An Alternative Architecture of Composites using EJB

Applies to:

SAP NetWeaver CE 7.1 (SP 05), JDK 1.5_12 For more information, visit the Composition homepage.

Summary

This document will provide complete reference and guide for lose coupling between different layers. And if we want to change our ERP system we don't need to change our whole application. We can switch between Enterprise Service and Web service without touching our code. The whole switch framework code is reusable in any switching based application. The Enterprise service/ Web service implementation is done via java proxy classes.

Author: Biplab Ray

Company: IBM India Pvt Ltd.

Created on: 08th August 2008

Author Bio

Biplab Ray is working for IBM India Pvt Ltd as System Engineer and development of composite applications using CAF, GP, WebDynpro. He also works in technologies like Java/J2EE and has depth understanding on SOA, Webservice, Enterprise service, XML.

Table of Contents

Introduction

Composite applications are defined as applications that sit on top of other applications and reuse their functionality by service calls. The layers making up a composite application are:

- Backend Layer
- Backend Connectivity Layer
- Backend Abstraction Layer
- Business Logic Layer
- User Interface Layer
- Process Layer
- Portal Layer

Architecture Goals

A composite application has to fulfill the following characteristics:

- Own lifecycle
- Loosely coupled with backend system
- Integration with backend via stateless service calls
- Backend independency
- Easy to adopt/enhance for customers
- Model-driven Architecture

Model-driven Architecture

Model-driven development should be used on all layers of a composite. SAP NetWeaver Composition Environment comprises the following tools supporting model-driven development:

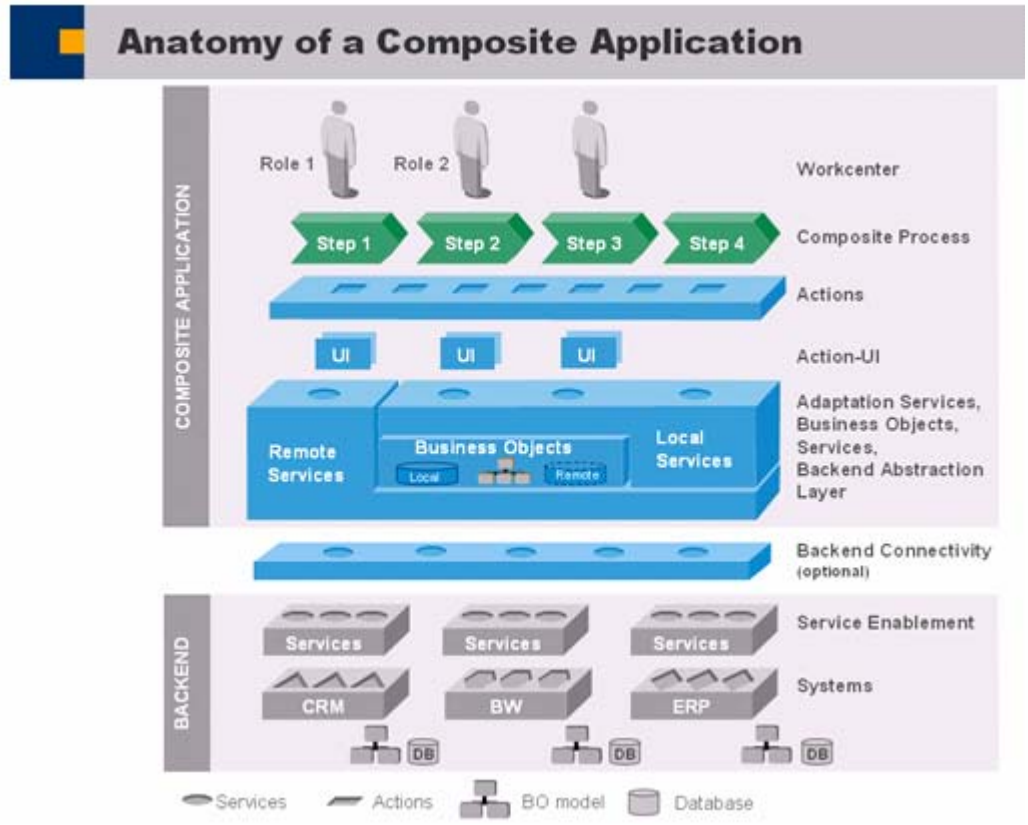
- CAF Core/ EJB application for business objects and service composition
- Visual Composer for simple, straightforward online user interface
- WebDynpro for more advanced online interface
- Adobe Interactive Forms for form based offline user interface
- Guided Procedures for process modeling and orchestration

In my article I am discussing with EJB for business objects service composition.

Architecture

Basic Architecture of A Composite

The anatomy of the composite (see Figure 1 for details) is a high-level description and is independent from the tools used for implementation of the different layers.



Short Description of Layers

Backend Layer

The backend layer provides the data and functionalities available in the backend systems, which are used in the composite, by services. As figure 1 already indicates, the development of the services is not part of the development of the composite itself. Instead the services have to be provided by the backend systems, which contain the core business logic and data.

Backend Connectivity Layer

As composite applications are defined as applications sitting on top of other applications which reuse existing functionality via service calls, the question comes up how the consumption of those services technically looks like. The technology of choice is the **direct connection of the business logic layer with the backend via standards-based Web Services technology** and it is indeed the recommended solution of choice for composite applications.

By default, the composite's business objects and services layer makes direct calls to the provided backend enterprise services and by this abstracts the calls to the higher layers (UI-/process layer). However, it is also possible for the higher layers to call backend services directly. Due to the lost flexibility this architecture

should be avoided if possible. In the target Enterprise SOA based architecture the technology of choice to connect composites with the backend systems is Web Services.

However, independent of the solution chosen for overcoming these connectivity challenges, the main idea of composites is in all cases the same: no technology related interfaces appear on higher levels (e.g. business logic layer, process layer)! The higher levels can rely on stable interfaces which will not change in case the underlying technology, by which the services are called, changes. Following this approach a decoupling of functionality and technology will be achieved.

Business Logic and Backend Abstraction Layer

Within the business logic layer the business logic and the business objects specific for the composite are implemented. The **unified business object model** provides the flexibility to make transparent usage of business objects with local or remote persistency. The **unified service model** provides service abstraction and shields higher layers from service implementation details making them replaceable. So it is recommended to make use of this abstraction in the UI- and process layer to benefit from its flexibility to adapt the final solution to different target IT-landscapes.

In backend abstraction layer consist only the interfaces which will be implemented into the backend connectivity layer. This Backend Abstraction Layer (BAL) is implemented because at the time of development there may be all ES are not available the development is continue with WS, when all ES are available we can switch between ES and WS with the help of Switching Framework application.

User Interface Layer

New user interfaces can be created on top of the services provided by the business logic layer. By only using services of the business logic layer a clear decoupling between the UI and the business logic is implied. The user interface layer comprises online as well as offline UIs.

On this layer model driven development shall be used to model the User Interface screens including the screen flow and the possible user interaction.

Process Logic Layer

Within the process logic layer it is defined, which process steps are executed in which sequence by which roles and how the context data of the process is passed between the participating process steps.

Within each process step exactly one action is executed. An action is a wrapper around a so-called callable object which either represents a user interface or a service. Such an action has importing and exporting parameters. To indicate how data can be passed from a previous action (a) to the next action (b), the appropriate parameters can be assigned to each other on the process layer (output parameters of action (a) are mapped to the input parameters of action (b)).

One of the benefits of actions is that they decouple process steps from services and user interfaces to allow business experts to model processes on a non technical level.

On this layer model driven development shall be used to model the process, the process steps and the actions thus avoiding hard-coded process flows within the business logic layer.

Portal Access Layer

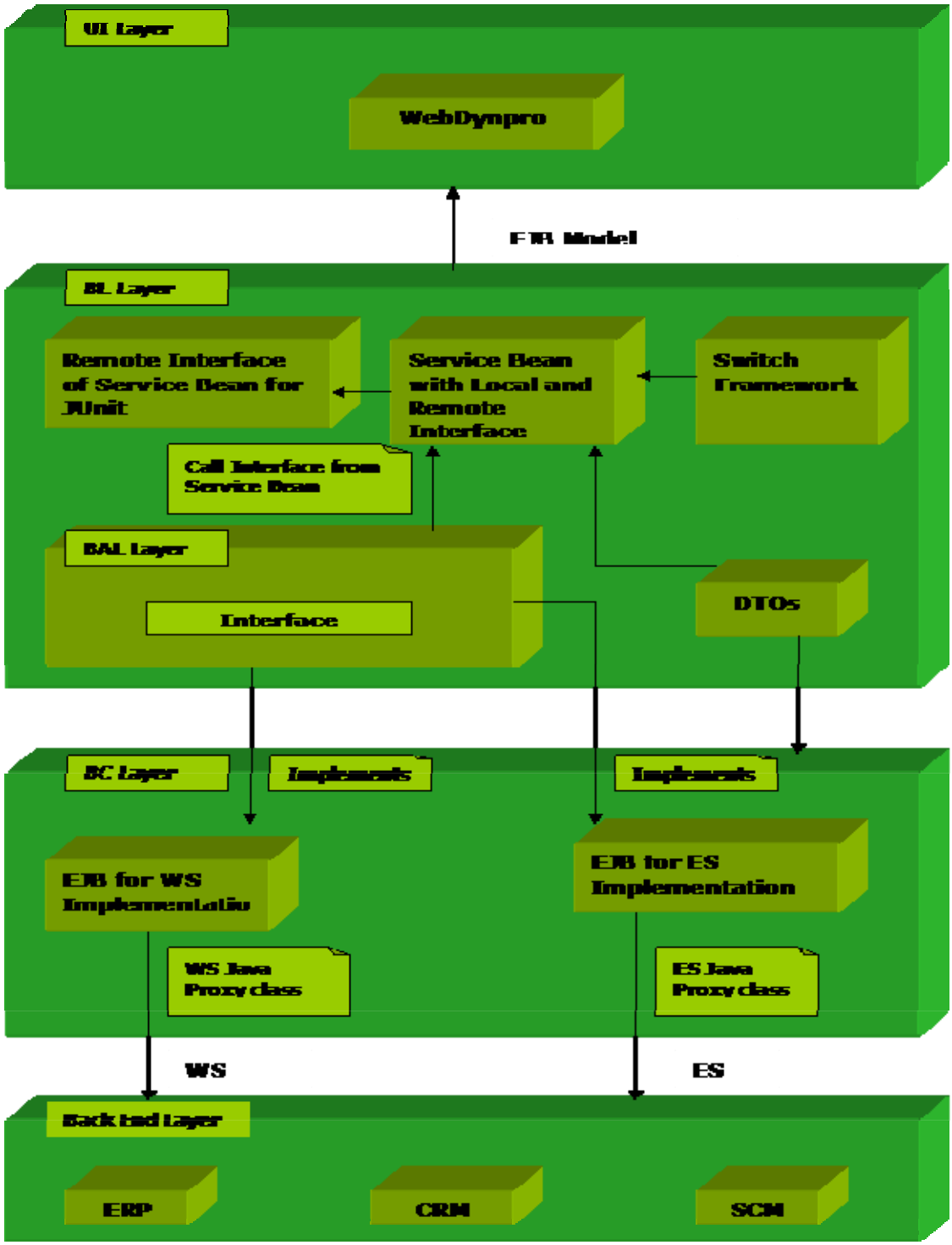
Within the portal access layer the user interfaces and processes are provided in a role-based manner using the work- and control-center concept. Let's have a closer look at this concept: a user may have different roles. For each role a work center exists. The work center provides a role and task-oriented view of data and activities (e.g. customer service care, employee self service, purchase order management). It consists of a set of pages organizing and supporting the user's activities in a certain area (work set), e.g. the work set 'employee self service' with the activities e.g. leave request and address change. On the other hand side the control center provides an overview of all work centers the user is utilizing.

Besides the provisioning of the control center/work center concept by the portal layer, it also takes care of the central entry point for work items the end user has to care of. This is covered by the so-called Universal Work List which is needed as the means of forwarding follow-up process steps between users.

Implementation

Now I am going to implement a whole application using the following architecture diagram for the application.

The architecture is little bit modified because in this application there is no process layer and portal layer.



Architectural Diagram for our application.

Switch Framework (Between ES and WS)

Switch framework is the combination of two project one CAF project for data holding (**arch.confcore**) and other one is for get service type ES/WS and jndi name from the CAF project, it is an ejb project (**arch/config**) this project should have an ear project, I have marched this ejb project to the **arch/blapp** ear project. Due to this reason in previous diagram **Switch Framework** is in the BL Layer.

Before we are going to start our actual application we have to implement the switch framework application using Composite Application Framework where we can switch between Enterprise Service and Web Service.

The **arch/config** ejb project is described later.

If our client wants to switch from Web Service to Enterprise Service they don't have to change codes only they have to change only WS to ES into this application under Service Browser of CAF Application.

Now I am going to create the switch frame work application.

In this application we need two Business Objects.

1. Custom Property
2. Standard Property

Custom Property BO for customization our application and Standard Property for standard operations of the application. The name of the application is **arch.confcore**

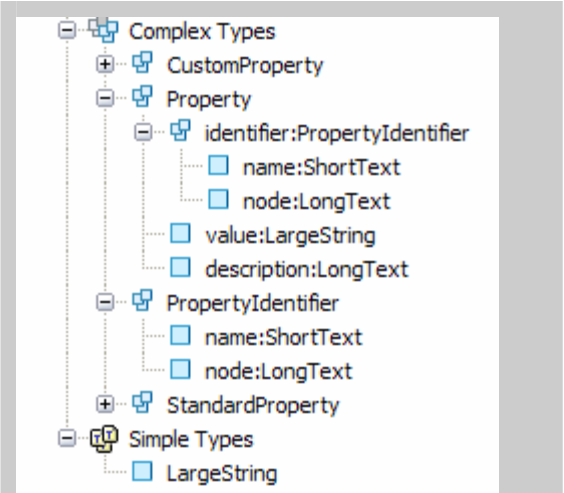
Custom Property: The purpose of this Business Object is to hold the data for our switch means ES to WS by value attribute of Property Structure. If we want to change our whole application from WS implementation to ES implementation then only we have to change the Value attribute via CAF service browser. There should be two type of implementation into our ejb (under **arch/bc** project) like WS and ES there should be two ejb for two implementation (WS and ES). Next I will explain the whole CAF service browser for Custom Property BO.

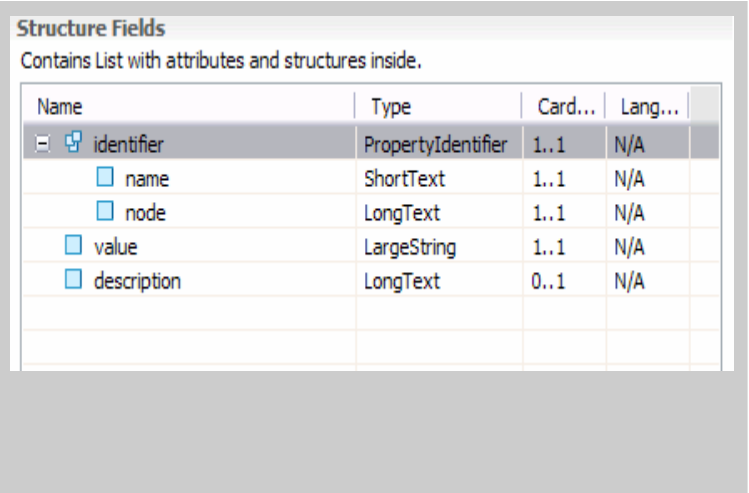
Standard Property: The purpose of this Business Object is to hold the data for our ejbs interface name and jndi lookup strings. With the help of Custom Property BO we can understand which type of is the current application is it WS or ES implementation application. In this BO I have mention special type of interface string means if there are two type of ejb implementation one for WS and one for ES but interface is same which is implemented into the both ejb. In this BO I have entered the **node** value means interface value with the extension WS or ES like: **Full Qualified interface name + WS or ES** under **node** attribute, and **value** attribute hold the data for the **jndi** lookup of the **ejb** which is **implemented** the **interface**. In case of ES it will search only with extension of the interface with ES. Next I will explain the whole CAF service browser for Standard Property BO.

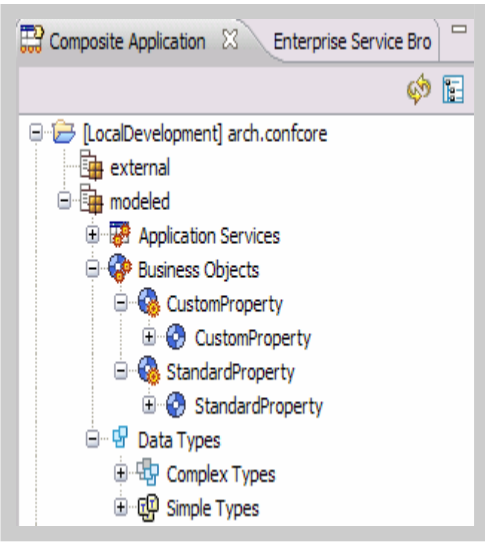
Open your CE IDE. To create the CAF project please does as follows:

Please go to File -> Project -> Development Component -> Composite Application -> My Components. Then provide the required parameters as project name, vendor name, Domains etc.

Please create Data Types before create BOs.



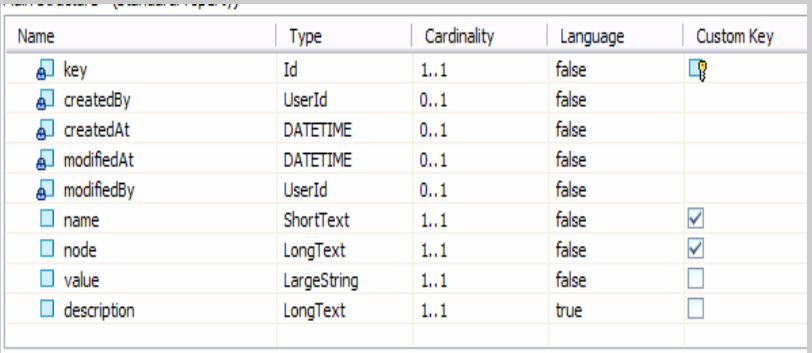




The CAF Application Name is : arch.confcore

Now we need to add some attributes and operations to these two business objects.

The attributes for two business objects are as follows



Now we have to create some operations for BO: Custom and Standard Property.

Custom Property Operations:

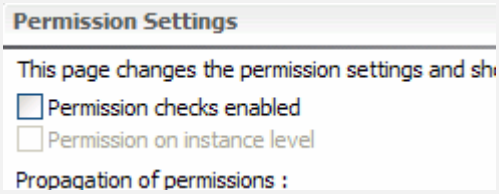
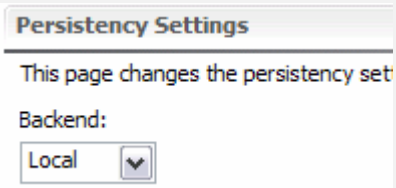
Operation Name	Visibility	Return Type	Input Parameters(Select)
findByCustomKeys	public	Output:CustomProperty with cardinality : 0..n	name, node
findByName	public	Output:CustomProperty with cardinality : 0..n	name
findByNode	public	Output:CustomProperty with cardinality : 0..n	node

Standard Property Operations:

Operation Name	Visibility	Return Type	Input Parameters(Select)
findByCustomKeys	public	Output:StandardProperty	name, node

		with cardinality : 0..n	
findByName	public	Output:StandardProperty with cardinality : 0..n	name
findByNode	public	Output:StandardProperty with cardinality : 0..n	node

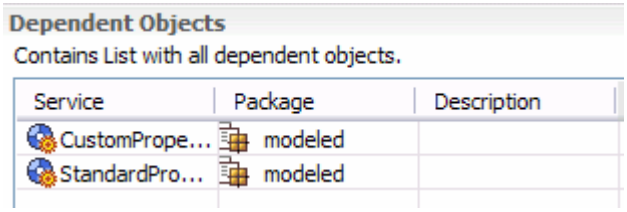
The Permission & Persistency for two BOs should be as follows:

Permission	Persistency
	

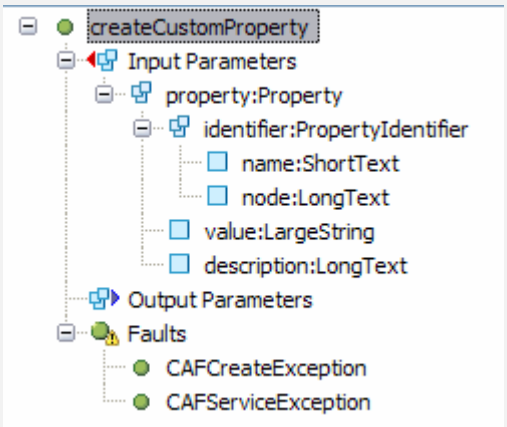
Now we have to create an application service with few operations.

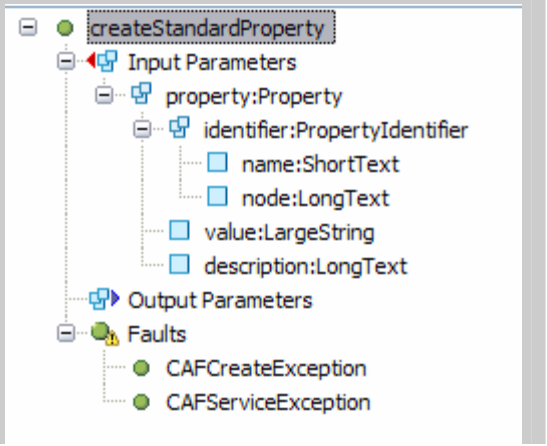
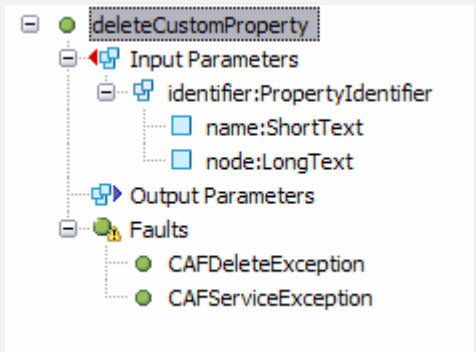
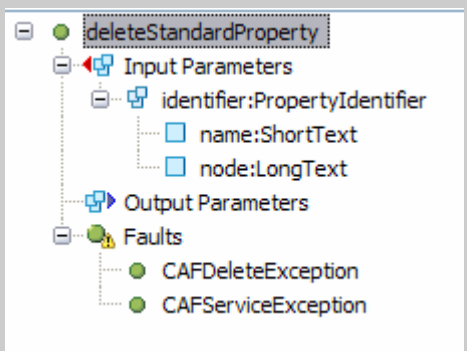
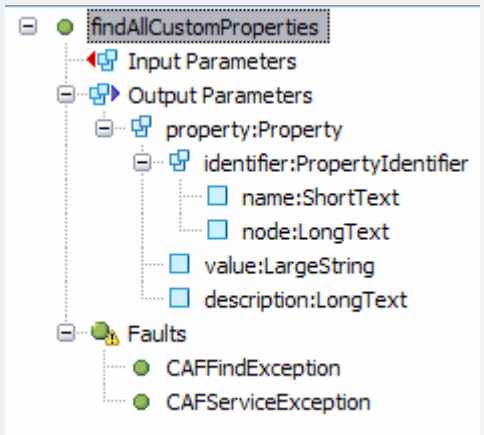
In my application the name of the application service: **ConfigurationCore** this is the only application service in my CAF project.

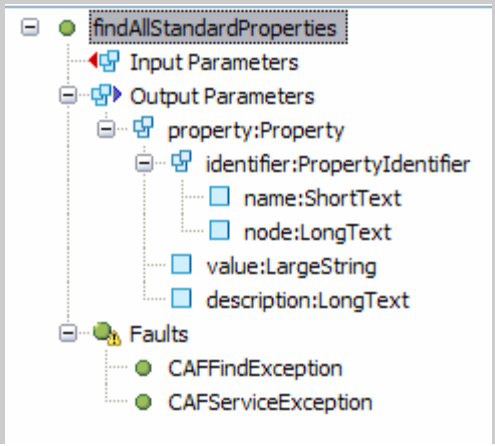
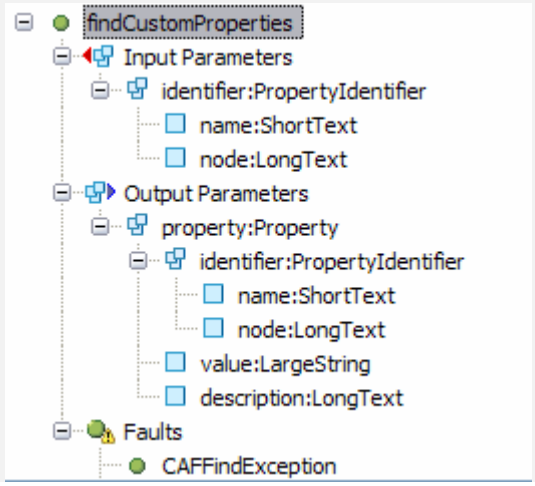
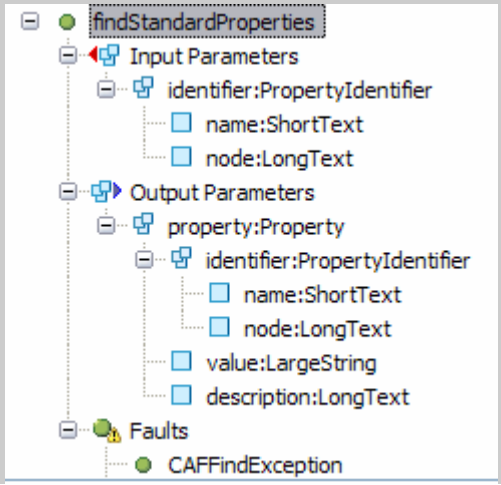
We have to add our two BOs to the application service.

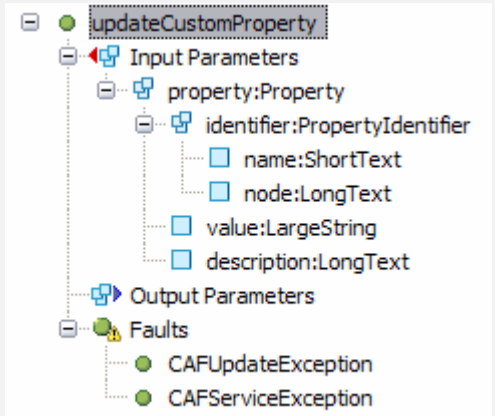
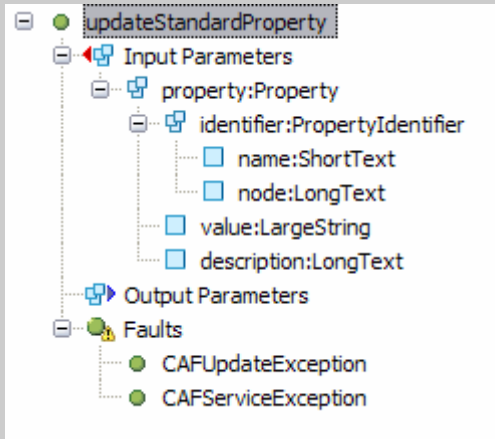
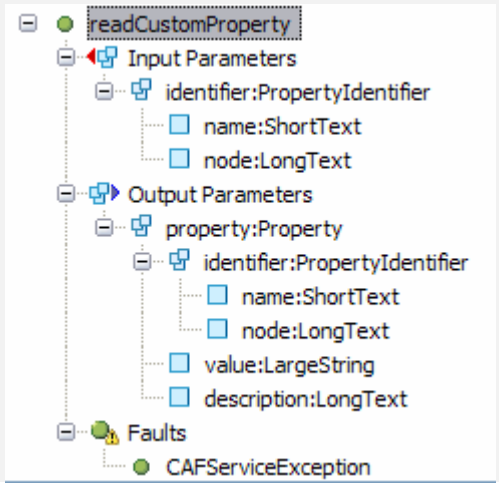


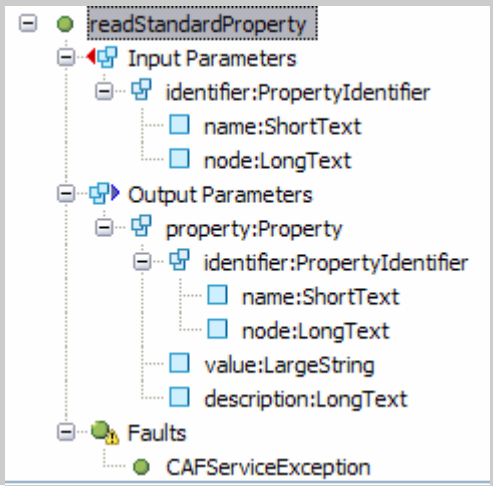
The operations of the application service are as follows:

Operation Name	Transaction Type	Permission	Implemented	Details Description
createCustomProperty **Purpose**: This method creates the custom property, in the service browse when we click on **findAll** method of Custom Property, where we can add name, node, value, description values and press save button it will create the property in the caf db.	Required	Not Checked	Checked	

createStandardProperty **Purpose**: Similar to previous only input parameter values are different.	Required	Not Checked	Checked	
deleteCustomProperty **Purpose**: It will delete the custom property.	Required	Not Checked	Checked	
deleteStandardProperty **Purpose**: This method deletes the standard property.	Required	Not Checked	Checked	
findAllCustomProperties **Purpose**: This method returns the list of the custom property. The value of name, node, value, description.	Required	Not Checked	Checked	

findAllStandardProperties **Purpose**: Similar to previous only output values are different.	Required	Not Checked	Checked	
findCustomProperties **Purpose**: If we want to find a specific Custom property we have to pass the name and node value as input parameters it will return the specific custom property.	Required	Not Checked	Checked	
findStandardProperties **Purpose**: Similar to previous only output values are different.	Required	Not Checked	Checked	

updateCustomProperty **Purpose**: If we want to update a custom property, we can do the same via this method.	Required	Not Checked	Checked	
updateStandardProperty **Purpose**: Similar to previous. Only input parameter values rae different.	Required	Not Checked	Checked	
readCustomProperty **Purpose**: This method read only the specific custom property via some input values like name and node value of a specific custom property; it will return the name, node, value, and description of the custom property.	Required	Not Checked	Checked	

readStandardProperty **Purpose**: Similar to previous. Only the input and output values are different.	Required	Not Checked	Checked	

Now I am going to implement the Application Service methods.

Method Signature	Method Body

protected Collection<Property> convertToPropertyCollection(Collection<?> xProps) **throws** CAFServiceException	Collection<Property> collection = **new** ArrayList<Property>();
	Iterator<?> iterator = xProps.iterator();
	while(iterator.hasNext()){
	Property property = **null**;
	Object object = iterator.next();
	if(object **instanceof** StandardProperty){
	property = **this**.convertToProperty((StandardProperty)object);
	}**else if** (object **instanceof** CustomProperty) {
	property = **this**.convertToProperty((CustomProperty)object);
	}**else**{
	throw new CAFServiceException(*_location*);
	}
	collection.add(property);
	
	}
	
	return collection;

protected Property convertToProperty(StandardProperty standardProperty)	Property property = **new** Property();
	// Property Identifier
	PropertyIdentifier propertyIdentifier = **new** PropertyIdentifier();
	propertyIdentifier.setName(standardProperty.getName());
	propertyIdentifier.setNode(standardProperty.getNode());
	property.setIdentifier(propertyIdentifier);
	// For Other Attributes
	property.setValue(standardProperty.getValue());
	property.setDescription(standardProperty.getDescription());
	}
	return property;

protected Property convertToProperty(CustomProperty customProperty)	Property property = **new** Property();
	// Property Identifier
	PropertyIdentifier propertyIdentifier = **new** PropertyIdentifier();
	propertyIdentifier.setName(customProperty.getName());
	propertyIdentifier.setNode(customProperty.getNode());
	property.setIdentifier(propertyIdentifier);
	// For Other Attributes
	property.setValue(customProperty.getValue());
	property.setDescription(customProperty.getDescription());
	// For Language
	Set<String> setLanguage = customProperty.getDescriptionLanguages();
	if(!setLanguage.isEmpty()){
	for (String string : setLanguage) {
	property.setDescription(customProperty.getDescription(string));
	}
	}
	return property;

public java.util.Collection<com.ibm.arch.confcore.types.Property> findAllCustomProperties()**throws** com.sap.caf.rt.exception.CAFFindException, com.sap.caf.rt.exception.CAFServiceException	Collection<CustomProperty> collection = **this**.getCustomPropertyService().findAll(); **return this**.convertToPropertyCollection(collection);
public com.ibm.arch.confcore.types.Property readStandardProperty(com.ibm.arch.confcore.types.PropertyIdentifier identifier)**throws** com.sap.caf.rt.exception.CAFServiceException	StandardProperty standardProperty = **this**.getStandardPropertyService().readByCustomKeys(identifier.getName(), identifier.getNode()); **return this**.convertToProperty(standardProperty);
public java.util.Collection<com.ibm.arch.confcore.types.Property> findAllStandardProperties()**throws** com.sap.caf.rt.exception.CAFFindException, com.sap.caf.rt.exception.CAFServiceException	Collection<StandardProperty> collection = **this**.getStandardPropertyService().findAll(); **return this**.convertToPropertyCollection(collection);
public void deleteCustomProperty(com.ibm.arch.confcore.types.PropertyIdentifier identifier)**throws** com.sap.caf.rt.exception.CAFDeleteException, com.sap.caf.rt.exception.CAFServiceException	CustomProperty customProperty = **this**.getCustomPropertyService().readByCustomKeys(identifier.getName(), identifier.getNode()); **this**.getCustomPropertyService().delete(customProperty);
public java.util.Collection<com.ibm.arch.confcore.types.Property> findCustomProperties(com.ibm.arch.confcore.types.PropertyIdentifier identifier)**throws** com.sap.caf.rt.exception.CAFFindException, com.sap.caf.rt.exception.CAFServiceException	QueryFilter queryFilterName = QueryFilterFactory.*createFilter*(identifier.getName()); QueryFilter queryFilterNode = QueryFilterFactory.*createFilter*(identifier.getNode()); Collection<CustomProperty> collection = **this**.getCustomPropertyService().findByCustomKeys(queryFilterName, queryFilterNode); **return this**.convertToPropertyCollection(collection);
public void createCustomProperty(com.ibm.arch.confcore.types.Property property)**throws** com.sap.caf.rt.exception.CAFServiceException, com.sap.caf.rt.exception.CAFCreateException	CustomProperty customProperty = **this**.getCustomPropertyService().create(property.getIdentifier().getName(), property.getIdentifier().getNode(), property.getValue(), property.getDescription()); **this**.getCustomPropertyService().update(customProperty);
public java.util.Collection<com.ibm.arch.confcore.types.Property> findStandardProperties(com.ibm.arch.confcore.types.PropertyIdentifier identifier) **throws** com.sap.caf.rt.exception.CAFFindException, com.sap.caf.rt.exception.CAFServiceException	QueryFilter queryFilterName = QueryFilterFactory.*createFilter*(identifier.getName()); QueryFilter queryFilterNode = QueryFilterFactory.*createFilter*(identifier.getNode()); Collection<CustomProperty> collection = **this**.getCustomPropertyService().findByCustomKeys(queryFilterName, queryFilterNode); **return this**.convertToPropertyCollection(collection);
public void createStandardProperty(com.ibm.arch.confcore.types.Property property) **throws** com.sap.caf.rt.exception.CAFServiceException, com.sap.caf.rt.exception.CAFCreateException	StandardProperty standardProperty = **this**.getStandardPropertyService().create(property.getIdentifier().getName(), property.getIdentifier().getNode(), property.getValue(), property.getDescription()); **this**.getStandardPropertyService().update(standardProperty);
public void deleteStandardProperty(com.ibm.arch.confcore.types.PropertyIdentifier identifier) **throws** com.sap.caf.rt.exception.CAFDeleteException, com.sap.caf.rt.exception.CAFServiceException	StandardProperty standardProperty = **this**.getStandardPropertyService().readByCustomKeys(identifier.getName(), identifier.getNode()); **this**.getStandardPropertyService().delete(standardProperty);
Public com.ibm.arch.confcore.types.Property readCustomProperty(com.ibm.arch.confcore.types.PropertyIdentifier identifier) **throws** com.sap.caf.rt.exception.CAFServiceException	CustomProperty customProperty = **this**.getCustomPropertyService().readByCustomKeys(identifier.getName(), identifier.getNode()); **return this**.convertToProperty(customProperty);
public void updateCustomProperty(com.ibm.arch.confcore.types.Property property) **throws** com.sap.caf.rt.exception.CAFUpdateException, com.sap.caf.rt.exception.CAFServiceException	CustomProperty customProperty = **this**.getCustomPropertyService().readByCustomKeys(property.getIdentifier().getName(), property.getIdentifier().getNode()); customProperty.setDescription(property.getDescription()); customProperty.setValue(property.getValue()); **this**.getCustomPropertyService().update(customProperty);
public void updateStandardProperty(com.ibm.arch.confcore.types.Property property) **throws** com.sap.caf.rt.exception.CAFUpdateException, com.sap.caf.rt.exception.CAFServiceException	StandardProperty standardProperty = **this**.getStandardPropertyService().readByCustomKeys(property.getIdentifier().getName(), property.getIdentifier().getNode()); standardProperty.setDescription(property.getDescription());

```
standardProperty.setValue(property.getValue());
this.getStandardPropertyService().update(standardProperty);
```

After deploy the CAF application.

Please log into the server for caf. **http://servername:port/caf**

Then click on the **TestTools** hyperlink the second option. Then please go to the application as shown in the below fig.

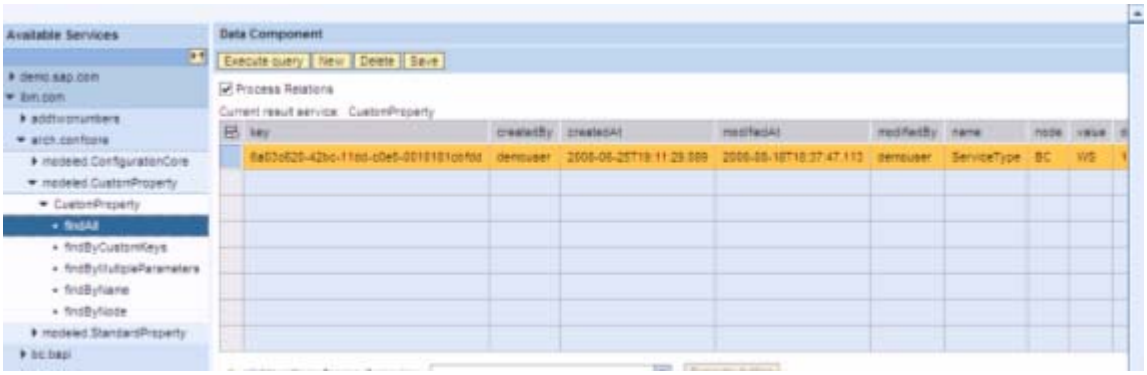
We will get the following screen under caf service browser and we have to enter the following parameters for Custom Property operation name: **findAll**, please Press **New** Button and then enter the following values.

Name: ServiceType

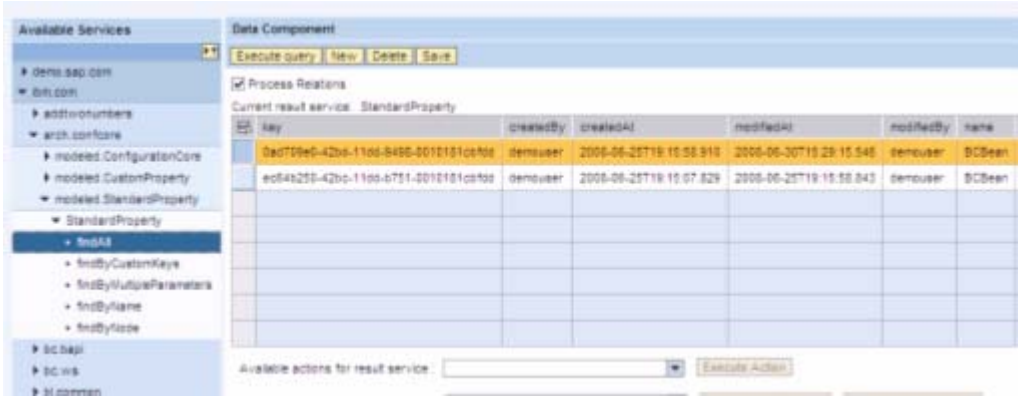
Node: BC

Value: ES/WS

Description: 1



Please open Standard Property and then click on **findAll** method and press New Button and enter the following values:

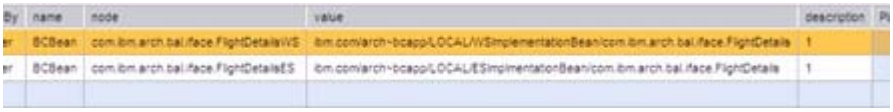


Name: BCBean

Node: com.ibm.arch.bal.iface.FlightDetails**WS/ES**

Value: ibm.com/arch~bcapp/LOCAL/ **WS or ES** ImplementationBean/com.ibm.arch.bal.ifce.FlightDtails

Description: 1

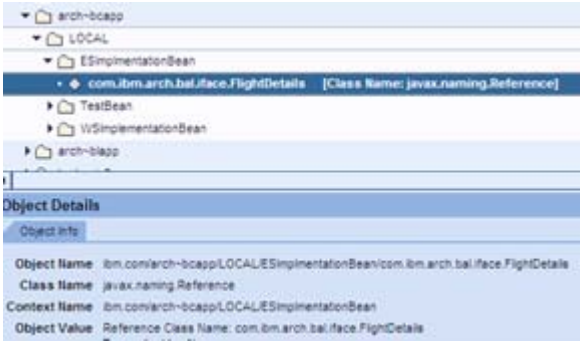


To find the above values you have to logon to the server via nwa and if your application is deployed into the server then follow the steps to get the values.

To logon to the server via nwa, http://servername:port/nwa

Please go to the Problem Management Tab - > JNDI Browser

Then please go to your desired application. And copy the interface which is implemented under the desired EJB, and copy Object Name i.e Value



Business Logic and Backend Abstraction

To implement business logic and backed abstraction layer (bal) we have to create a EJB application project.

In this project we have to create service bean for ui layers with remote and local interface. This service bean is necessary to implement to connect the beans which will be implement the interfaces in bal layer in the bc(backend connectivity layer) layer bean are call backend layer via WS/ES.

Please create an EJB application with name arch/bl and one ear project with the following name arch/blapp.

To create an ejb project please follow the steps:

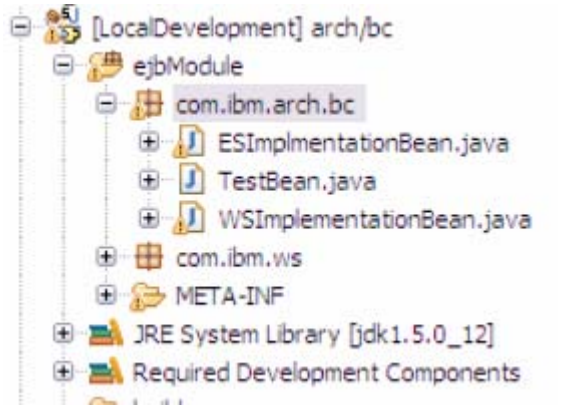
Please open your CE IDE.

Then Please go to the File -> Project - > Development Component - > EJB Module - > My Components. Then provide the required input parameters like name of the project, vendor name etc.

After creating the ejb project please go to the java perspective and open the project you will get ejbModule folder right click on that folder the follow the steps:

Package-> provide the package name. Then right click on the package new -> other -> type ejb-> choose EJB Session Bean 3.0. Then provide the required information like ejb name etc, please don't select remote or local interface, press next button and choose the interface created in the bal layer and press finish button. It will create your ejb with the interface in the bal layer.

Then it will be as below:

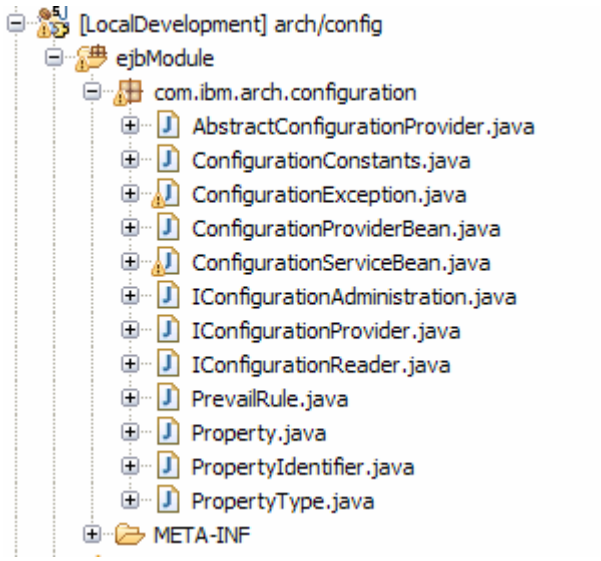


Note: Every EJB project their should be a **ear** project. By the above way you can create the ear project and add the required ejb project to this application project.

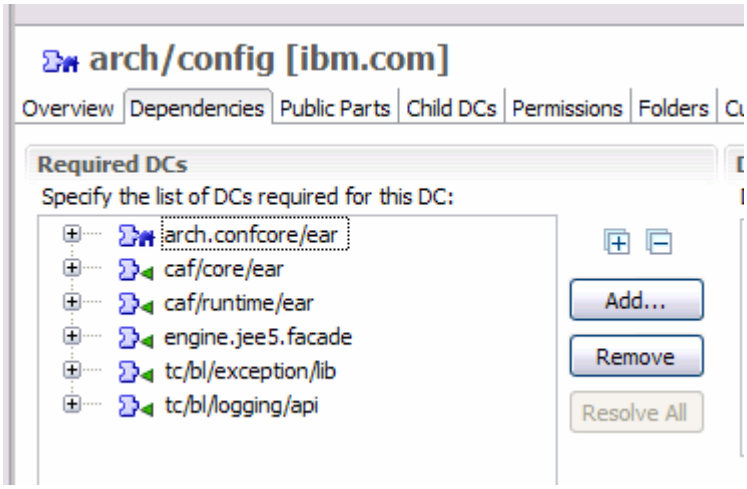
Then create an **arch/config** EJB application and attach this application to the arch/blapp ear project, we need this ejb project because here we will pass our interface name from service bean and here we recognize the service type i.e ES/WS and get the jndi name (ibm.com/arch~bcapp/LOCAL/WSImplementationBean/com.ibm.arch.bal.ifce.FlightDtails).

In this project **arch/config** I am calling the caf application service and determine the type of application WS/ES and get the corresponding jndi names for each ejb.

Please create the following java files as shown in the below picture.



Before start coding please add **arch.confcore/ear** to the arch/config ejb project as below.



Class Name: AbstractConfigurationProvider.java it is a abstract class

This class implements: IConfigurationReader

Parameters: @EJB

```
private IConfigurationAdministration configurationService;
```

Method Name	Method Body

private final Property readProperty(PropertyIdentifier identifier, PropertyType type, boolean faultTolerant) throws ConfigurationException	Property property = null;
	try{
	if(type == PropertyType.*STANDARD*){
	property = configurationService.readStandardProperty(identifier);
	}else if (type == PropertyType.*CUSTOM*) {
	property = configurationService.readCustomProperty(identifier);
	}else{
	throw new ConfigurationException("Property Type" + type + "not known");
	}
	}catch (Exception e) {
	// TODO: handle exception

	if(!faultTolerant){
	throw new ConfigurationException("Exception During Retrival",
	e);
	}
	}
	
	return property;

public final Property getProperty(String node, String name) throws ConfigurationException	return this.getProperty(node, name, PrevailRule.*CUSTOM*, false);
public final Property getProperty(String node, String name, boolean fallback) throws ConfigurationException	return this.getProperty(node, name, PrevailRule.*CUSTOM*, fallback);
public Property getProperty(String node, String name,PrevailRule prevailRule) throws ConfigurationException	return this.getProperty(node, name, prevailRule, false);

public final Property getProperty(String node, String name,
PrevailRule prevailRule, boolean fallback)
throws ConfigurationException
	PropertyIdentifier identifier = new PropertyIdentifier();
	identifier.setName(name);
	identifier.setNode(node);
	Property property = null;
	if(prevailRule.equals(PrevailRule.*STANDARD*)){
	property = this.readProperty(identifier, PropertyType.*STANDARD*,
	true);
	if(property == null){
	property = this.readProperty(identifier, PropertyType.*CUSTOM*,
	true);
	}
	if(property == null){
	throw new ConfigurationException("Fail to Read Property");
	}
	}else{
	property = this.readProperty(identifier, PropertyType.*CUSTOM*,
	true);
	if(property == null){
	property = this.readProperty(identifier, PropertyType.*STANDARD*,
	true);
	}
	if(property == null){
	throw new ConfigurationException("Fail to read Property");
	}
	}
	return property;

Class Name: ConfigurationConstants.java it is interface

Parameters:

- `public static final String PROPERTY_NODE_SEPARATOR = ";";`
- `public static final String PROPERTY_VALUE_SEPARATOR = ";";`
- `public static final String PROPERTY_NAME_BCBEAN = "BCBean";`

Class Name: ConfigurationException.java

Method Name	Method Body
`public ConfigurationException(String message)`	`super(message);`
`public ConfigurationException(String message, Exception e)`	`super(message, e);`

Class Name: ConfigurationProviderBean.java

Extends: AbstractConfigurationProvider.java

Implements: `IConfigurationReader.java, IConfigurationProvider.java`

Method Name	Method Body

public <BCInterface> BCInterface getConnectivity(Class<BCInterface> ifType)**throws** ConfigurationException	String serviceType = **this**.getServiceType();
	String node = ifType.getName() + serviceType;
	String jndiName = **null**;
	try{
	jndiName = **this**.getProperty(node, com.ibm.arch.configuration.ConfigurationConstants.*PROPERTY_NAME_BCBEAN*).getValue();
	}**catch** (ConfigurationException e) {
	// **TODO**: handle exception
	}
	BCInterface interface1 = **null**;
	try{
	Context ctx = **new** InitialContext();
	interface1 = ifType.cast(ctx.lookup(jndiName));
	}**catch** (NamingException e) {
	// **TODO**: handle exception
	}
	
	return interface1;

public String getServiceType() **throws** ConfigurationException	String node = "BC"; String name = "ServiceType"; **return this**.getProperty(node, name).getValue();

Class Name: ConfigurationServiceBean.java

Parameters:

```
1. @EJB
(beanName="com.ibm.arch.confcore.modeled.bonode.customproperty.customproperty.CustomProperty")

private
com.ibm.arch.confcore.modeled.bonode.customproperty.customproperty.CustomPropertyServiceLocal _dep_to_CustomProperty;
2. @EJB
(beanName="com.ibm.arch.confcore.modeled.bonode.standardproperty.standardproperty.StandardProperty")

private
com.ibm.arch.confcore.modeled.bonode.standardproperty.standardproperty.StandardPropertyServiceLocal _dep_to_StandardProperty;
```

Method Name	Method Body
protected com.ibm.arch.confcore.modeled. bonode.customproperty.customproperty. CustomPropertyServiceLocal getCustomPropertyService()	**return** _dep_to_CustomProperty;
protected com.ibm.arch.confcore. modeled.bonode.standardproperty. standardproperty.StandardPropertyServiceLocal getStandardPropertyService()	**return** _dep_to_StandardProperty;

protected Collection<Property> convertToPropertyCollection(Collection<?> xProps) **throws** ConfigurationException	Collection<Property> collection = **new** ArrayList<Property>();
	Iterator<?> iterator = xProps.iterator();
	while(iterator.hasNext()){
	Property property = **null**;
	Object object = iterator.next();
	if(object **instanceof** StandardProperty){
	property = **this**.convertToProperty((StandardProperty)object);

	}**else if** (object **instanceof** CustomProperty) {
	property = **this**.convertToProperty((CustomProperty)object);
	}**else**{
	throw new ConfigurationException("" + object);
	}
	collection.add(property);
	
	}
	
	return collection;

protected Property convertToProperty(StandardProperty standardProperty)	Property property = **new** Property();
	// Property Identifier
	
	PropertyIdentifier propertyIdentifier = **new** PropertyIdentifier();
	propertyIdentifier.setName(standardProperty.getName());
	propertyIdentifier.setNode(standardProperty.getNode());
	property.setPropertyIdentifier(propertyIdentifier);
	
	// For Other Attributes
	property.setValue(standardProperty.getValue());
	property.setDescription(standardProperty.getDescription());
	// For Language
	
	Set<String> setLanguage = standardProperty.getDescriptionLanguages();
	if(!setLanguage.isEmpty()){
	for (String string : setLanguage) {
	property.setDescription(standardProperty.getDescription(string));
	}
	}
	return property;

protected Property convertToProperty(CustomProperty customProperty)	Property property = **new** Property();
	// Property Identifier
	PropertyIdentifier propertyIdentifier = **new** PropertyIdentifier();
	propertyIdentifier.setName(customProperty.getName());
	propertyIdentifier.setNode(customProperty.getNode());
	property.setPropertyIdentifier(propertyIdentifier);
	// For Other Attributes
	property.setValue(customProperty.getValue());
	property.setDescription(customProperty.getDescription());
	// For Language
	Set<String> setLanguage = customProperty.getDescriptionLanguages();
	if(!setLanguage.isEmpty()){
	for (String string : setLanguage) {
	property.setDescription(customProperty.getDescription(string));
	}
	}
	return property;

public void createCustomProperty(Property property)
 	
throws ConfigurationException
	StandardProperty standardProperty = **null**;
	try {
	standardProperty =
	this.getStandardPropertyService().create(property.getPropertyIdentifier().getName(),
	property.getPropertyIdentifier().getNode(), property.getValue(), property.getDescription());
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block
	e.printStackTrace();
	}
	try {
	this.getStandardPropertyService().update(standardProperty);
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block
	e.printStackTrace();
	}

public void createStandardProperty(Property property)
throws ConfigurationException
	CustomProperty customProperty = **null**;
	try {
	customProperty =
	this.getCustomPropertyService().create(property.getPropertyIdentifier().getName(),
	property.getPropertyIdentifier().getNode(), property.getValue(), property.getDescription());
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block

	e.printStackTrace();
	}
	try {
	this.getCustomPropertyService().update(customProperty);
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block
	e.printStackTrace();
	}

public void
deleteCustomProperty(PropertyIdentifier identifier)
throws ConfigurationException
	CustomProperty customProperty = **null**;
	try {
	customProperty = **this**.getCustomPropertyService().readByCustomKeys(identifier.getName(), identifier.getNode());
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block
	e.printStackTrace();
	}
	try {
	this.getCustomPropertyService().delete(customProperty);
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block
	e.printStackTrace();
	}

public void
deleteStandardProperty(PropertyIdentifier identifier)
throws ConfigurationException
	StandardProperty standardProperty = **null**;
	try {
	standardProperty = **this**.getStandardPropertyService().readByCustomKeys(identifier.getName(), identifier.getNode());
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block
	e.printStackTrace();
	}
	try {
	this.getStandardPropertyService().delete(standardProperty);
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block
	e.printStackTrace();
	}

public Collection<Property>
findAllCustomProperties()
throws ConfigurationException
	Collection<CustomProperty> collection = **null**;
	try{
	collection = **this**.getCustomPropertyService().findAll();
	}**catch** (Exception e) {
	// **TODO**: handle exception
	throw new ConfigurationException("Failed to find all Custom Properties" + e);
	}
	return this.convertToPropertyCollection(collection);

public Collection<Property>
findAllStandardProperties()
throws ConfigurationException
	Collection<StandardProperty> collection = **null**;
	try {
	collection = **this**.getStandardPropertyService().findAll();
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block
	e.printStackTrace();
	}
	return this.convertToPropertyCollection(collection);

public Collection<Property>
findCustomProperties(
PropertyIdentifier identifier) **throws**
ConfigurationException
	QueryFilter queryFilterName = QueryFilterFactory.*createFilter*(identifier.getName());
	QueryFilter queryFilterNode = QueryFilterFactory.*createFilter*(identifier.getNode());
	Collection<CustomProperty> collection = **null**;
	try {
	collection = **this**.getCustomPropertyService().findByCustomKeys(queryFilterName, queryFilterNode);
	} **catch** (Exception e) {
	// **TODO** Auto-generated catch block
	e.printStackTrace();
	}
	return this.convertToPropertyCollection(collection);

public Collection<Property> findStandardProperties(PropertyIdentifier identifier) **throws** ConfigurationException	QueryFilter queryFilterName = QueryFilterFactory.*createFilter*(identifier.getName()); QueryFilter queryFilterNode = QueryFilterFactory.*createFilter*(identifier.getNode()); Collection<StandardProperty> collection = **null**; **try** {

	collection = this.getStandardPropertyService().findByCustomKeys(queryFilterName, queryFilterNode);
	} catch (Exception e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
	}
	return this.convertToPropertyCollection(collection);

public Property readCustomProperty(PropertyIdentifier identifier)
throws ConfigurationException
	CustomProperty customProperty = null;
	try {
	customProperty = this.getCustomPropertyService().readByCustomKeys(identifier.getName(), identifier.getNode());
	} catch (Exception e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
	}
	return this.convertToProperty(customProperty);

public Property readStandardProperty(PropertyIdentifier identifier)
throws ConfigurationException
	StandardProperty standardProperty = null;
	try {
	standardProperty = this.getStandardPropertyService().readByCustomKeys(identifier.getName(), identifier.getNode());
	} catch (Exception e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
	}
	return this.convertToProperty(standardProperty);

public void updateCustomProperty(Property property)
throws ConfigurationException
	CustomProperty customProperty = null;
	try {
	customProperty = this.getCustomPropertyService().readByCustomKeys(property.getPropertyIdentifier().getName(), property.getPropertyIdentifier().getNode());
	} catch (Exception e) {
	// TODO Auto-generated catch block
	e.printStackTrace();
	}
	try
	{
	this.getCustomPropertyService().update(customProperty);
	}catch (Exception e) {
	// TODO: handle exception
	}

public void updateStandardProperty(Property property)
throws ConfigurationException
	StandardProperty standardProperty = null;
	try {
	standardProperty = this.getStandardPropertyService().readByCustomKeys(property.getPropertyIdentifier().getName(), property.getPropertyIdentifier().getNode());
	} catch (Exception e1) {
	// TODO Auto-generated catch block
	e1.printStackTrace();
	}
	try{
	this.getStandardPropertyService().update(standardProperty);
	}catch (Exception e) {
	// TODO: handle exception
	}
	}

Interface Name: IConfigurationAdministration.java

Annotation	Method Signature
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	public abstract void updateStandardProperty(Property property) throws ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	public abstract void updateCustomProperty(Property property) throws ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	public abstract Property readStandardProperty(PropertyIdentifier identifier) throws ConfigurationException;

@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	**public abstract** Property readCustomProperty(PropertyIdentifier identifier) **throws** ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	**public abstract** java.util.Collection<Property> findStandardProperties(PropertyIdentifier identifier) **throws** ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	**public abstract** java.util.Collection<Property> findCustomProperties(PropertyIdentifier identifier) **throws** ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	**public abstract** java.util.Collection<Property> findAllStandardProperties() **throws** ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	**public abstract** java.util.Collection<Property> findAllCustomProperties() **throws** ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	**public abstract void** deleteStandardProperty(PropertyIdentifier identifier) **throws** ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	**public abstract void** deleteCustomProperty(PropertyIdentifier identifier) **throws** ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	**public abstract void** createStandardProperty(Property property) **throws** ConfigurationException;
@javax.ejb.TransactionAttribute (javax.ejb.TransactionAttributeType.*REQUIRED*)	**public abstract void** createCustomProperty(Property property) **throws** ConfigurationException;

Interface Name: IConfigurationProvider.java

Method Name	Method Signature
getConnectivity	**public** <BCInterface> BCInterface getConnectivity(Class<BCInterface> ifType) **throws** ConfigurationException;
getServiceType	**public** String getServiceType() **throws** ConfigurationException;

Interface Name: IConfigurationReader.java

Method Name	Method Signature
getProperty	**public** Property getProperty(String node, String name) **throws** ConfigurationException;
getProperty	**public** Property getProperty(String node, String name, **boolean** fallback) **throws** ConfigurationException;
getProperty	**public** Property getProperty(String node, String name, PrevailRule prevailRule) **throws** ConfigurationException;
getProperty	**public** Property getProperty(String node, String name, PrevailRule prevailRule, **boolean** fallback) **throws** ConfigurationException;

Class Name: PrevailRule.java Type: enum

Implements: Serializable

Enum: CUSTOM, STANDARD

Class Name:Property.java

Parameters:

```
private PropertyIdentifier propertyIdentifier;
```

- `private String value;`
- `private String description;`

Method Name	Method Body
public PropertyIdentifier getPropertyIdentifier()	return propertyIdentifier;
public void setPropertyIdentifier(PropertyIdentifier propertyIdentifier)	this.propertyIdentifier = propertyIdentifier;
public String getValue()	return value;
public void setValue(String value)	this.value = value;
public String getDescription()	return description;
public void setDescription(String description)	this.description = description;

Class Name: PropertyIdentifier.java

Parameters:

- private String name;
- private String node;

Method Name	Method Body
public String getName()	return name;
public void setName(String name)	this.name = name;
public String getNode()	return node;
public void setNode(String node)	this.node = node;

Class Name: PropertyType.java Type: enum

Implements: Serializable

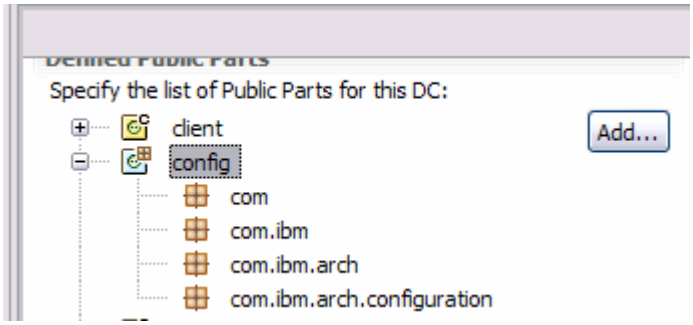
Enum: CUSTOM, STANDARD

Please create public parts of arch/config ejb project.

Purpose: Assembly

Public Part Name: config

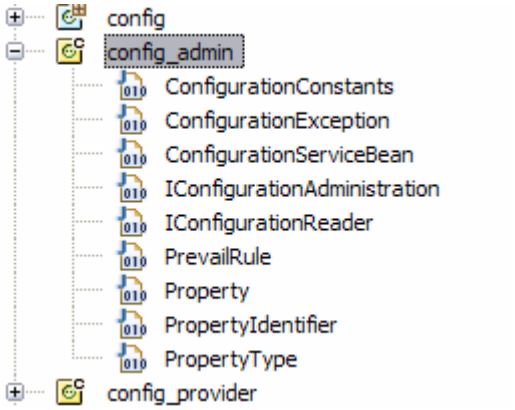
Please do the same and pass only class files as below:



Purpose: Compilation

Public Part Name: config_admin

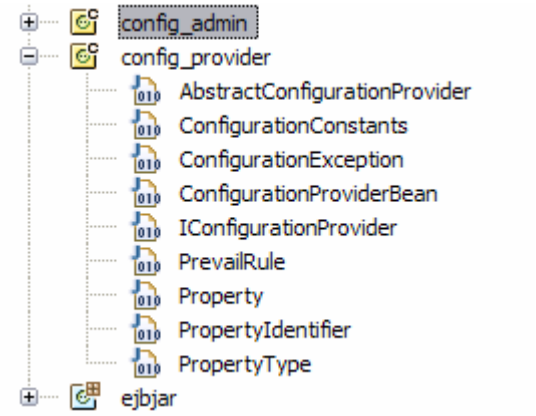
Please do the same and pass only class files as below:



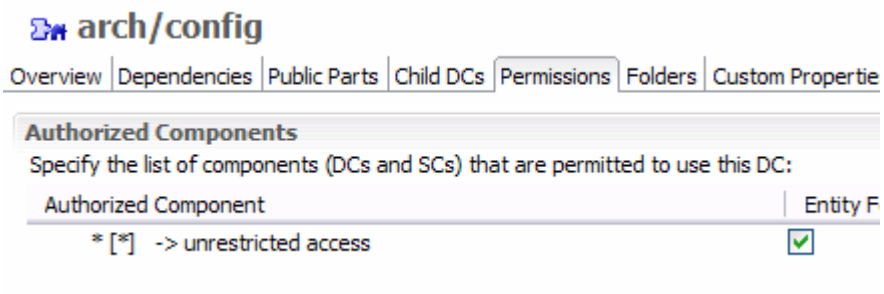
Purpose: Compilation

Public Part Name: config_provider

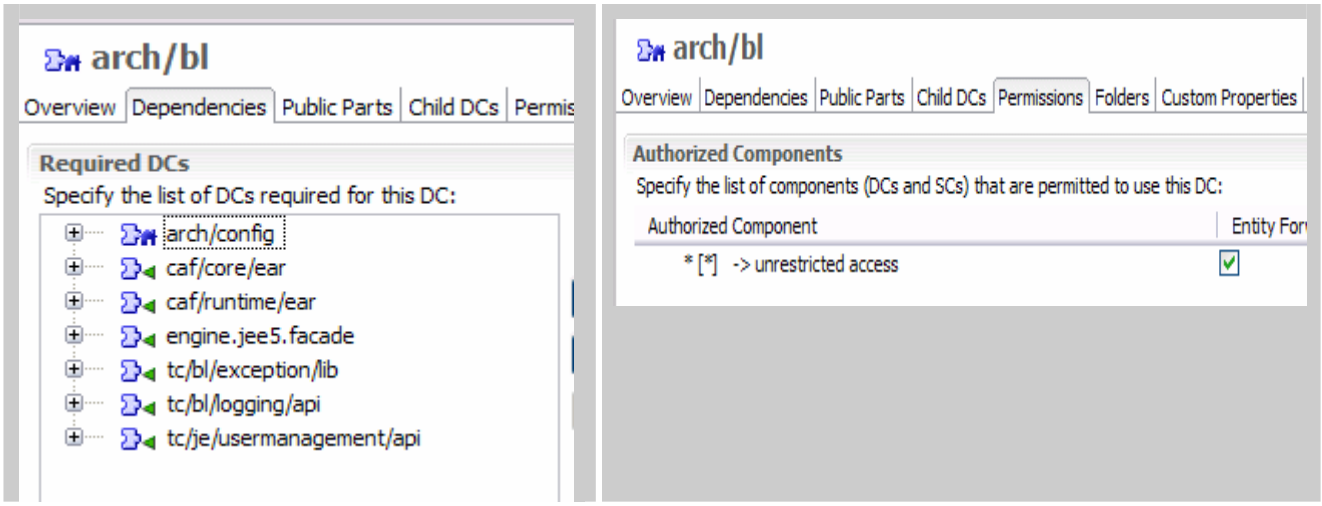
Please do the same and pass only class files as below:



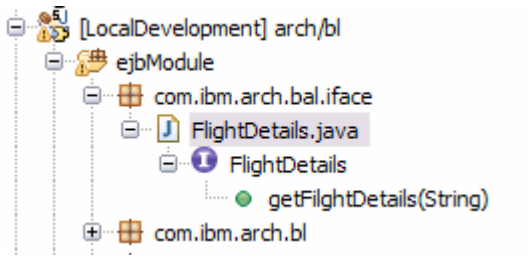
Please do the same as below for arch/config ejb project.



Before going to create any java file into arch/bl project we need to configure few things, please do the same as shown below picture:



Please create the following package and interface (FlightDetails.java) and method name as shown.

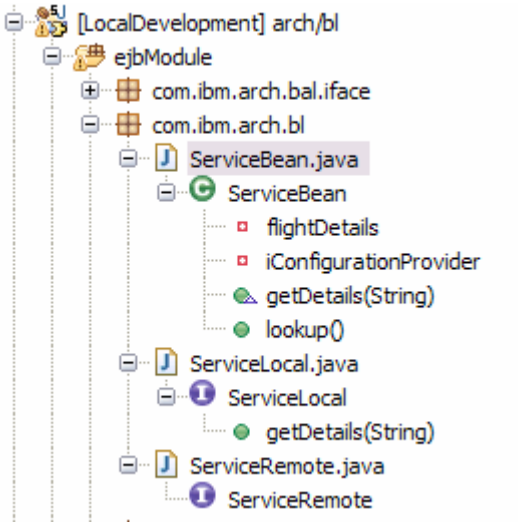


The following package: com.ibm.arch.bal.iface is BAL layer of this application.

Interface: FlightDetails.java

Method Name	Method Signature
`getFilghtDetails`	`String getFilghtDetails(String string)`

Please create the service bean as shown in the picture.



Bean Name: ServiceBean.java

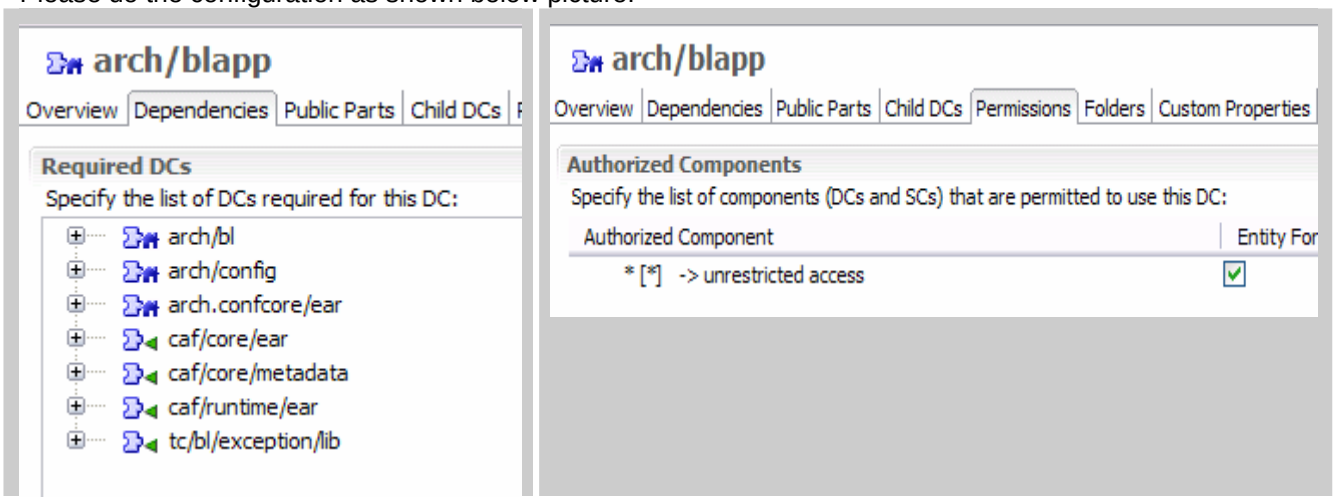
Annotation	Parameters
@EJB	**private** IConfigurationProvider iConfigurationProvider;
	private FlightDetails flightDetails;

Annotation	Method Name	Method Body
@PostConstruct	**public void** lookup() **throws** Exception	flightDetails = iConfigurationProvider. getConnectivity (com.ibm.arch.bal.iface.FlightDetails.**class**);
	public String getDetails(String string)	String string2 = flightDetails.getFilghtDetails(string); return string2;

Interface Name: ServiceLocal.java

Method Name	Method Signature
getDetails	String getDetails(String string);

Please do the configuration as shown below picture.



Backend Connectivity

In this layer I am going to consume ES/WS into ejb's and implement the interfaces into BAL layer.

The WS/ES they are point to ERP2005s system or CRM system etc.

Before we are going to consume wsdl of WS/ES we have to configure thw wsdl endpoint.

Configure wsdl via SOA Management under nwa.

After logon to nwa please go to the tab SOA Management -> Webservice Template

Then choose service type: wsdl

URL: Provide wsdl url

End Point name: as you choose

Http Authentication Type: User Id/Password

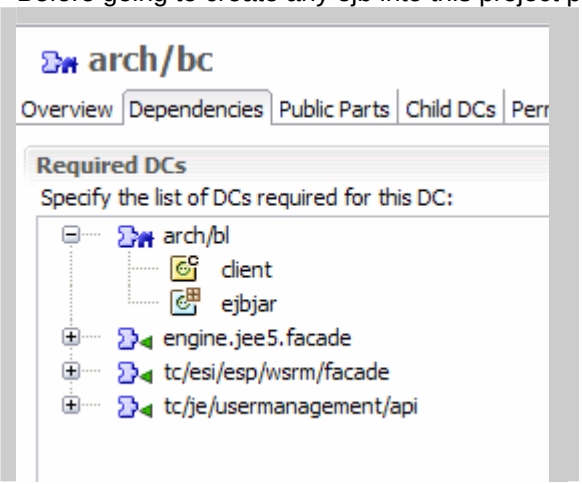
Please press button Details and provide user id/password.

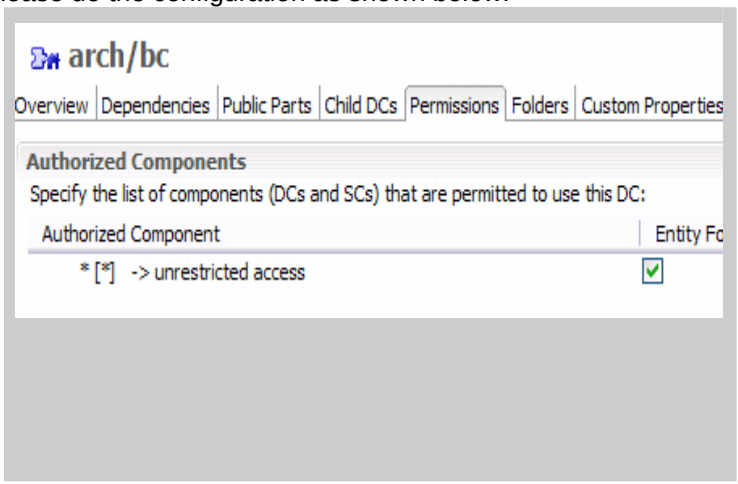
Now our wsdl for WS/ES is configure to consume.

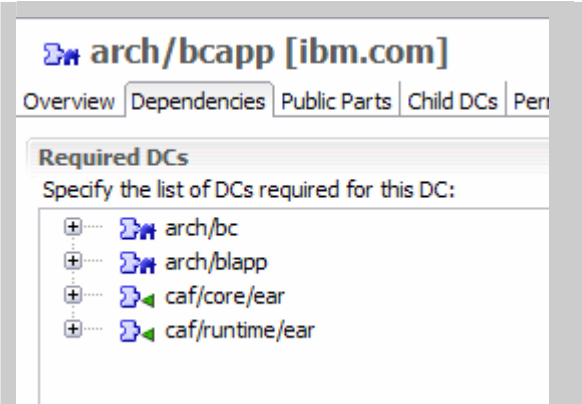
In the bc layer I have create two ejb's for a particular interface in the bal layer, because one ejb for ws and one ejb for es implementation, so we can switch between them without touching our codes just we have to change switch in caf service browser from ES to WS or vice versa.

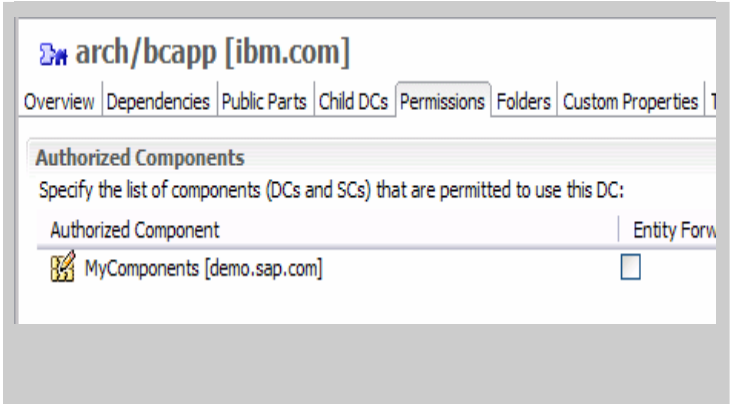
Please create an arch/bc ejb project with arch/bcapp ear project.

Before going to create any ejb into this project please do the configuration as shown below.



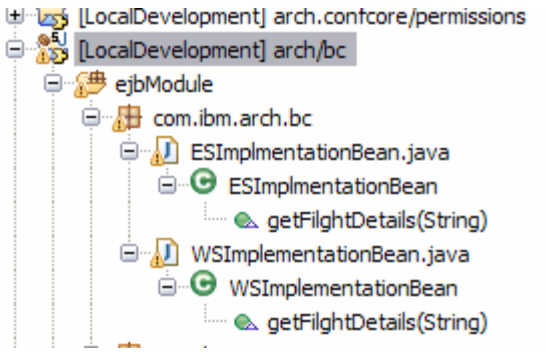






Now please create two ejb's which are implement the same interface of BAL layer.

Please do the same as shown below:



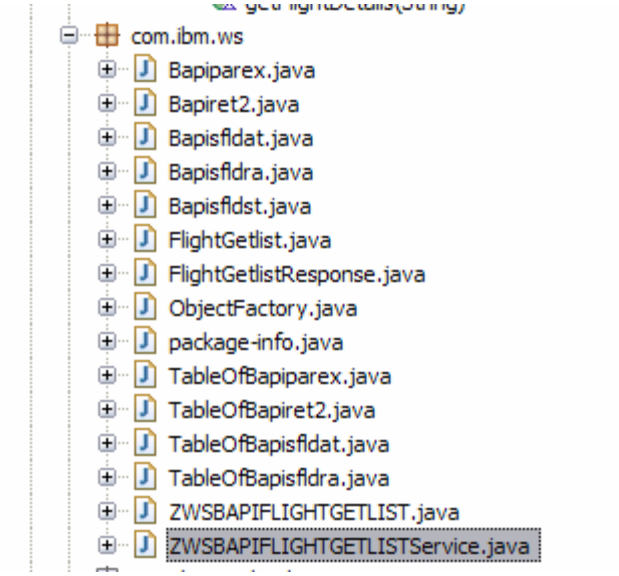
Please follow the steps to call a wsdl into our application.

- Right click on the desired package in to bc ejb project.
- Choose import.
- Choose Web service-> wsdl
- Remote Location/ File System
- Provide url of wsdl and user id/password
- Package name and press finish button.

After import the wsdl to create client i.e java proxy classes

Please do the steps

- Please right click on the imported wsdl.
- Choose Web service - > Generate Client
- Java proxy classes will be created under your desired package.



Using this java classes I am going to implement the two ejb because es was not available at the development time due to this reason I am using the ws for es and ws implementation.

Class Name: ESImplmentationBean.java

Implements: FlightDetails.java

Parameter:

```
@WebServiceRef(name = "ZWSBAPIFLIGHTGETLISTService")
private ZWSBAPIFLIGHTGETLISTService service;
```

Method Name	Method Body

public String getFilghtDetails(String string)	ZWSBAPIFLIGHTGETLIST port = service.getZWS_BAPI_FLIGHT_GETLISTSoapBinding();
	javax.xml.ws.BindingProvider bp = (BindingProvider)port;
	bp.getRequestContext().put(BindingProvider.*USERNAME_PROPERTY*, "bray");
	bp.getRequestContext().put(BindingProvider.*PASSWORD_PROPERTY*, "ibm");
	java.lang.String airline = string;
	javax.xml.ws.Holder<com.ibm.ws.TableOfBapisfldra> dateRange = **new** Holder<com.ibm.ws.TableOfBapisfldra>();
	dateRange.value = **new** TableOfBapisfldra();
	com.ibm.ws.Bapisfldst destinationFrom = **new** Bapisfldst();
	com.ibm.ws.Bapisfldst destinationTo = **new** Bapisfldst();
	javax.xml.ws.Holder<com.ibm.ws.TableOfBapiparex> extensionIn = **new** Holder<com.ibm.ws.TableOfBapiparex>();
	extensionIn.value = **new** TableOfBapiparex();
	javax.xml.ws.Holder<com.ibm.ws.TableOfBapiparex> extensionOut = **new** Holder<com.ibm.ws.TableOfBapiparex>();
	extensionOut.value = **new** TableOfBapiparex();
	javax.xml.ws.Holder<com.ibm.ws.TableOfBapisfldat> flightList = **new** Holder<com.ibm.ws.TableOfBapisfldat>();
	flightList.value = **new** TableOfBapisfldat();
	java.lang.Integer maxRows = **new** Integer(10);
	javax.xml.ws.Holder<com.ibm.ws.TableOfBapiret2> returnTemp = **new** Holder<com.ibm.ws.TableOfBapiret2>();
	returnTemp.value = **new** TableOfBapiret2();
	port.flightGetlist(airline, dateRange, destinationFrom, destinationTo, extensionIn, extensionOut, flightList, maxRows, returnTemp);
	String string2 = flightList.value.getItem().get(0).getCityfrom() + "Calling from ES";
	
	return string2;

Class Name: WSImplementationBean.java

Implements: FlightDetails.java

Parameter:

```
@WebServiceRef(name = "ZWSBAPIFLIGHTGETLISTService")
private ZWSBAPIFLIGHTGETLISTService service;
```

Method Name	Method Body

public String getFilghtDetails(String string)	ZWSBAPIFLIGHTGETLIST port = service.getZWS_BAPI_FLIGHT_GETLISTSoapBinding();
	javax.xml.ws.BindingProvider bp = (BindingProvider)port;
	bp.getRequestContext().put(BindingProvider.*USERNAME_PROPERTY*, "bray");
	bp.getRequestContext().put(BindingProvider.*PASSWORD_PROPERTY*, "ibm");
	java.lang.String airline = string;
	javax.xml.ws.Holder<com.ibm.ws.TableOfBapisfldra> dateRange = **new** Holder<com.ibm.ws.TableOfBapisfldra>();
	dateRange.value = **new** TableOfBapisfldra();

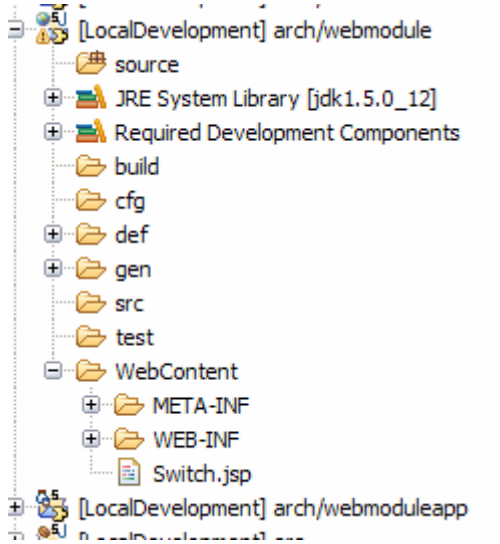
```
com.ibm.ws.Bapisfldst destinationFrom = new Bapisfldst();
com.ibm.ws.Bapisfldst destinationTo = new Bapisfldst();
javax.xml.ws.Holder<com.ibm.ws.TableOfBapiparex> extensionIn =
new Holder<com.ibm.ws.TableOfBapiparex>();
extensionIn.value = new TableOfBapiparex();
javax.xml.ws.Holder<com.ibm.ws.TableOfBapiparex> extensionOut =
new Holder<com.ibm.ws.TableOfBapiparex>();
extensionOut.value = new TableOfBapiparex();
javax.xml.ws.Holder<com.ibm.ws.TableOfBapisfldat> flightList = new
Holder<com.ibm.ws.TableOfBapisfldat>();
flightList.value = new TableOfBapisfldat();
java.lang.Integer maxRows = new Integer(10);
javax.xml.ws.Holder<com.ibm.ws.TableOfBapiret2> returnTemp = new
Holder<com.ibm.ws.TableOfBapiret2>();
returnTemp.value = new TableOfBapiret2();
port.flightGetlist(airline, dateRange, destinationFrom, destinationTo,
extensionIn, extensionOut, flightList, maxRows, returnTemp);
String string2 = flightList.value.getItem().get(0).getCityfrom() + "Calling
from WS";

return string2;
```

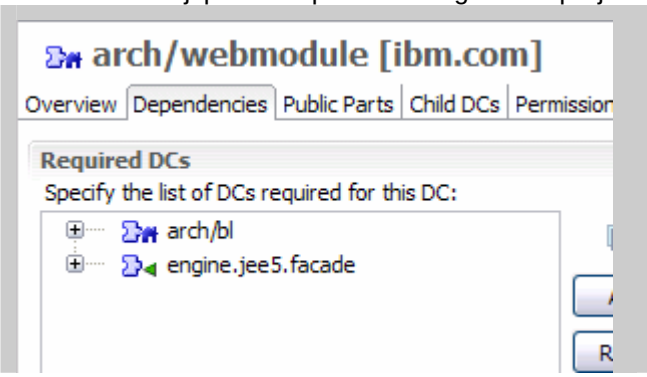
UI Layer

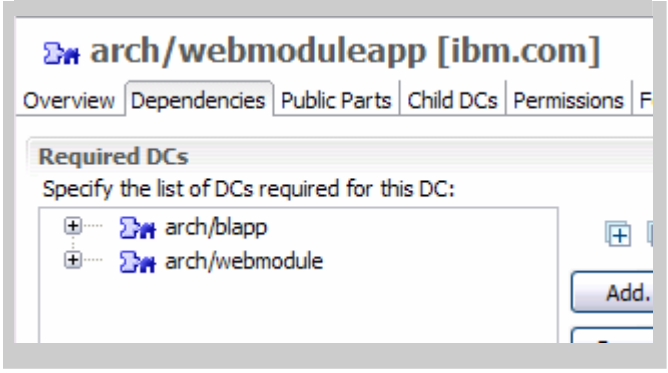
To get the out put of the application we have to cerate a ui application due to this reason I am going to create web application.

Please create a web module project name: arch/webmodule with an ear project name: arch/webmodule



Before create jsp/servelt please configure the project as shown below.





Now create a jsp page with name Switch.jsp and write the code as shown below:

```
Context context = new InitialContext();
ServiceLocal local = (ServiceLocal)context.lookup("ibm.com/arch~blapp/LOCAL/ServiceBean/com.ibm.arch.bl.ServiceLocal");

out.println("ABC - " + local.getDetails("AA"));
```

Out put for WS: “ABC -” + City + Calling from WS

Out put for ES: “ABC -” + City + Calling from ES

Note

WS: Web service
ES: enterprise Service
BC: Backend Connectivity
BL: Business Logic
BAL: Backend Abstraction Layer

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.