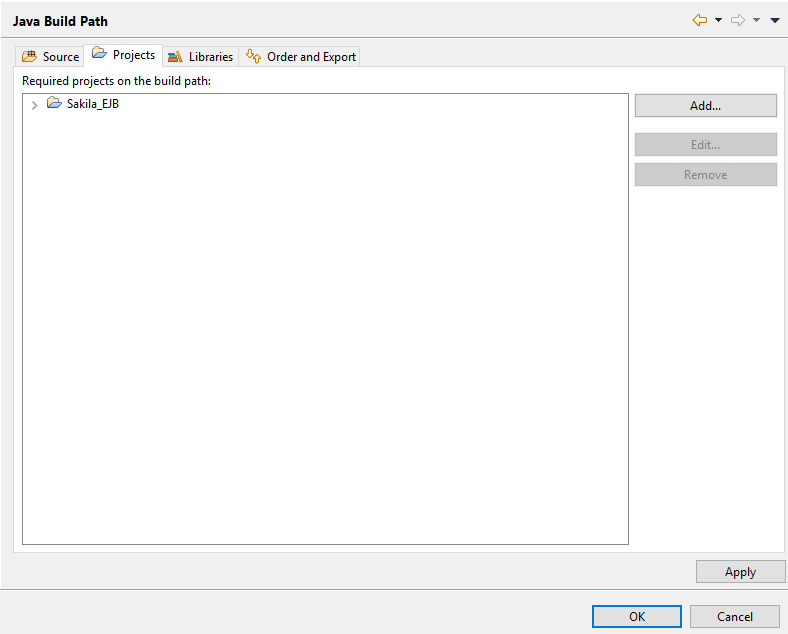
The EJB has to know the JPA project. Go to the EJB\_Sakila project properties.





The WEB has to know the EJB project. Go to the WEB\_Sakila project properties.





The EAR already knows all the project because you added them while setting up the projects

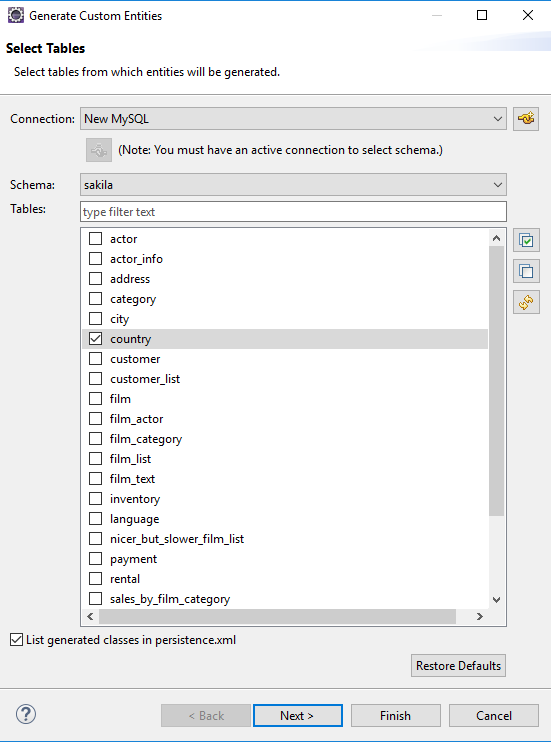
# Code

Now let’s test if everything is connected correctly

You will always keep the Code in MVC-Standards

First of all you create the model which is located in the EJB project. Therefore you have to create a new JPA Entity from the table “Country” which will be in the JPA project “src” folder.

New-> JPA Entities from Tables



The JPA should look something like this:

**package** com.rothlin.hwz.model;

**import** java.io.Serializable;

**import** javax.persistence.\*;

**import** java.sql.Timestamp;

/\*\*

\* The persistent class for the country database table.

\*

\*/

@Entity

@NamedQueries({ @NamedQuery(name = "Country.findAll", query = "SELECT c FROM Country c"),

@NamedQuery(name = "Country.findCountryByName", query = "SELECT c FROM Country c WHERE c.country LIKE :countryName") })

**public** **class** Country **implements** Serializable {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Id

@Column(name = "country\_id")

**private** **int** countryId;

**private** String country;

@Column(name = "last\_update")

**private** Timestamp lastUpdate;

**public** Country() {

}

**public** Country(**int** countryId, String country, Timestamp lastUpdate) {

**this**.countryId = countryId;

**this**.country = country;

**this**.lastUpdate = lastUpdate;

}

**public** **int** getCountryId() {

**return** **this**.countryId;

}

**public** **void** setCountryId(**int** countryId) {

**this**.countryId = countryId;

}

**public** String getCountry() {

**return** **this**.country;

}

**public** **void** setCountry(String country) {

**this**.country = country;

}

**public** Timestamp getLastUpdate() {

**return** **this**.lastUpdate;

}

**public** **void** setLastUpdate(Timestamp lastUpdate) {

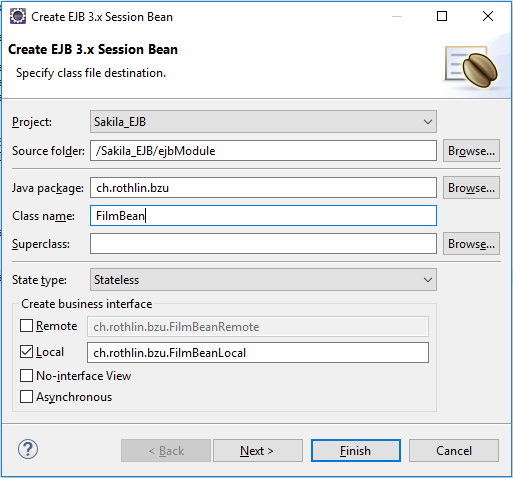
**this**.lastUpdate = lastUpdate;

}

}

Because you want to connect to the database later and also read data from it, which are all the different countries on the “sakila” database, you have to create the CountryBean.java which you have to put into the “src” folder of the EJB project. Also create a local interface

New-> Session Bean (EJB 3.x)



After some coding the interface should look something like this:

**package** com.rothlin.hwz.bean;

**import** java.util.List;

**import** javax.ejb.Local;

**import** com.rothlin.hwz.model.Country;

@Local

**public** **interface** CountryBeanLocal {

**public** List<Country> searchForCountries(String countryInput, String searchOption) **throws** Exception;

}

And the bean should look something like this:

**package** com.rothlin.hwz.bean;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.Statement;

**import** java.sql.Timestamp;

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.ejb.Stateless;

**import** javax.persistence.EntityManager;

**import** javax.persistence.PersistenceContext;

**import** com.rothlin.hwz.model.Country;

/\*\*

\* Session Bean implementation class CountryBean

\*/

@Stateless

**public** **class** CountryBean **implements** CountryBeanLocal {

@PersistenceContext

**private** EntityManager em;

//SQL statement example

@Override

**public** List<Country> searchForCountries(String searchPattern, String searchCriteria) **throws** Exception {

String sql = "SELECT \* FROM country WHERE country LIKE ";

Connection connection = DriverManager

.*getConnection*("jdbc:mysql://localhost:3306/sakila?user=root&password=admin");

Statement statement = connection.createStatement();

**if** (searchCriteria.equals("contains"))

sql = sql + "'%" + searchPattern + "%'";

**else** **if** (searchCriteria.equals("beginnsWith"))

sql = sql + "'" + searchPattern + "%'";

statement.executeQuery(sql);

ResultSet resultSet = statement.getResultSet();

List<Country> countryResults = **new** ArrayList<Country>();

**while** (resultSet.next()) {

**int** countryId = resultSet.getInt(1);

String country = resultSet.getString(2);

Timestamp lastUpdate = resultSet.getTimestamp(3);

countryResults.add(**new** Country(countryId, country, lastUpdate));

}

**return** countryResults;

}

// JPA example

// @Override

// public List<Country> searchForCountrys(String searchPattern, String

// searchCriteria) {

// if (searchCriteria.equals("contains"))

// List<Country> countryResults =

// em.createNamedQuery("Country.findCountryByName").setParameter("countryName",

// "%"+searchPattern+"%").getResultList();

// if (searchCriteria.equals("beginnsWith"))

// List<Country> countryResults =

// em.createNamedQuery("Country.findCountryByName").setParameter("countryName",

// searchPattern+"%").getResultList();

// return countryResults;

// }

}

Next you have to add the CountryController.java, which is located in the “src” folder of the WEB project.

New -> Class

The code should look something like this after some coding:   
First Part:

package com.rothlin.hwz.controller;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

import javax.ejb.EJB;

import javax.enterprise.context.ApplicationScoped;

import javax.inject.Named;

import com.rothlin.hwz.bean.CountryBeanLocal;

import com.rothlin.hwz.model.Country;

@Named

@ApplicationScoped

public class CountryController implements Serializable {

private static final long serialVersionUID = 1L;

private String searchPattern;

private String searchCriteria = "contains";

private int searchResultCount;

private boolean resultsFound;

private List<Country> countryResults = new ArrayList<Country>();

@EJB

private CountryBeanLocal countryBean;

public String searchCountries() {

try {

countryResults = countryBean.searchForCountries(searchPattern, searchCriteria);

} catch (Exception e) {

e.printStackTrace();

}

return "index.xhtml";

public List<Country> getCountryResults() {

return countryResults;

}

public void setCountryResults(List<Country> countryResults) {

this.countryResults = countryResults;

}

Second Part:

public int getSearchResultCount() {

searchResultCount = countryResults.size();

return searchResultCount;

}

public void setSearchResultCount(int searchResultCount) {

this.searchResultCount = countryResults.size();

}

public boolean isResultsFound() {

if (countryResults.isEmpty()){

resultsFound = false;

}else{

resultsFound = true;

}

return resultsFound;

}

public void setResultsFound(boolean resultsFound) {

this.resultsFound = resultsFound;

}

public String getSearchPattern() {

return searchPattern;

}

public void setSearchPattern(String searchPattern) {

this.searchPattern = searchPattern;

}

public String getSearchCriteria() {

return searchCriteria;

}

public void setSearchCriteria(String searchCriteria) {

this.searchCriteria = searchCriteria;

}

}

Last but not least you have to add the view, therefore you have to create a index.xhtml file in the “WebContent” folder of your WEB project.

New -> HTML file -> edit the file name to index.xhtml

After some coding the code should look something like this:

<h:html xmlns=*"http://www.w3.org/1999/xhtml"*

xmlns:f=*"http://xmlns.jcp.org/jsf/core"*

xmlns:h=*"http://xmlns.jcp.org/jsf/html"*

xmlns:ui=*"http://xmlns.jcp.org/jsf/facelets"*>

<h:head>

</h:head>

<h:body>

<h:form>

<h:panelGroup>

<h:outputText value=*"Country Search Input:"* />

</h:panelGroup>

<h:panelGroup>

<h:selectOneRadio value=*"#{countryController.searchOption}"*>

<f:selectItem itemValue=*"contains"* itemLabel=*"Contains"* />

<f:selectItem itemValue=*"beginnsWith"* itemLabel=*"Beginns with"* />

</h:selectOneRadio>

</h:panelGroup>

<h:panelGroup>

<h:inputText value=*"#{countryController.countryInput}"* />

<h:commandButton value=*"Search"* type=*"submit"*

action=*"#{countryController.searchCountry}"* />

</h:panelGroup>

<br/>

<h:panelGroup rendered=*"#{countryController.resultsFound}"*>

<h:outputText value=*"#{countryController.searchResultCount}"* />

<h:outputText value=*" Countries were found!"* />

</h:panelGroup>

<h:panelGroup>

<h:dataTable value=*"#{countryController.countryResults}"*

var=*"countryR"*>

<h:column>

<f:facet name=*"header"*>Country ID</f:facet>

<h:outputText value=*"#{countryR.countryId}"* />

</h:column>

<h:column>

<f:facet name=*"header"*>Country</f:facet>

<h:outputText value=*"#{countryR.country}"* />

</h:column>

<h:column>

<f:facet name=*"header"*>Last Updated</f:facet>

<h:outputText value=*"#{countryR.lastUpdate}"* />

</h:column>

</h:dataTable>

</h:panelGroup>

</h:form>

</h:body>

</h:html>