SAP JPA 1.0, EJB 3.0 and WebService -Modeling Your First JPA Entity in CE 7.1

Applies to:

SAP NetWeaver Composition Environment 7.1 EHP1. For more information, visit the Java homepage.

Summary

This article explains the necessary steps involved in creating, configuring, invoking, and deploying a SAP JPA 1.0 entity using an example.

Author:

Company: HCL Technologies, Chennai, India.

Created on: 28 May 2009

Author Bio

Sampath Gunda Is working as a Java Developer in HCL Technologies, Chennai, India.

Table of Contents

Introduction 3

Creating the Development Component Projects 3

Creating the Development Components 3

Adding JPA Facet to the EJB Project 7

Adding Data Source Alias Facets to Enterprise Application (EAR) Project 9

Creating and Adjusting a JPA Entity 10

Creating Employee Entity 10

Entering a New Data Source Alias in the Data Source XML 12

Defining the Persistence Unit in the Persistence XML 13

Generating the Entities' Tables 14

Developing the Business Logic by Implementing an EJB Stateless session bean 15

Exposing the EJB Session Bean as a Web Service 18

Prepare, deploy and run the application 21

Creating the dependency between the EAR and Dictionary development components 21

Building and deploying the application 22

Test the EmployeeService in WS Navigator 23

Related Content 26

Copyright ..... **Error! Bookmark not defined.**

Introduction

This article explains the necessary steps involved in creating, configuring, invoking, and deploying a SAP JPA 1.0 entity using an example.

The sample application that we are going to create in this article is based on a simple employee data model and uses only one entity called “Employee”. It allows you to create a new employee, find an existing one using its unique identifier, update an existing employee data, list all employees, and finally to remove an existing employee.

First, we will create an Employee JPA entity to persist the Employee data. Then we will create an EJB stateless session bean called EmployeeBean and invoke the Employee entity from it. Finally, we will expose the EmployeeBean session bean as a web service (EmployeeService) and deploy it.

This sample example will have the following components:

1. Employee : JPA Entity
2. EmployeeBean: EJB 3.0 Stateless session bean
3. EmployeeService: Web Service

This article is divided into following steps:

1. Creating the necessary development component projects
2. Creating the Employee JPA entity.
3. Implementing an EJB Stateless session bean called EmployeeBean and invoking the Employee JPA entity from it.
4. Exposing the EmployeeBean as a Web Service
5. Prepare, deploy and run the sample application

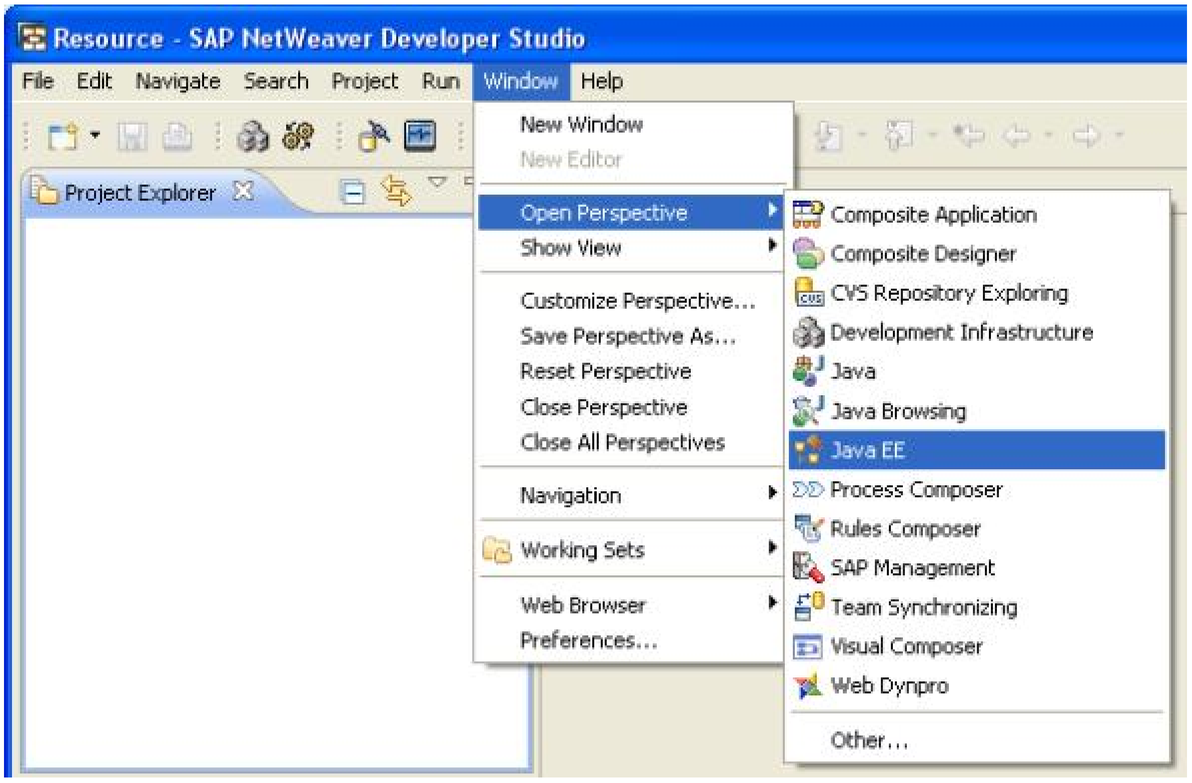
Now, let see these steps elaborately in the following sections.

Creating the Development Component Projects

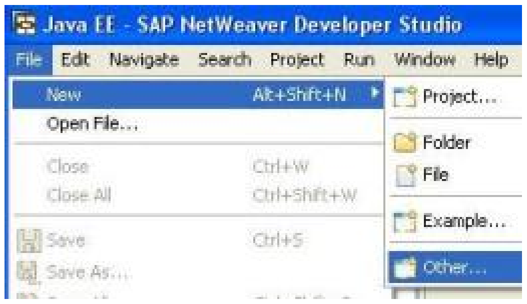
Creating the Development Components

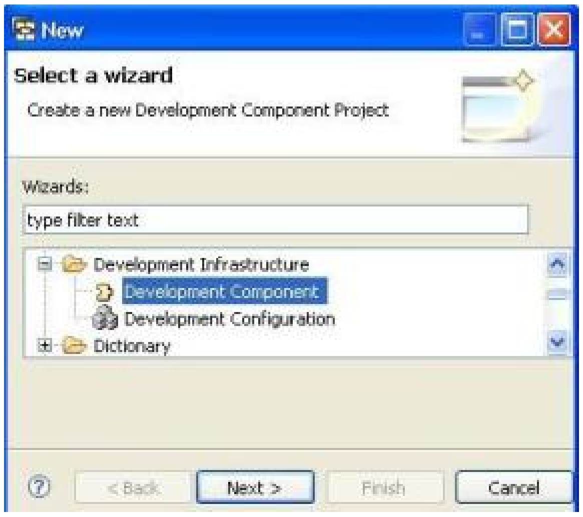
This section describes how to create the necessary development component projects for our sample application. Three development component projects are needed: EJB Module, Enterprise Application, and Dictionary. Let us create these three development components.

In your SAP NetWeaver Developer Studio, open the Java EE perspective.



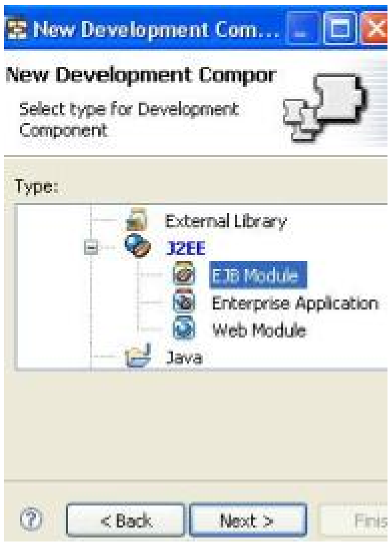
To create a new development component, choose *File →New →Other →Development infrastructure →Development Component →Next*





Specify the type and name of the development component for EJB module, Enterprise Application Project and Dictionary correspondingly as given below:

For EJB Module — choose *J2EE →EJB Module →Next →LocalDevelopment →MyComponents →Next*. Then enter the name **employee_app** and choose *Finish.*



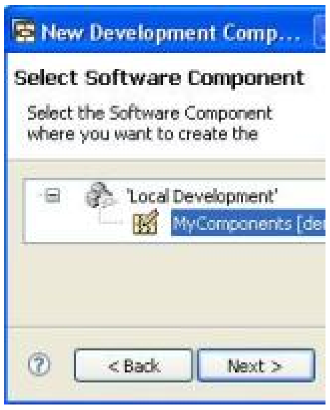
New Development Com...

New Development Compor

Select type for Development Component

Type:

- External Library
- J2EE
  - EJB Module
  - Enterprise Application
  - Web Module
- Java

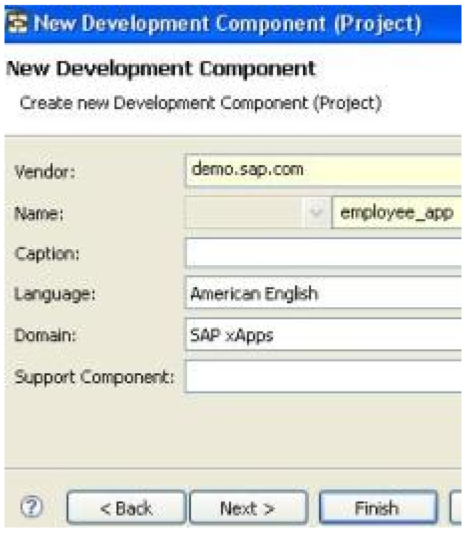


New Development Comp...

Select Software Component

Select the Software Component where you want to create the

- 'Local Development'
  - MyComponents [de



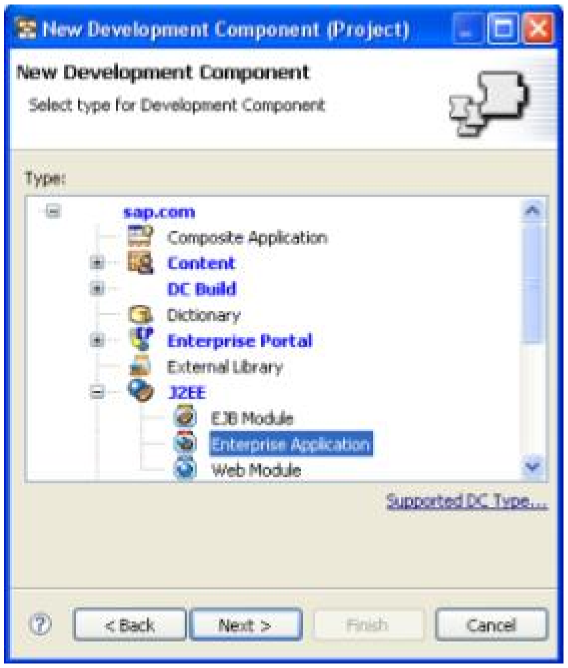
New Development Component (Project)

New Development Component

Create new Development Component (Project)

Vendor:	demo.sap.com
Name:	employee_app
Caption:	
Language:	American English
Domain:	SAP xApps
Support Component:	

For Enterprise Application — choose *J2EE →Enterprise Application →Next →LocalDevelopment →MyComponents →Next.*



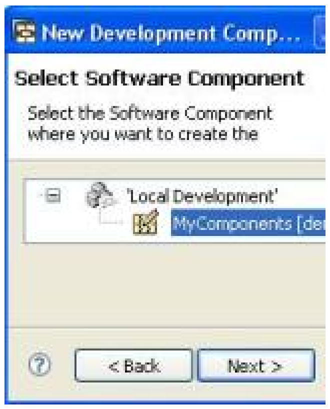
New Development Component (Project)

New Development Component

Select type for Development Component

Type:

- sap.com
  - Composite Application
  - Content
  - DC Build
  - Dictionary
  - Enterprise Portal
  - External Library
  - J2EE
    - EJB Module
    - Enterprise Application
    - Web Module



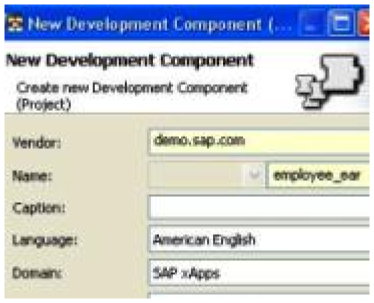
New Development Comp...

Select Software Component

Select the Software Component where you want to create the

- 'Local Development'
  - MyComponents [de

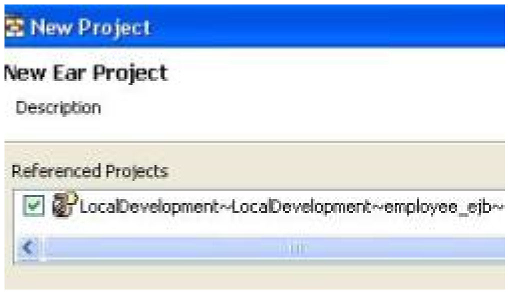
Then enter the **name employee_ear** and choose *Next,* and then select the checkbox indicator for making it reference the above created EJB development component project (employee_ejb) and choose *Finish.*


New Development Component (...

New Development Component

Create new Development Component (Project)

Vendor:	demo.sap.com
Name:	employee_ear
Caption:	
Language:	American English
Domain:	SAP xApps


New Project

New Ear Project

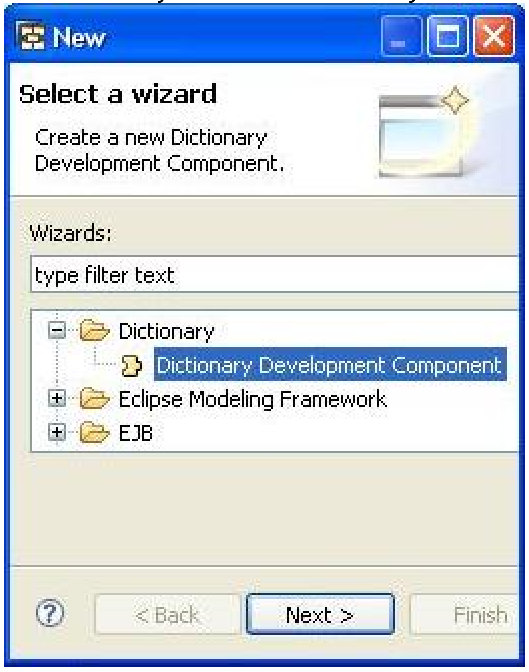
Description

Referenced Projects

LocalDevelopment~LocalDevelopment~employee_ejb~

Now your Enterprise Application development component has valid references to EJB development component project.

For Dictionary — choose *Dictionary →Next →LocalDevelopment →MyComponents →Next*.


New

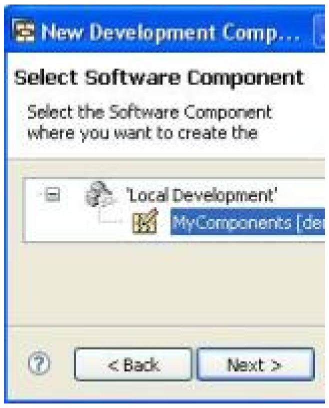
Select a wizard

Create a new Dictionary Development Component.

Wizards:

type filter text

- Dictionary
  - Dictionary Development Component
- Eclipse Modeling Framework
- EJB

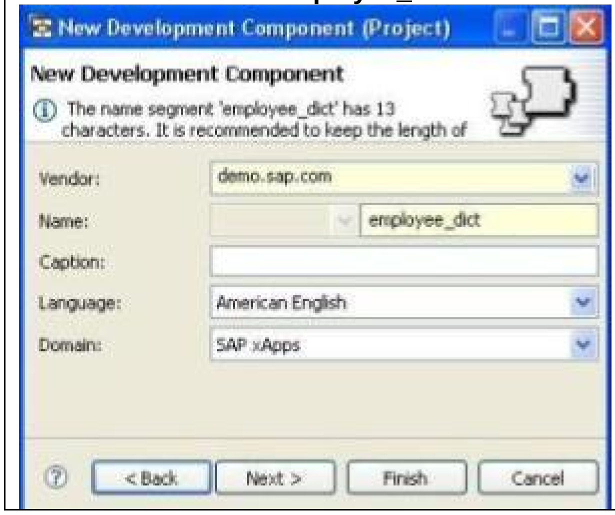

New Development Comp...

Select Software Component

Select the Software Component where you want to create the

- 'Local Development'
  - MyComponents [de

Then enter the name **employee_dict** and choose *Next* then *Finish*.


New Development Component (Project)

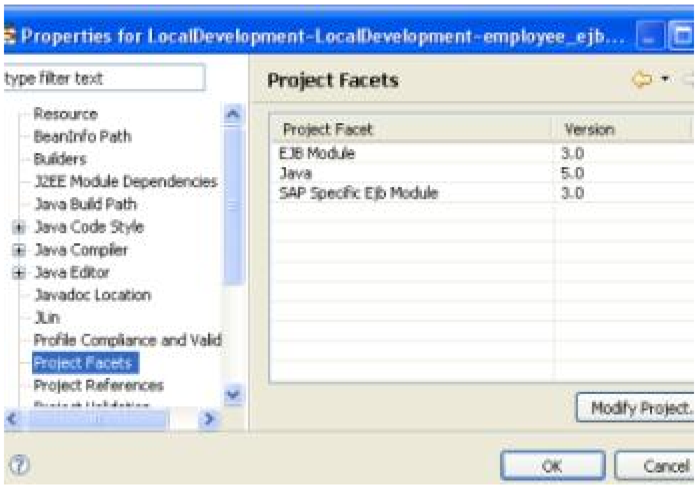
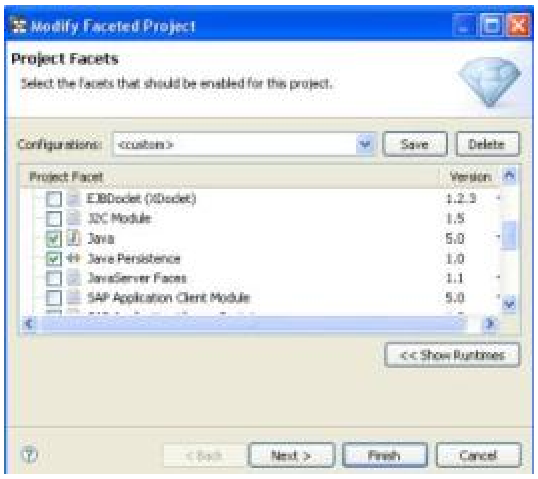
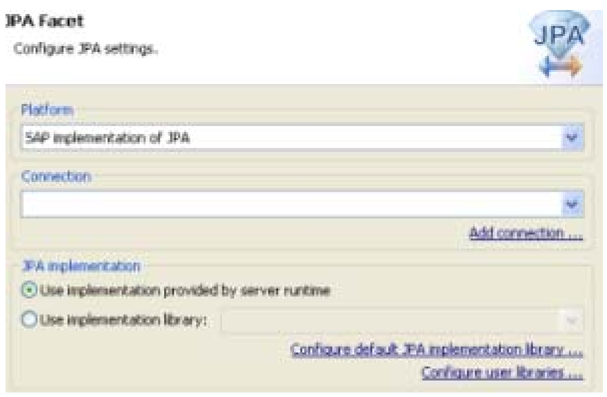
New Development Component

The name segment 'employee_dict' has 13 characters. It is recommended to keep the length of

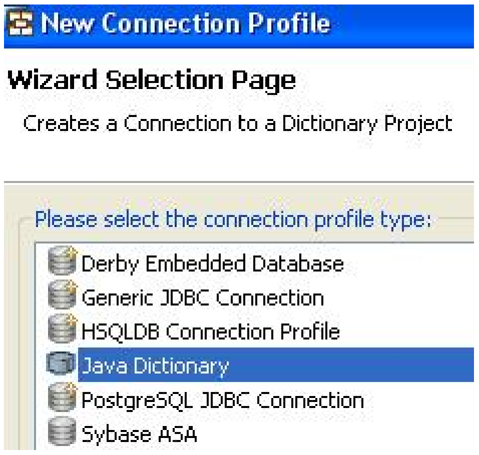
Vendor:	demo.sap.com
Name:	employee_dict
Caption:	
Language:	American English
Domain:	SAP xApps

Adding JPA Facet to the EJB Project

In this section, we will add JPA Facet to the employee_ejb project and create a relationship (connection) between employee_ejb project, containing the JPA entities, and employee_dict project which is utilizing the forward mapping functionality of the toolset. This way you generate the database tables straight from the JPA entities.

From the context menu of the employee_ejb (EJB project) choose *Properties →Project Facets → Modify Project.*	 Properties for LocalDevelopment-LocalDevelopment-employee_ejb... type filter text Resource BeanInfo Path Builders J2EE Module Dependencies Java Build Path Java Code Style Java Compiler Java Editor Javadoc Location JUnit Profile Compliance and Valid Project Facets Project References Project Facets Project Facet — Version EJB Module — 3.0 Java — 5.0 SAP Specific Ejb Module — 3.0 Modify Project... OK Cancel
Select the *Java Persistence* checkbox indicator and choose *Next*.	 Modify Faceted Project Project Facets Select the facets that should be enabled for this project. Configurations: <custom> Save Delete Project Facet — Version ☐ EJBDoclet (XDoclet) — 1.2.3 ☐ J2C Module — 1.5 ☑ Java — 5.0 ☑ Java Persistence — 1.0 ☐ JavaServer Faces — 1.1 ☐ SAP Application Client Module — 5.0 << Show Runtimes < Back Next > Finish Cancel
From the Platform dropdown menu select *SAP implementation of JPA* and choose *Add connection*.	 JPA Facet Configure JPA settings. Platform SAP implementation of JPA Connection Add connection ... JPA implementation ● Use implementation provided by server runtime ○ Use implementation library: Configure default JPA implementation library ... Configure user libraries ...

In the New Connection Profile window, choose *Java Dictionary* and then *Next*. Enter the connection name **employee_conn** and select the *Auto-connect at startup* indicator .Finally choose *Next*.

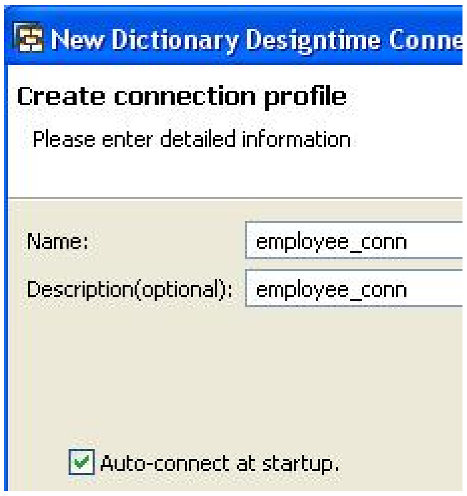


Wizard Selection Page

Creates a Connection to a Dictionary Project

Please select the connection profile type:

- Derby Embedded Database
- Generic JDBC Connection
- HSQLDB Connection Profile
- Java Dictionary
- PostgreSQL JDBC Connection
- Sybase ASA



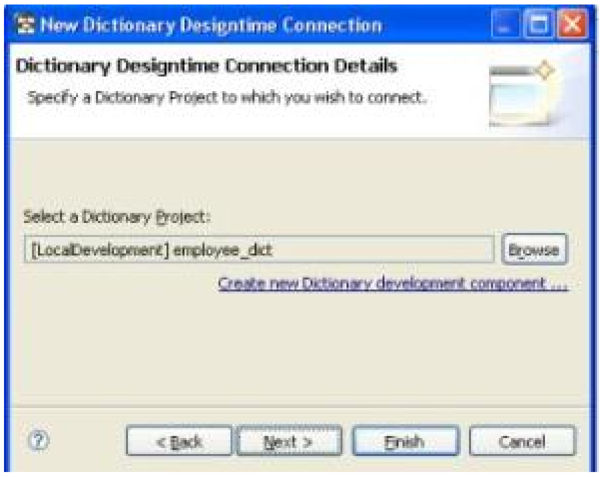
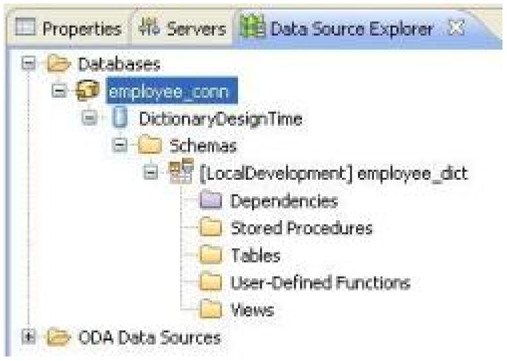
Create connection profile

Please enter detailed information

Name: employee_conn

Description(optional): employee_conn

☑ Auto-connect at startup.

Browse to the dictionary project (employee_dict, the project which we created earlier) to which the connection is made. Choose *Finish* and then *OK*.	 New Dictionary Designtime Connection Dictionary Designtime Connection Details Specify a Dictionary Project to which you wish to connect. Select a Dictionary Project: [LocalDevelopment] employee_dict Browse Create new Dictionary development component ... < Back · Next > · Finish · Cancel
Now, You can see the employee_conn connection to employee_dict project through the *Data Source Explorer* view. To open the Data Source Explorer view, Choose *Window→ Show View →Other→ Connectivity→ Data Source Explorer*	 Properties · Servers · Data Source Explorer Databases employee_conn DictionaryDesignTime Schemas [LocalDevelopment] employee_dict Dependencies Stored Procedures Tables User-Defined Functions Views ODA Data Sources

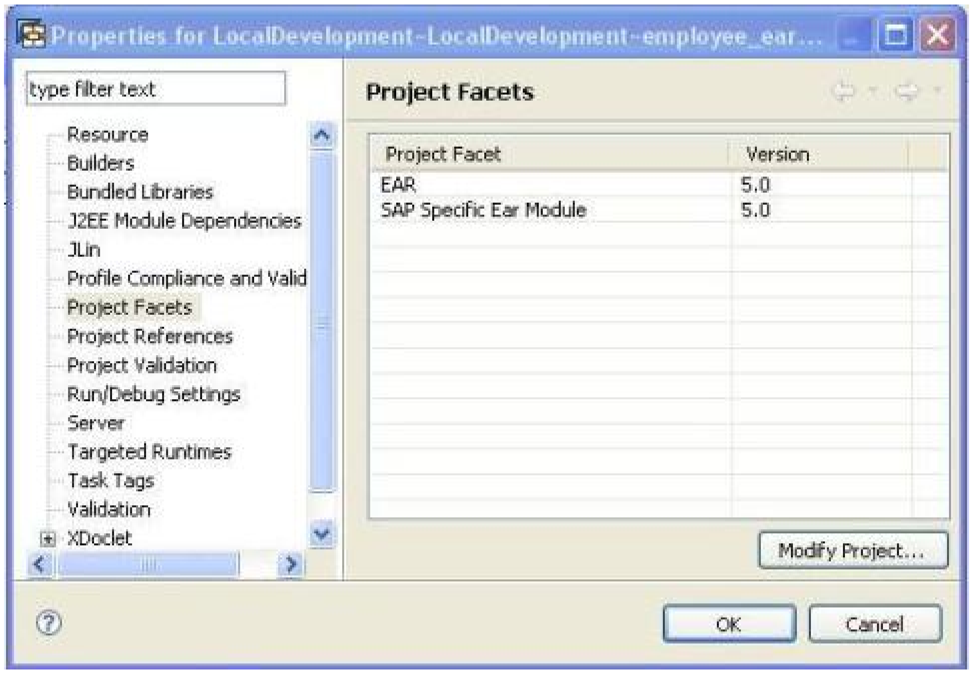
Adding Data Source Alias Facets to Enterprise Application (EAR) Project

It is common practice not to refer to the physical name of a data source directly, but to work with a so-called data source alias, i.e. a logical name for a data source, that is declared once within an application and then used (inside the application) wherever needed.

A Data Source alias is defined by a data-source-aliases.xml file. To deploy a data source alias with a JEE application, you need to add the *SAP Data Source Aliases Provider Module* facet to your EAR project (employee_ear).

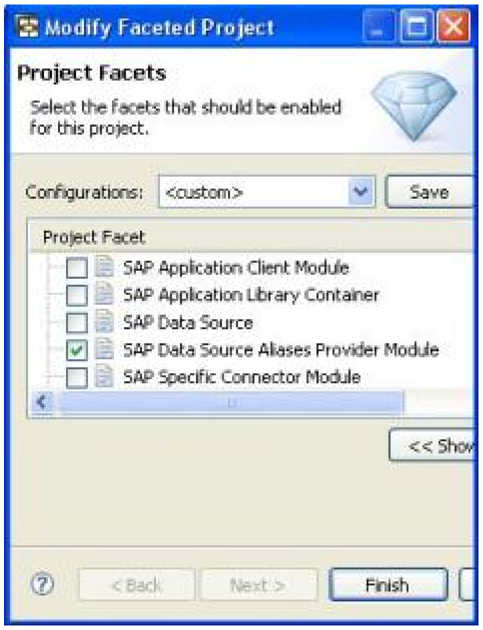
Once you add the *SAP Data Source Aliases Provider Module* facet to EAR project (employee_ear) project, you can find the data-source-aliases.xml file under EAR project / META-INF.

From the context menu of the employee_ear (EAR project) choose *Properties →Project Facets →Modify Facets →Modify Project.*

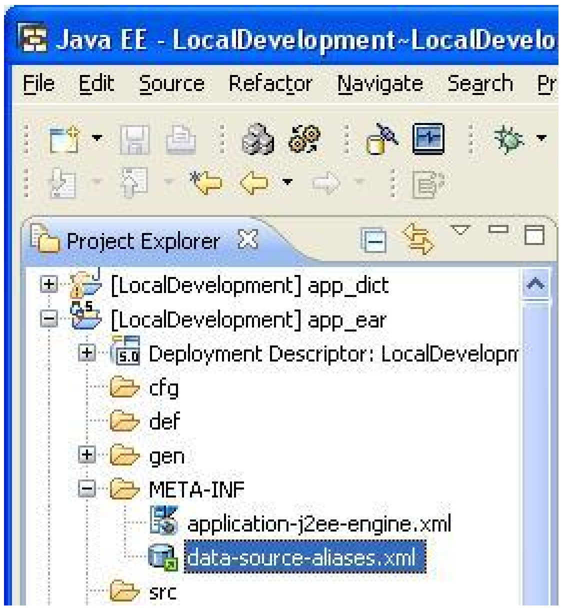


Project Facet	Version
EAR	5.0
SAP Specific Ear Module	5.0

Select the *SAP Data Source Aliases Provider Module* checkbox indicator. Choose *Finish* and then *OK.*



- [ ] SAP Application Client Module
- [ ] SAP Application Library Container
- [ ] SAP Data Source
- [✓] SAP Data Source Aliases Provider Module
- [ ] SAP Specific Connector Module

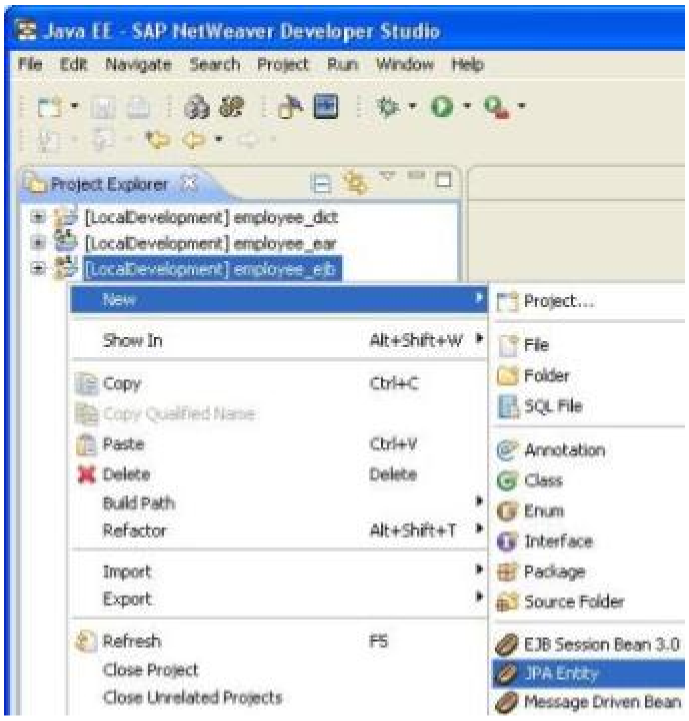

Java EE - LocalDevelopment~LocalDevelo
File Edit Source Refactor Navigate Search Pr
Project Explorer
[LocalDevelopment] app_dict
[LocalDevelopment] app_ear
Deployment Descriptor: LocalDevelopm
cfg
def
gen
META-INF
application-j2ee-engine.xml
data-source-aliases.xml
src

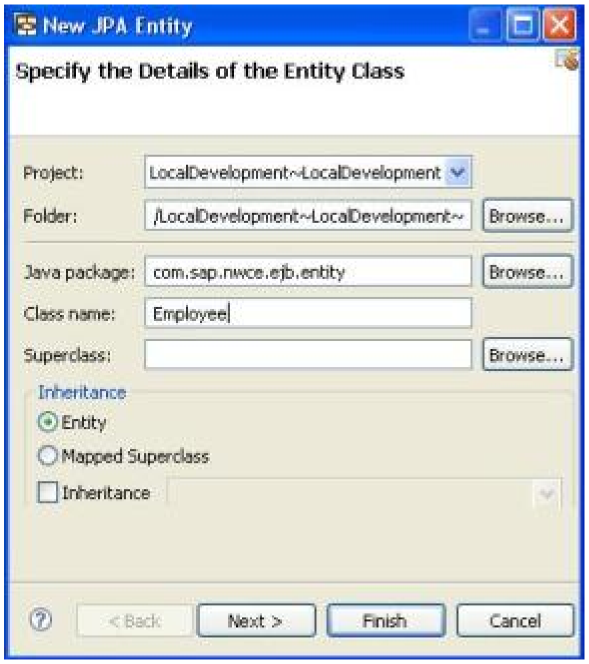
Creating and Adjusting a JPA Entity

This section describes how to create and adjust the Employee entity, how to create a new data source alias, and how to generate the entities' database tables.

Creating Employee Entity

From the context menu of employee_ejb project, choose *New* →*JPA Entity*, enter class name **Employee** and Package name **com.sap.nwce.ejb.entity**.Then choose *Next*.




New JPA Entity

Specify the Details of the Entity Class

Project: LocalDevelopment~LocalDevelopment

Folder: /LocalDevelopment~LocalDevelopment~

Java package: com.sap.nwce.ejb.entity

Class name: Employee

Superclass:

Inheritance

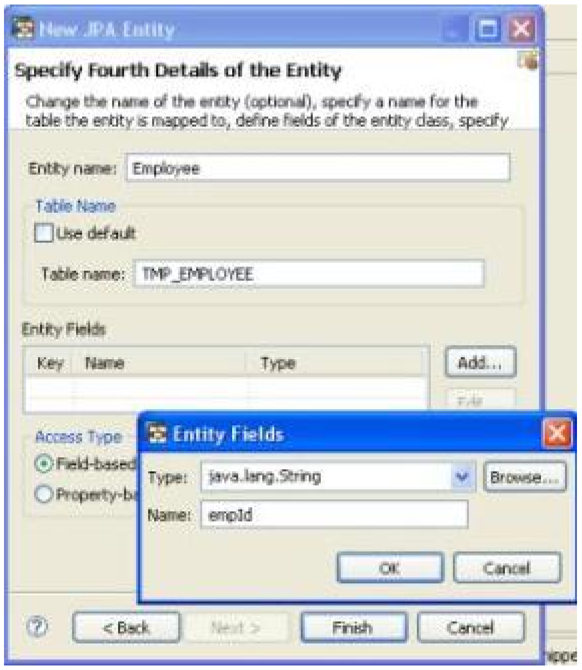
(●) Entity

( ) Mapped Superclass

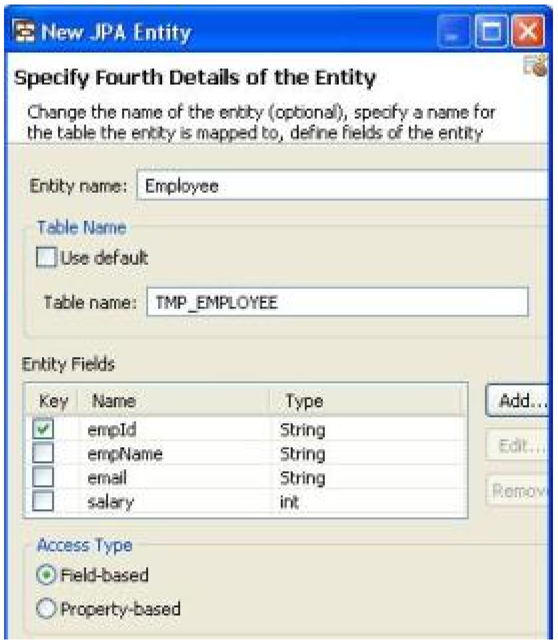
[ ] Inheritance

If you want, you can change the default Table name. Uncheck the “Use default” checkbox and enter **TMP_EMPLOYEE** as table name. It will map the employee entity onto a table named TMP_EMPLOYEE.

Add the entity fields (i.e. the entity database columns) **empId, empName, email and salary** by repeatedly choosing *Add.* Choose empId as the key of Employee entity. Choose *Finish* to complete the entity creation.


New JPA Entity
Specify Fourth Details of the Entity
Change the name of the entity (optional), specify a name for the table the entity is mapped to, define fields of the entity class, specify
Table Name
Entity Fields
Key
Name
Type
Add...
Edit
Access Type
Field-based
Property-b
Entity Fields
Browse...
OK
Cancel
< Back
Next >
Finish
Cancel

Entity name: Employee
[ ] Use default
Table name: TMP_EMPLOYEE
Type: java.lang.String
Name: empId


New JPA Entity
Specify Fourth Details of the Entity
Change the name of the entity (optional), specify a name for the table the entity is mapped to, define fields of the entity
Table Name
Entity Fields
Add...
Edit...
Remov
Access Type

Entity name: Employee
[ ] Use default
Table name: TMP_EMPLOYEE

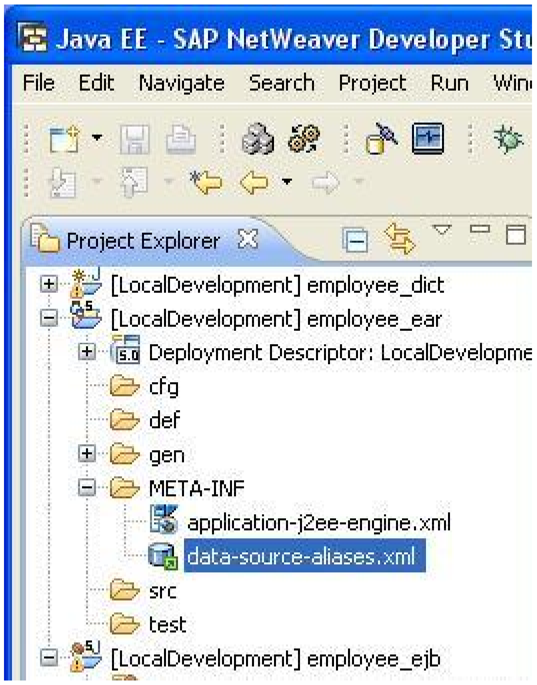
Key	Name	Type
✓	empId	String
	empName	String
	email	String
	salary	int

(● Field-based) ○ Property-based

Open the Employee class, and from the secondary mouse button *select Source →Organize Imports.* Save the changes.

You will get an error saying “table TMP_EMPLOYEE doesn’t exist”. Ignore this error. Once you create the table TMP_EMPLOYEE, it will get removed.

Entering a New Data Source Alias in the Data Source XML

To configure the data source, open the data-source-aliases.xml under EAR project / META-INF.	

Java EE - SAP NetWeaver Developer Stu

File Edit Navigate Search Project Run Win

Project Explorer

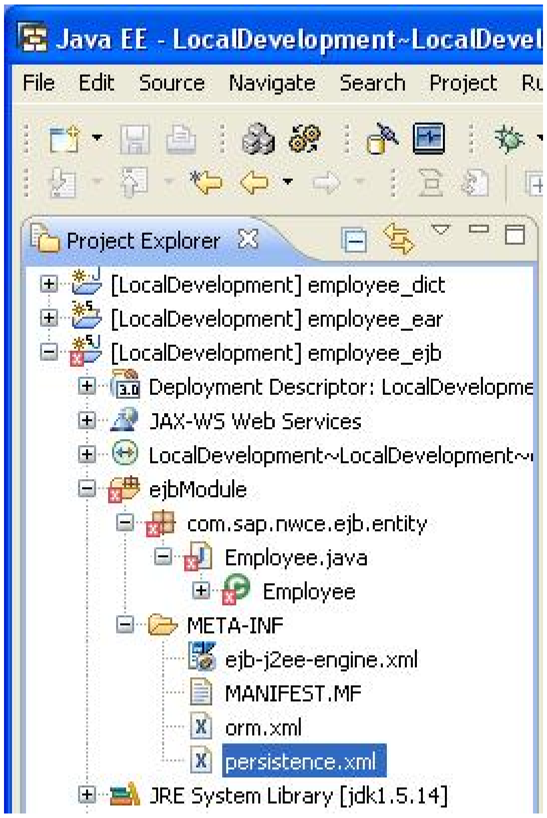
```
[LocalDevelopment] employee_dict
[LocalDevelopment] employee_ear
    Deployment Descriptor: LocalDevelopme
    cfg
    def
    gen
    META-INF
        application-j2ee-engine.xml
        data-source-aliases.xml
    src
    test
[LocalDevelopment] employee_ejb
```

Leave the default data source name as it is, but enter **EMPLOYEE_ALIAS** for the alias name. We have to use this alias name while defining the Persistence Unit in the Persistence XML file.

```
<?xml version="1.0" encoding="UTF-8"?>

<data-source-aliases xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="data-source-aliases.xsd">

  <application-
name>LocalDevelopment~LocalDevelopment~employee_ear~demo.sap.com</application-
name>

  <aliases>

    <data-source-name>${com.sap.datasource.default}</data-source-name>

    <alias>EMPLOYEE_ALIAS</alias>

  </aliases>

</data-source-aliases>
```

Defining the Persistence Unit in the Persistence XML

Entity classes are packaged and deployed as persistence units. A Persistence unite is a logical grouping of Entity classes, mapping metadata and database related configuration data. Within a JPA application, persistence unit is defined in a configuration file called persistence.xml.

To configure the persistence unit and its name, open the *persistence.xml* under *employee_ejb/ejbModule/META-INF.*	 Java EE - LocalDevelopment~LocalDevel File Edit Source Navigate Search Project Ru Project Explorer [LocalDevelopment] employee_dict [LocalDevelopment] employee_ear [LocalDevelopment] employee_ejb Deployment Descriptor: LocalDevelopme JAX-WS Web Services LocalDevelopment~LocalDevelopment~ ejbModule com.sap.nwce.ejb.entity Employee.java Employee META-INF ejb-j2ee-engine.xml MANIFEST.MF orm.xml persistence.xml JRE System Library [jdk1.5.14]

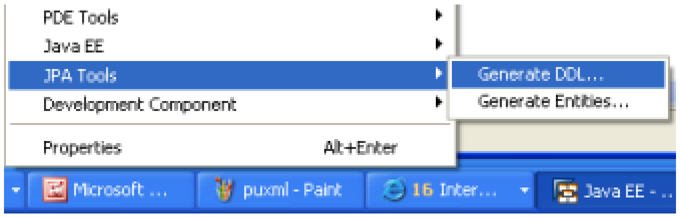
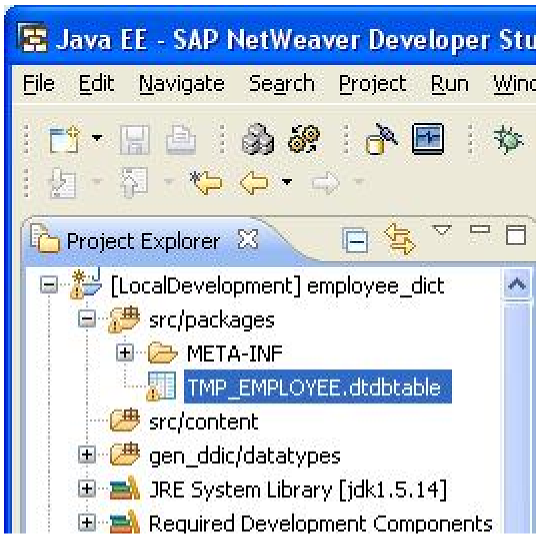
Select the *Source tab* page and enter the following source code:

Persistence Unit Name: **EMPLOYEE_PU**

Data Source: **EMPLOYEE_ALIAS** (The data source alias which we have created in the above section)

```
<?xml version="1.0" encoding="UTF-8"?>

<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">

      <persistence-unit name="EMPLOYEE_PU">

      <jta-data-source>EMPLOYEE_ALIAS</jta-data-source>

      </persistence-unit>

</persistence>
```

Generating the Entities' Tables

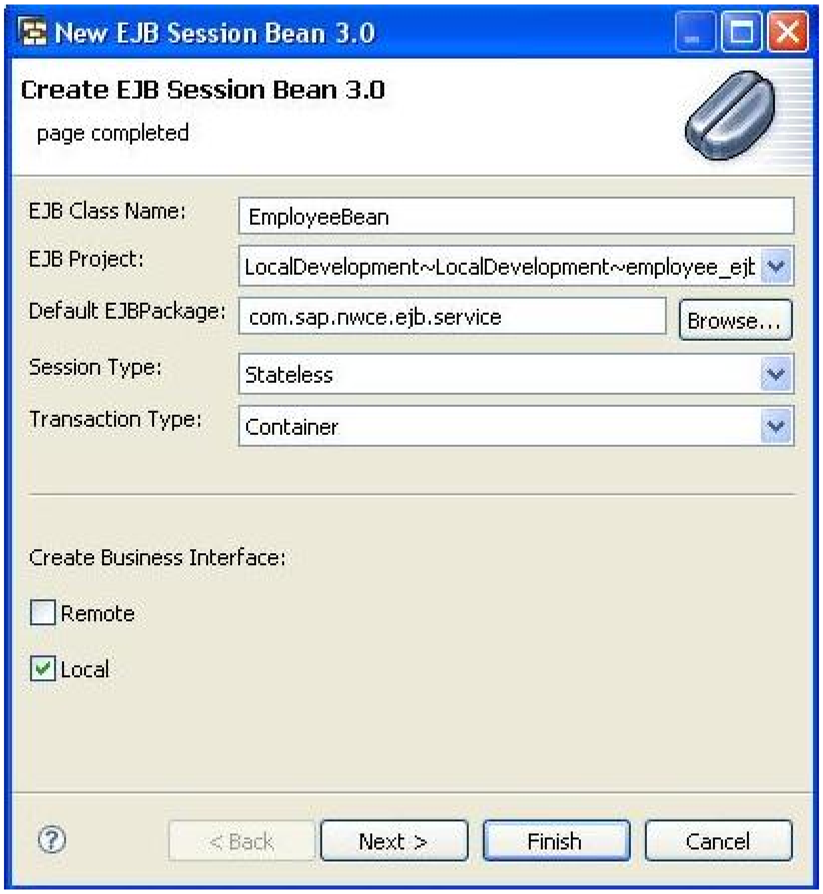
Open the context menu of the *employee_ejb* project. Select *JPA Tools* →*Generate DDL.*	
Now you can see the generated table under the *employee_dict.*	

Java EE - SAP NetWeaver Developer Stu
File Edit Navigate Search Project Run Winc
Project Explorer
[LocalDevelopment] employee_dict
src/packages
META-INF
TMP_EMPLOYEE.dtdbtable
src/content
gen_ddic/datatypes
JRE System Library [jdk1.5.14]
Required Development Components

Developing the Business Logic by Implementing an EJB Stateless session bean

The business logic of our sample application is defined in a stateless session bean named EmployeeBean. We will create a separate Java package (com.sap.nwce.ejb.service) that stores the session bean and then manually add the code to implement the CRUD and readAllEmployees operations for the Employee entity.

From the context menu of the *employee_ejb*, choose *New →EJB Session Bean 3.0*. Enter the package name **com.sap.nwce.ejb.service**. Enter **EmployeeBean** for the EJB class name and choose *Finish.*


New EJB Session Bean 3.0

Create EJB Session Bean 3.0

page completed

EJB Class Name: EmployeeBean

EJB Project: LocalDevelopment~LocalDevelopment~employee_ejt

Default EJBPackage: com.sap.nwce.ejb.service

Session Type: Stateless

Transaction Type: Container

Create Business Interface:

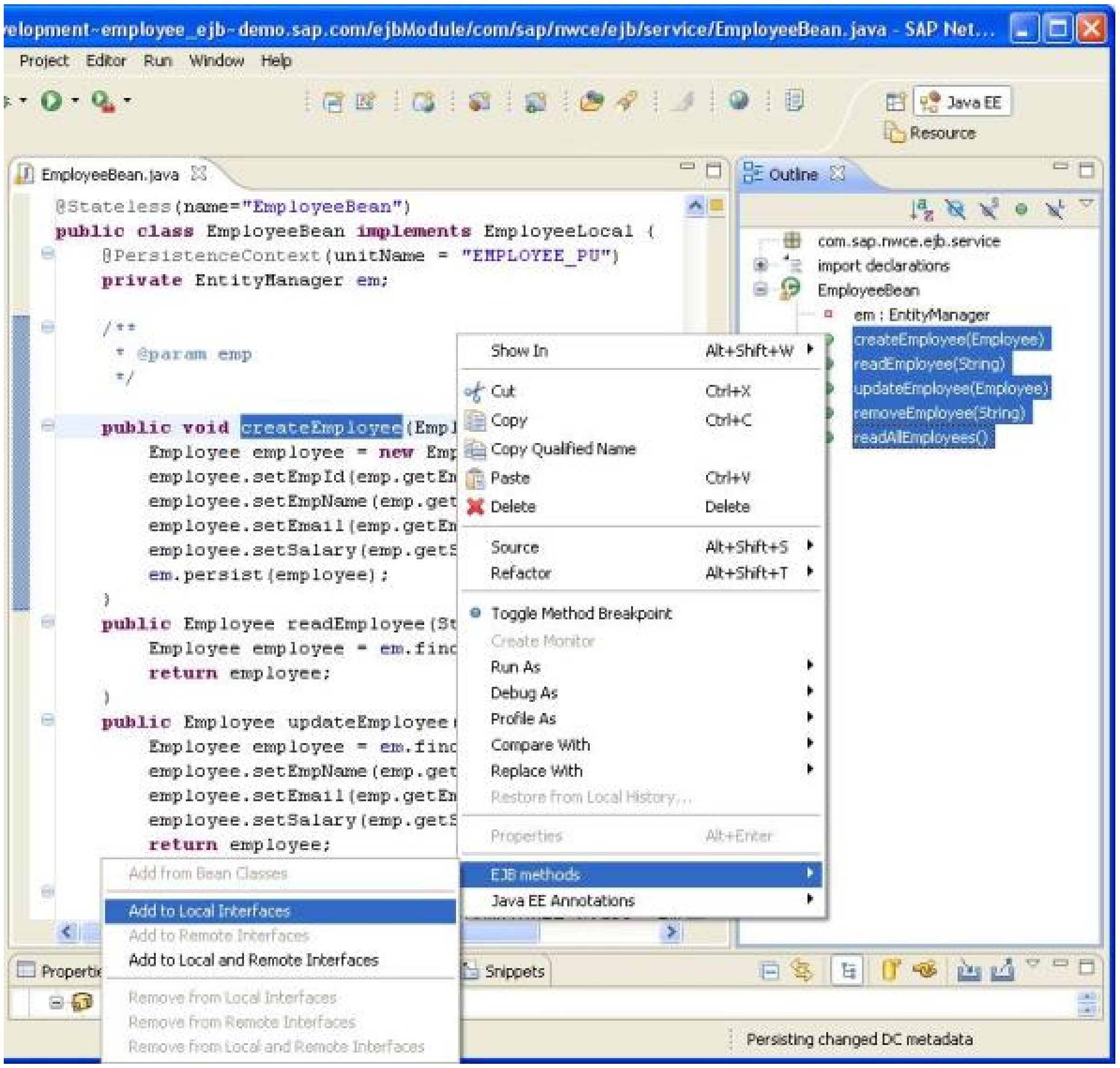
- [ ] Remote
- [✓] Local

Open the EmployeeBean class and add the following code:

```
package com.sap.nwce.ejb.service;
import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import javax.persistence.Query;
import com.sap.nwce.ejb.entity.Employee;
@Stateless(name="EmployeeBean")
public class EmployeeBean implements EmployeeLocal {
   @PersistenceContext(unitName = "EMPLOYEE_PU")
   private EntityManager em;

   public void createEmployee(Employee emp) {
          Employee employee = new Employee();
          employee.setEmpId(emp.getEmpId());
          employee.setEmpName(emp.getEmpName());
          employee.setEmail(emp.getEmail());
          employee.setSalary(emp.getSalary());
          em.persist(employee);
   }
   public Employee readEmployee(String empId) {
          Employee employee = em.find(Employee.class, empId);
          return employee;
   }
   public Employee updateEmployee(Employee emp) {
          Employee employee = em.find(Employee.class, emp.getEmpId());
          employee.setEmpName(emp.getEmpName ());
          employee.setEmail(emp.getEmail());
          employee.setSalary(emp.getSalary());
          return employee;
   }
   public void removeEmployee(String empId) {
          Employee employee = em.find(Employee.class, empId);
          if(employee!=null)
          em.remove(employee);
   }
   public List<Employee> readAllEmployees() {
          Query query = em.createQuery("Select e from Employee e");
          return  query.getResultList();
          }
}
```

Via the secondary mouse button, select *Source →Organize Imports →javax.persistence.Query →java.util.List*

Add all the methods to local interface by choosing the corresponding method from *the Outline view* and then from its context menu *choose EJB methods—> Add to local Interfaces.*

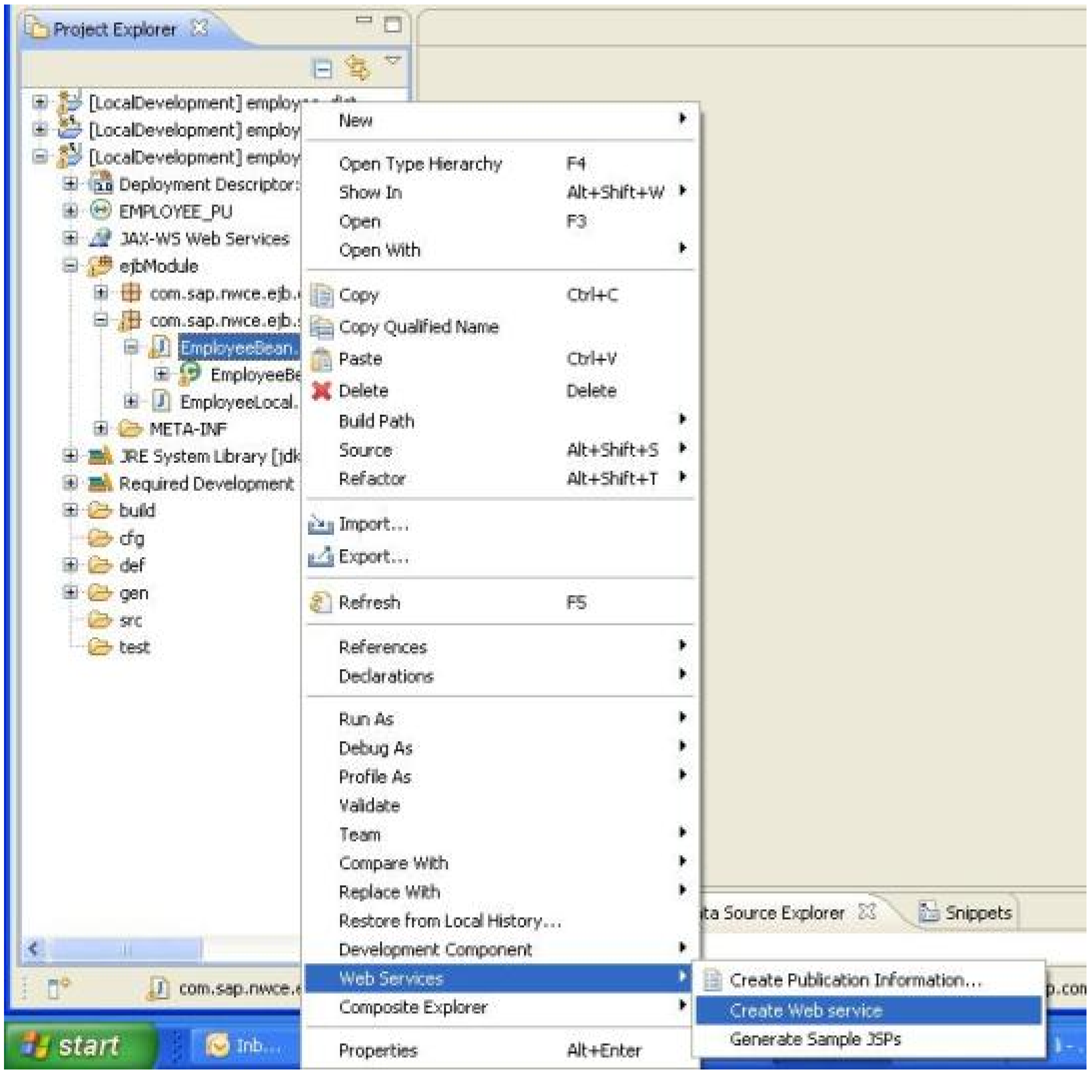


```
@Stateless(name="EmployeeBean")
public class EmployeeBean implements EmployeeLocal {
    @PersistenceContext(unitName = "EMPLOYEE_PU")
    private EntityManager em;

    /**
     * @param emp
     */

    public void createEmployee(Empl
        Employee employee = new Emp
        employee.setEmpId(emp.getEm
        employee.setEmpName(emp.get
        employee.setEmail(emp.getEm
        employee.setSalary(emp.getS
        em.persist(employee);
    }
    public Employee readEmployee(St
        Employee employee = em.finc
        return employee;
    }
    public Employee updateEmployee(
        Employee employee = em.finc
        employee.setEmpName(emp.get
        employee.setEmail(emp.getEm
        employee.setSalary(emp.getS
        return employee;
```

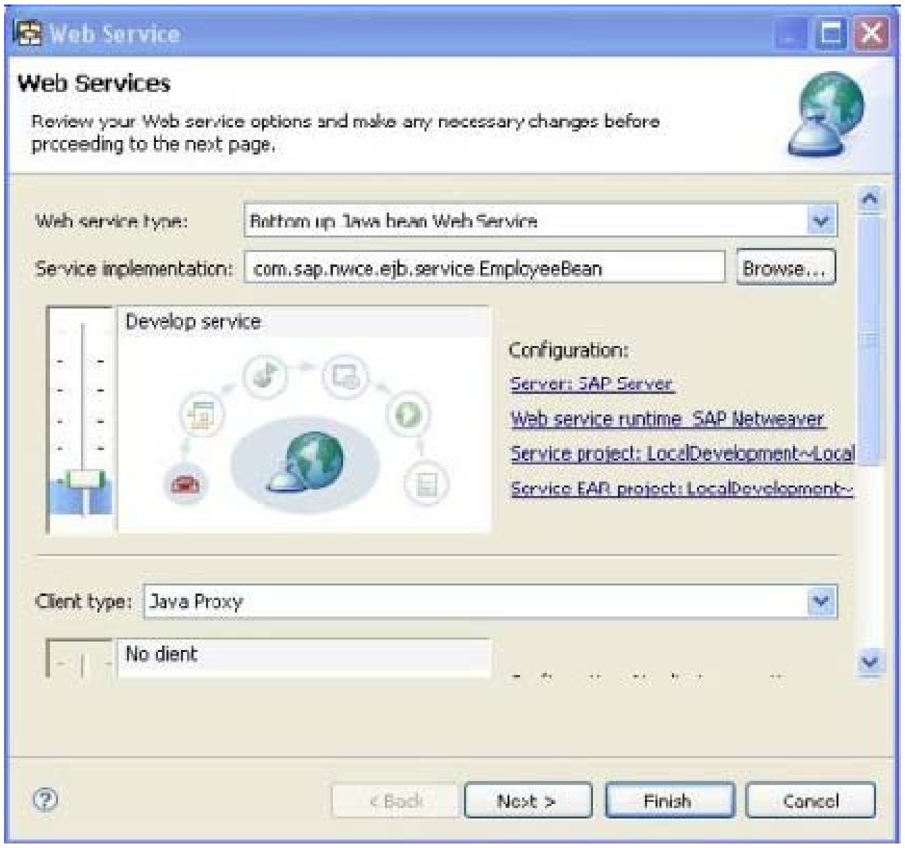
Exposing the EJB Session Bean as a Web Service

This section describes how to expose the EmployeeBean EJB session bean as a web service.

From the context menu of EmployeeBean EJB, choose *Web Services* → *Create Web Service*. The *Web Services* wizard opens.



Move the slider to the *Develop service* position. From *Configuration*, select the *SAP server* and *SAP NetWeaver* Web service runtime. Make sure that the service project (employee_ejb) and service EAR project (employee_ear) are set correctly.



Web Services

Review your Web service options and make any necessary changes before proceeding to the next page.

Web service type: Bottom up Java bean Web Service

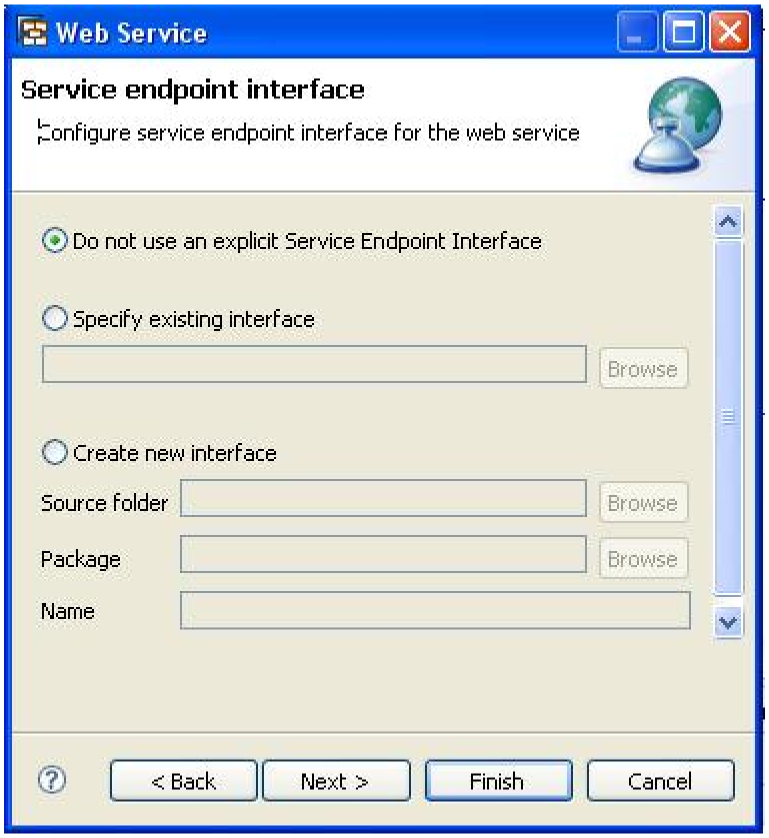
Service implementation: com.sap.nwce.ejb.service.EmployeeBean Browse...

Configuration:
Server: SAP Server
Web service runtime SAP Netweaver
Service project: LocalDevelopment~Local
Service EAR project: LocalDevelopment~

Client type: Java Proxy

No client

Choose *Next*. Depending on whether you want to use an SEI, you can choose any one of the three options as shown in following screen shot. For our application, we will use the first option.



Service endpoint interface

Configure service endpoint interface for the web service

[✓ Do not use an explicit Service Endpoint Interface]

Specify existing interface

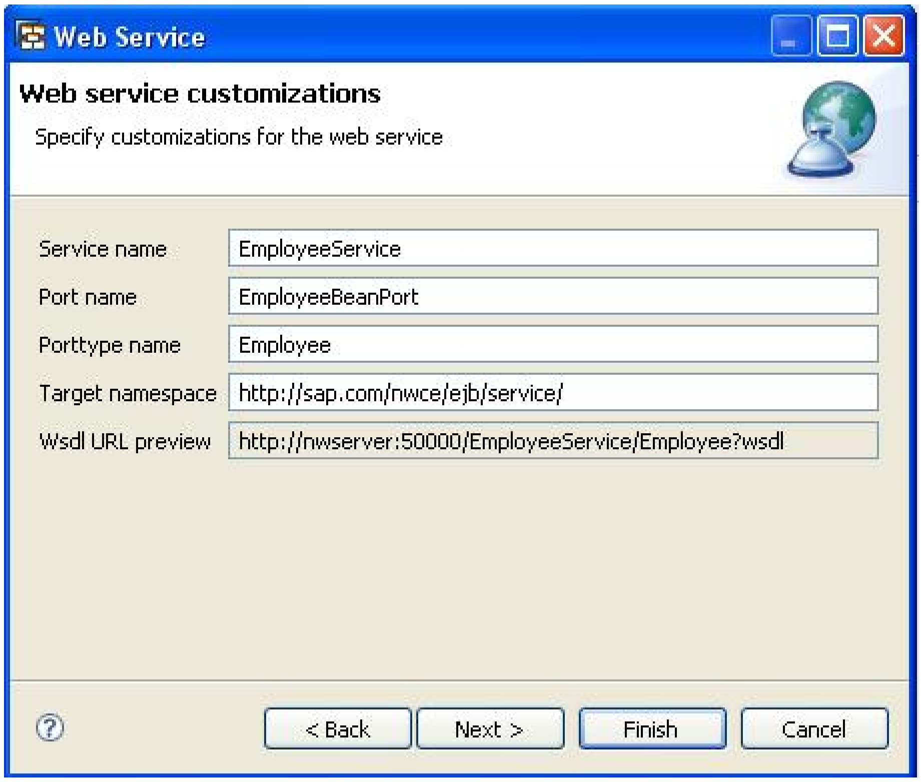
Create new interface

Source folder

Package

Name

Choose *Next.* Leave the default values and choose *Finish.* You can note down the web service URL for future reference. The Developer Studio adds Java EE 5 annotations for Web services to the implementation bean, as well as to the service endpoint interface, if applicable.


Web Service
Web service customizations
Specify customizations for the web service
< Back
Next >
Finish
Cancel

Service name	EmployeeService
Port name	EmployeeBeanPort
Porttype name	Employee
Target namespace	http://sap.com/nwce/ejb/service/
Wsdl URL preview	http://nwserver:50000/EmployeeService/Employee?wsdl

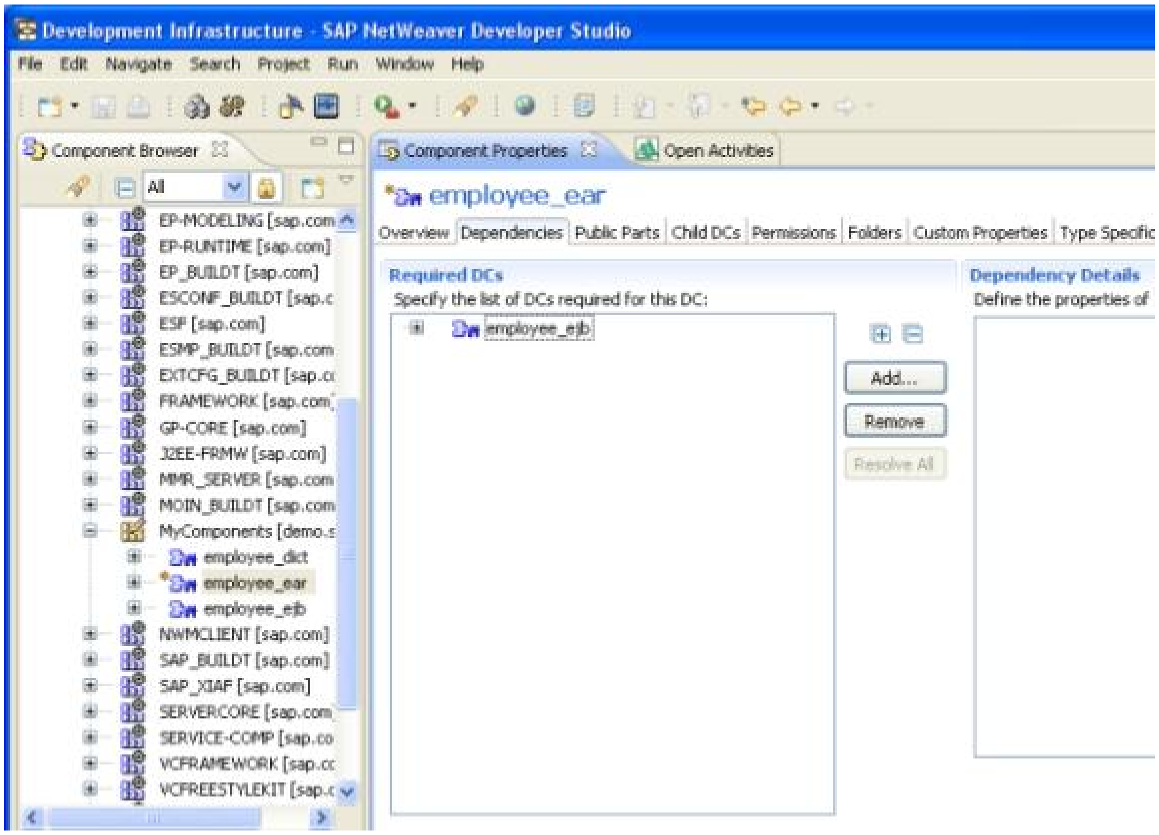
Prepare, deploy and run the application

Creating the dependency between the EAR and Dictionary development components

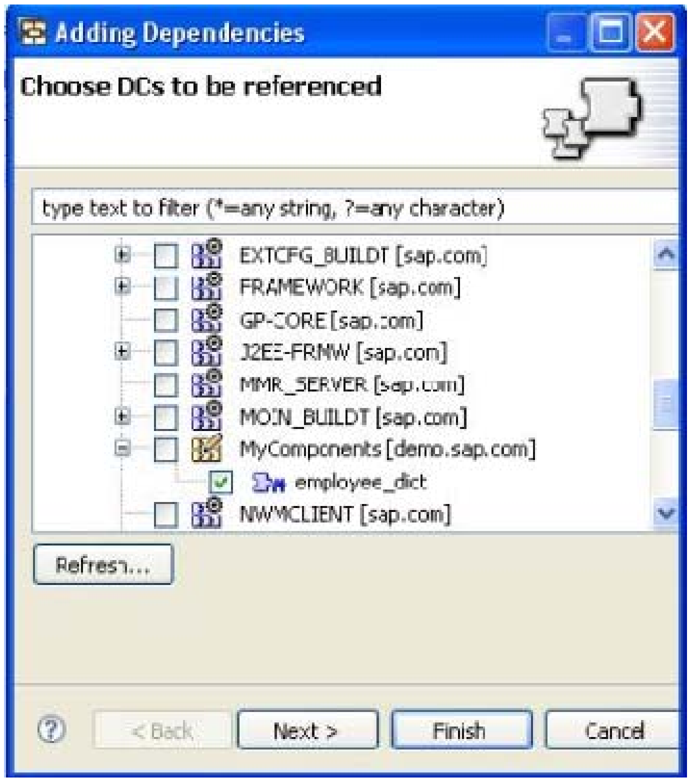
Open the Development Infrastructure perspective by choosing *Window →Open Perspective →Development Infrastructure.*

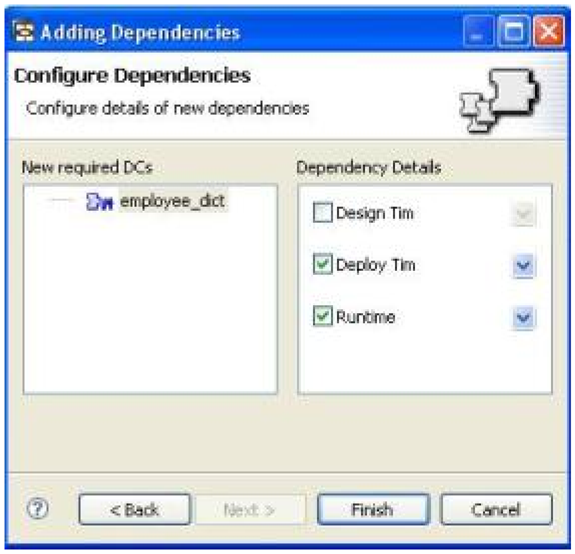
Under *Local Development/ My Components* select *employee_ear.*

Under Component Properties select *Dependencies* and choose *Add.*



Under *Local Development/ My Components* select the *employee_dict* indicator, then under *New required DCs* select *employee_dict* again and choose *Next,* then select the *Runtime* and *Deploy Time* indicators and choose *Finish.*



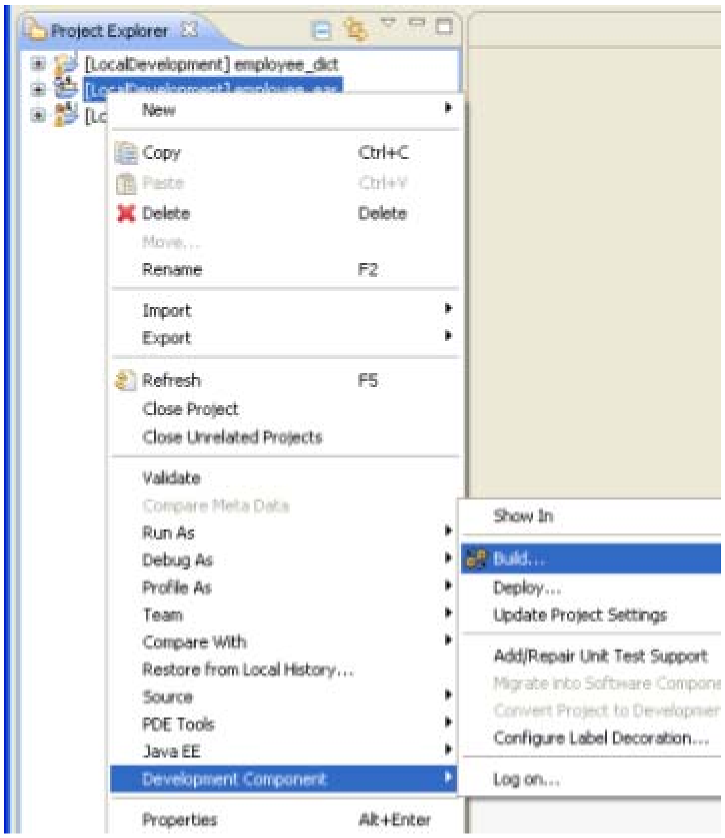


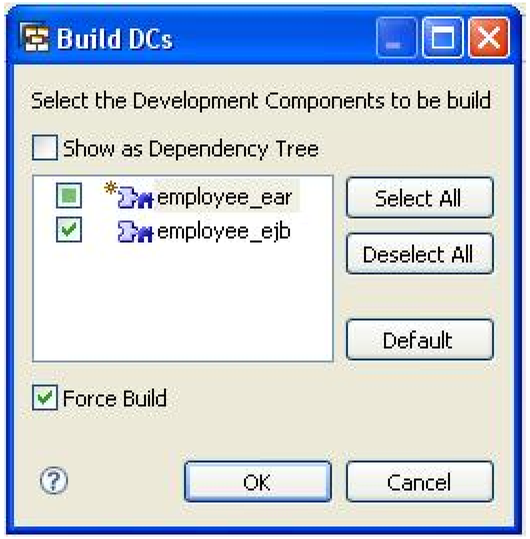
Building and deploying the application

Before deploying your application, make sure you have configured a running SAP system (SAP AS Java) via *Window/Preferences* of your SAP NetWeaver Developer Studio.

From the context menu of *employee_ear* select *Development Component →Build.*

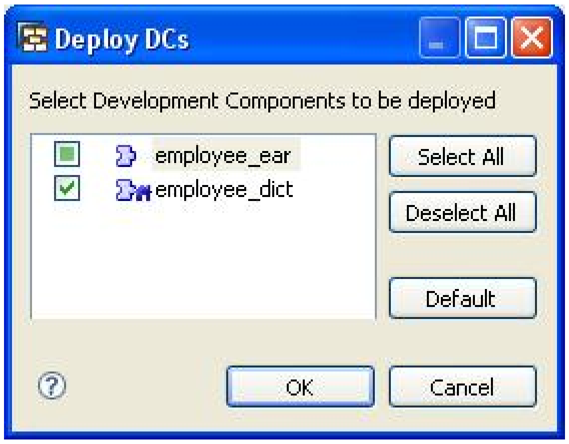
Select *employee_ejb* (employee_ear is selected by default) and choose *OK.*





From the context menu *of employee_ear* select *Development Component →Deploy*.

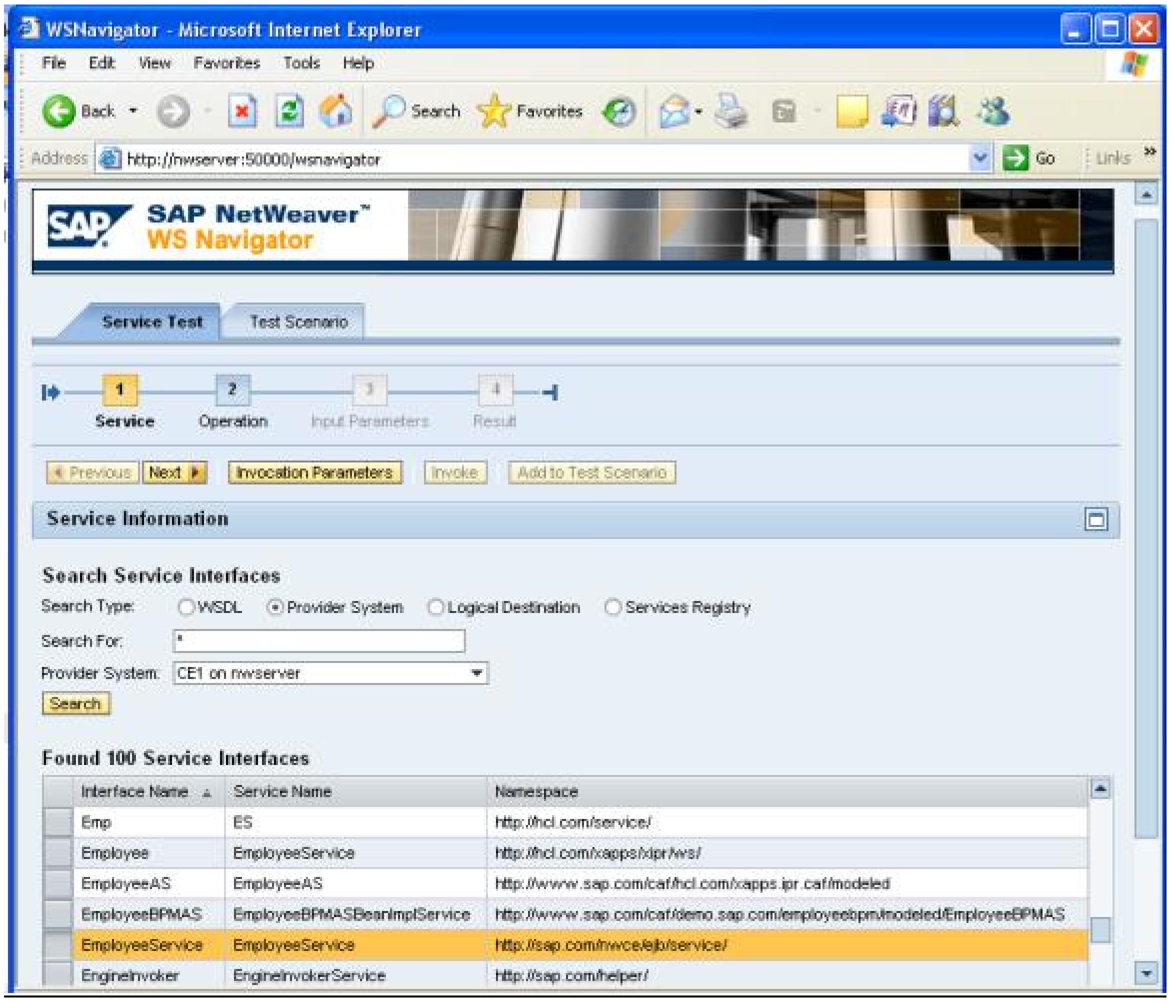
Select *employee_dict* (employee_ear is selected by default) and choose *OK.*



Test the EmployeeService in WS Navigator

Open the WS Navigator by typing the URL http://<server>:50000/wsnavigator.

Search your service (EmployeeService) by Provider System or WSDL.


WSNavigator - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://nwserver:50000/wsnavigator
SAP NetWeaver WS Navigator
Service Test
Test Scenario
1 Service
2 Operation
3 Input Parameters
4 Result
Previous Next Invocation Parameters Invoke Add to Test Scenario
Service Information
Search

Search Service Interfaces

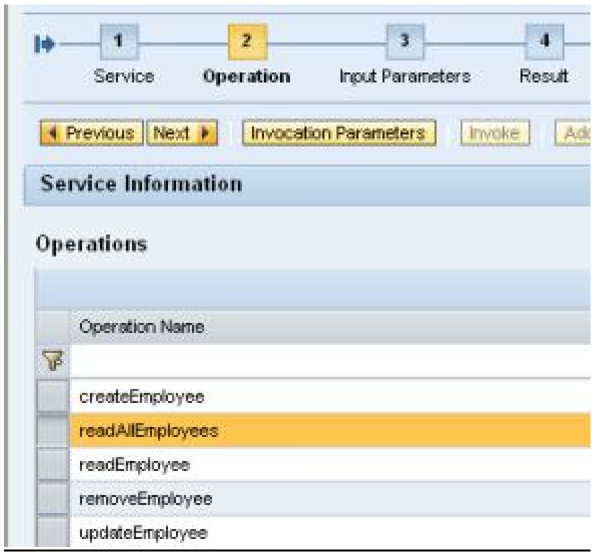
Search Type: WSDL [✓ Provider System] Logical Destination Services Registry

Search For: *

Provider System: CE1 on nwserver

Found 100 Service Interfaces

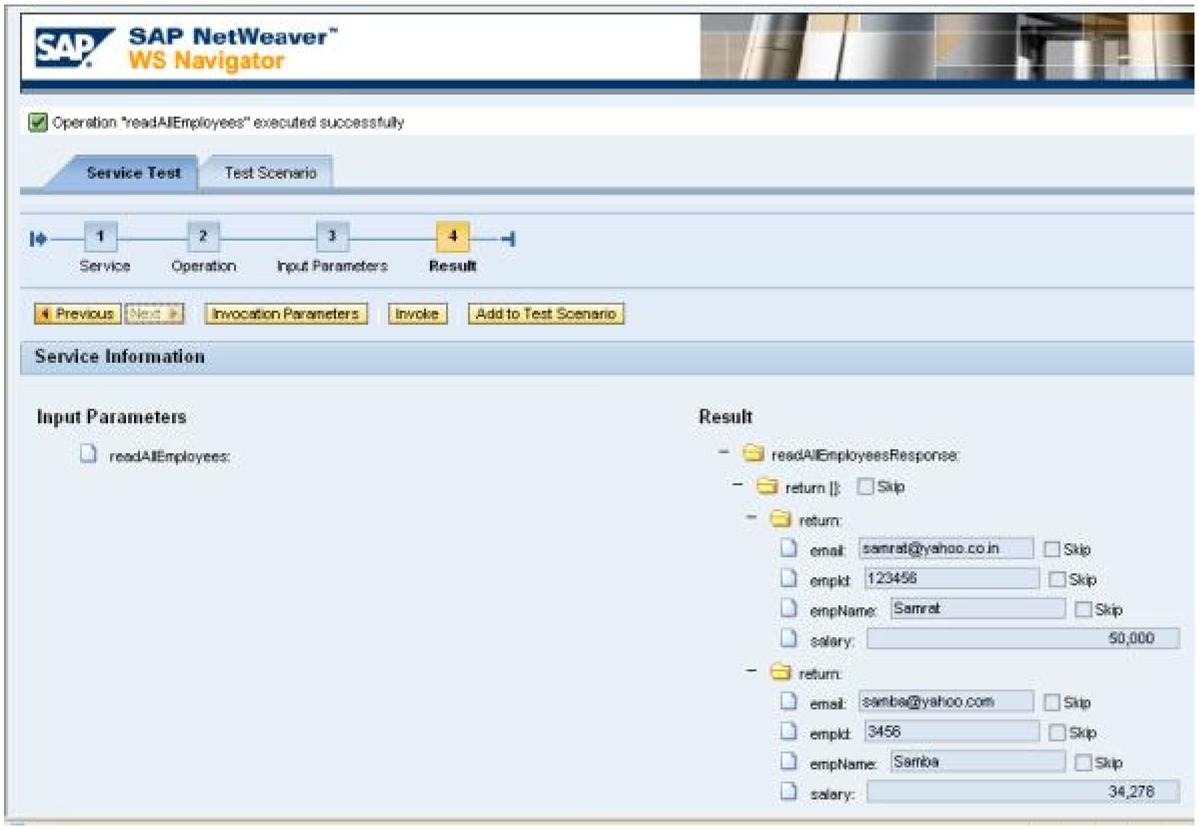
Interface Name	Service Name	Namespace
Emp	ES	http://hcl.com/service/
Employee	EmployeeService	http://hcl.com/xapps/xipr/ws/
EmployeeAS	EmployeeAS	http://www.sap.com/caf/hcl.com/xapps.ipr.caf/modeled
EmployeeBPMAS	EmployeeBPMASBeanImplService	http://www.sap.com/caf/demo.sap.com/employeebpm/modeled/EmployeeBPMAS
EmployeeService	EmployeeService	http://sap.com/nwce/ejb/service/
EngineInvoker	EngineInvokerService	http://sap.com/helper/



Operations

Operation Name
createEmployee
readAllEmployees
readEmployee
removeEmployee
updateEmployee

Now, you can execute the operations of EmployeeService.



Operation "readAllEmployees" executed successfully

Service Information

Input Parameters

readAllEmployees:

Result

readAllEmployeesResponse:
return []: Skip
return:
email: samrat@yahoo.co.in Skip
empId: 123456 Skip
empName: Samrat Skip
salary: 50,000
return:
email: samba@yahoo.com Skip
empId: 3456 Skip
empName: Samba Skip
salary: 34,278

Related Content

Getting Started with Java Persistence API and SAP JPA 1.0

Basics of the Java Persistence API – Understanding the Entity Manager

Basics of the Java Persistence API – Defining and Using Relationships

SAP NetWeaver Composition Environment 7.1

For more information, visit the Java homepage.

Disclaimer and Liability Notice