Advanced Service Composition with SAP Composite Application Framework Capabilities in SAP NetWeaver CE 7.1

Applies to:
SAP NetWeaver Composition Environment 7.1, SR3 – also the SDN downloadable SAP NetWeaver CE 7.1 trial version can be used for working this exercise.

Summary
This tutorial summarizes some features of SAP NetWeaver CE 7.1’s composite application framework (CAF). You may learn how to create a CAF project and to model business objects. The document explains then handling of dependencies between development components and the use and purpose of CAF application services. Get to know the integration of service with the help of the service registry and see how fast and easy own business logic can be implemented in the CAF development. Last but not least this paper discusses the re-use of application service as Web Services.

Authors: Roland Münster, Abdul Razack, Bogdan Vatkov
Company: SAP AG
Created on: Oct 2007

Author Bios
Roland Münster
is a member of the Regional Implementation Group (RIG) EMEA team of SAP NetWeaver. He is responsible for Java, Java EE, and related topics. Furthermore, he is a contact for technical questions in the area of SAP NW CE. Prior to joining the SAP NW RIG team, Roland was technical consultant and developer in different SAP areas for seven years. He also was project lead for SAP's Java training curriculum.

Abdul Razack
is a member of SAP NetWeaver Regional Implementation Group US responsible for implementation activities, knowledge transfer, and feedback to development. His career spans 14 years in various areas of Information Technology including product development, integration, solution definition, solution architect, and pre-sales activities.

Bogdan Vatkov
is a development architect with the SAP NetWeaver Composition Tools. He is focused in the areas of Model-Driven Architecture, Service-Oriented Architecture, Service Adaptation and Composition. He joined SAP as a software developer in the SAP NetWeaver Composite Application Framework area in 2004.
Table of Contents

Introduction to this exercise ................................................................................................................................... 3
Prerequisites ........................................................................................................................................................... 4
Exercise 1: Create Composite Application ....................................................................................................... 5
  Creating the Composite Application Project ...................................................................................................... 5
Exercise 2: Model Business Object Nodes ......................................................................................................... 10
  Building Business Object “Team” ..................................................................................................................... 10
Exercise 3: Define Dependencies ......................................................................................................................... 19
  Adding Dependency to Composite Application ............................................................................................... 19
  Adding Dependency to Enterprise Application ............................................................................................... 21
Exercise 4: Model Application Service ............................................................................................................ 26
Exercise 5: Import Enterprise Service from Service Registry ............................................................................. 34
Exercise 6: Add Business Logic .......................................................................................................................... 39
Exercise 7: Expose Application Service as Web Service .................................................................................... 48
Exercise 8: Run the Application .......................................................................................................................... 50
Appendix ............................................................................................................................................................. 56
  Setting up the Environment .............................................................................................................................. 56
  Working with Remote Projects .......................................................................................................................... 57
Copyright ............................................................................................................................................................. 79
Introduction to this exercise

The exercise will build a Project Manager Service as a part of a business object. This service will be exposed as a web service. Furthermore it’s discussed how external coding can be integrated into your application with the help of dependencies.

Expose as WebService

- Application Service
- Business Object
- Utility classes
- Override Operations
- Add dependencies
- Use external libraries
- Use EJB modules
- Composite Application

This tutorial is parted in several exercises. You should work them in the given sequence cause they are built on each other.

- Exercise 1 shows the creation of a composite application project within the SAP NetWeaver Developer Studio (SAP NWDS)
- Exercise 2 explains the modeling of business object nodes, see also recording Design Business Objects in CAF (14 min).
- This part (exercise 3) explains how you can model dependencies between different CAF projects, so different possibilities of re-use are given. (Here a CAF model is re-used that is created during the exercise “CE_Project_Example.pdf”. This exercise can be downloaded from SDN CE tutorial page).
- This exercise 4 shows the design of a CAF application service.
- This chapter (exercise 5) explains how the import of enterprise services from Service Registry is done with the help of SAP NetWeaver Developer Studio. (Prerequisite is a configured connection from your locally installed SAP NetWeaver NWDS to SAP’s ES Workplace (in SDN: http://www.sdn.sap.com/irj/sdn/esworkplace).
- Exercise 6 depicts the insertion of own business logic into CAF projects.
- It’s possible to turn a CAF application service into a Web Service with one mouse click, exercise 7 features this valuable possibility.
- Finally exercise 8 run the prepared application. Refer also recording Configure, test and run the scenario (3 min).
Prerequisites

You need to have configured your local SAP NetWeaver Developer Studio to be able to build a connection to SAP’s ES Workplace. The following screenshot shows the needed configurations:

1. In SAP NetWeaver Developer Studio:

2. Window > Preferences: Web Services > SAP Service Registry

3. Enter the connection information for the SAP UDDI Service which houses all Enterprise Service Definitions.

Additionally exercise 3 will build up a dependency to another composite application. This composite is the outcome of the tutorial “CE_Project_Example” (that can be gotten in SAP’s SDN). The following picture shows the relationships between these two projects.

You can get the solution / source code here: https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/802f6694-5ee0-2a10-81a9-8edd4b0fe620.
Exercise 1: Create Composite Application

In this exercise we will build the Composite Application ce350ex.

Creating the Composite Application Project

1. Launch the developer studio from the shortcut on your desktop or local folder (here: c:\sessioncontent\CE350).

2. Create a new project by choosing File->New->Project from the menu bar.
3. Choose Development Infrastructure/Development Component for the type of project and then click Next.
4. In the next screen choose *Composite Application* as type for Development Component.

   Click *Next*. 
5. The development component should be defined in the Local Development branch since we are not using NWDI. Expand the Local Development node and choose My Components.

   Click Next.

6. Enter an appropriate name for the component (here: “ce350ex”), click Next and click Finish.
7. Click Finish
Exercise 2: Model Business Object Nodes

In this exercise we will build the Business Object Nodes (or Entities) **Team** and **Member** which will have a relationship of type **Composition** (composition means that once the data for given Team gets deleted from its database table the Member data for all belonging members will be deleted as well). The Business Object Nodes are built using the SAP NetWeaver Developer Studio and the Composite Application Service perspective.

Refer to recording also recording [Design Business Objects in CAF](http://bit.ly/59L8K) (14 min).

**Building Business Object “Team”**

1. In the Composite Application Explorer view, right click on node *modeled* and choose *New Business Object* from the context menu.

![Composite Application Explorer](image)
2. Enter the name of the Business Object: “Team”

3. In the radio button group leave the default selection - Create new structure and click Finish. New Business Object with a root Business Object Node will be created and an editor will be opened for the node.

4. In the “Team” Business Object Node editor, navigate to tab Structure (the tabs are located in the bottom part of the editor) and press button Edit Main Structure to add some additional data fields to the Business Object Node.
5. In the column Existing Types, expand the type tree caf.core/primitive.

Add the following attributes by selecting the STRING type and then clicking the Add attribute icon in the middle column:

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>STRING</td>
</tr>
<tr>
<td>description</td>
<td>STRING</td>
</tr>
<tr>
<td>abbreviation</td>
<td>STRING</td>
</tr>
<tr>
<td>departmentNumber</td>
<td>STRING</td>
</tr>
</tbody>
</table>

6. In the column Existing Types, expand the type tree caf.core/services.

Add the following attribute by selecting the Id type and then clicking the Add attribute icon in the middle column:

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>projectId</td>
<td>Id</td>
</tr>
</tbody>
</table>

In the right column change the names of the attributes as shown on the picture.

7. Open the Properties view if it is not opened. Navigate to Window -> Show View... -> Properties. In the Structure Fields column of the Team structure editor, select the attribute description.
8. In the Properties view navigate to Relations / AttributeAspects / DDICAttribute / Length and change the value from 0 to 300.

9. In the Permissions tab of the Team Business Object Node editor, disable the Permissions checks enabled checkbox.
10. In the Composite Application Explorer view, right click on Business Object Node Team and chose New BO Node from the context menu.

11. Enter the name of the Business Object Node: “Member”

   In the radio button group leave the default selection - Create new structure and click Finish.
   New Business Object with a root Business Object Node will be created and an editor will be opened for the node.
12. In the Composite Application Explorer view, right click on node Data Types and chose New Simple Type from the context menu.

13. In the Simple type wizard, enter “Role” as Simple type name and STRING as Base type. Press Finish button.

14. Navigate to the Aspects Tab. As we would like that the Role type is an enumeration of values, enter the following XSD Facets, by selecting “enumeration” from Available Facets drop down and pressing Add XSD Faced button for each of the facets you want to add. In
the table below you can see the values of the enumeration.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEVELOPER</td>
<td></td>
</tr>
<tr>
<td>TEAMLEAD</td>
<td></td>
</tr>
<tr>
<td>QA</td>
<td></td>
</tr>
<tr>
<td>ARCHITECT</td>
<td></td>
</tr>
</tbody>
</table>

The resultant simple type should be as shown in the diagram. Make sure TEAMLEAD is one word do not include spaces in between.

15. In the “Member” Business Object Node editor, navigate to tab Structure and press button Edit Main Structure to add some additional data fields to the Business Object Node.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Cardinality</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>key</td>
<td>Id</td>
<td>1..1</td>
<td>false</td>
</tr>
<tr>
<td>createdBy</td>
<td>UserId</td>
<td>0..1</td>
<td>false</td>
</tr>
<tr>
<td>createdAt</td>
<td>DATETIME</td>
<td>0..1</td>
<td>false</td>
</tr>
<tr>
<td>modifiedAt</td>
<td>DATETIME</td>
<td>0..1</td>
<td>false</td>
</tr>
<tr>
<td>modifiedBy</td>
<td>UserId</td>
<td>0..1</td>
<td>false</td>
</tr>
</tbody>
</table>
16. In the column *Existing Types*, expand the type tree `caf.core/primitive`.

Add the following attributes by selecting the STRING type and then clicking the *Add attribute* icon in the middle column:

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>STRING</td>
</tr>
<tr>
<td>employeeId</td>
<td>STRING</td>
</tr>
</tbody>
</table>

In the right column change the names of the attributes as shown on the picture.

17. In the column *Existing Types*, expand the type tree `ce350ex/modeled`.

Add the following attribute by selecting the `Role` type and then clicking the *Add attribute* icon in the middle column:

<table>
<thead>
<tr>
<th>Fieldname</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>Role</td>
</tr>
</tbody>
</table>

18. In the *Permissions* tab of the *Member Business Object Node* editor, disable the *Permissions checks enabled* checkbox.

19. In the *Operations* tab of the *Member Business Object Node* editor click *Add* button to create a new find operation.
20. In the **New Operation** wizard write the name of the new operation as shown on the picture and select `employeeId` as its input parameter. Click Finish.

21. In the **Associations** tab of the **Team Business Object Node** editor rename the association to **Member node to “Members”**.

22. Then change the cardinality of the association to **“NONE_TO_MANY”**.
Exercise 3: Define Dependencies

In this exercise you will learn about different types of reuse for a composite application. Generally the following types of reuse are demonstrated:

- reuse of another composite application by defining dependency to it;
- reuse of EJBs that are defined in enterprise archive module, by defining dependency to the module;
- reuse a library which is packaged in .jar file and added to enterprise archive module, by defining dependency to the module.

Adding Dependency to Composite Application

1. In the Composite Application Explorer view, right click on node [LocalDevelopment] ce350ex and choose Edit from the context menu.
2. In the Project Editor, navigate to tab Reused Projects and press the Add button.

4. Press Yes.

Adding Dependency to Enterprise Application

5. In the Package Explorer view, right click on node [LocalDevelopment] ce350ex/ear and choose Development Component -> Show in Component Properties from the context menu.

6. In the Component Properties view, in the Dependencies tab, press Add button.
7. In the *Adding Dependencies* dialog, select node `LocalDevelopment/MyComponents/lib/ear` and press *Next* button.

8. In the *Dependency Details* pane, enable *Deploy Time* and press *Finish*. 
10. In the Adding Dependencies dialog, select node LocalDevelopment/MyComponents/lib/ear and press Next button.
11. In the **Dependency Details** pane, enable **Design Time** and press **Finish**.
Exercise 4: Model Application Service

In this exercise it will be shown how you can create an Application Service. Application Services implement the business logic of a CAF application and expose the functionality of the application to other systems or applications. An Application Service can be used in two directions:

- On the one side an AS acts as a façade for a given Business Object providing several business methods to add business logic and manipulate the underlying BO's.
- On the other side you can use Application Services for service composition. In that case the Application Service consumes one or more predefined (Web) Service (Service Orchestration), enhances their functionality by adding additional code, and exposes this new functionality externally as a new operation.

1. In the Composite Application Explorer view, right click on package modeled and choose New Application Service from the context menu.
2. Enter a name for the new service as shown on the figure. Click Finish. New application service will be created and an editor will be opened for it.

3. In the Operations tab of the ProjectManagerService application service editor click Add button to create a new operation.
4. Enter the name of the new operation as shown on the figure. Press Finish.

5. You should add input, output and fault parameters to the newly created operation. You can do this in the Operation Parameters section of the editor.

In the left part of the section, expand the type tree `caf.core/primitive`.

Add the following parameter by selecting the STRING type and then clicking the Add to Input button in the middle column:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>teamName</td>
<td>STRING</td>
</tr>
</tbody>
</table>

In the left part of the section, expand the type tree `caf.core/services`.

Add the following parameter by selecting the Id type and then clicking the Add to Input button in the middle column:

<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Id</td>
</tr>
</tbody>
</table>
Add the following parameter by selecting the Id type and then clicking the **Add to Output** button in the middle column:

<table>
<thead>
<tr>
<th>Parametername</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>projectId</td>
<td>Id</td>
</tr>
</tbody>
</table>

In the left part of the section, expand the type tree `caf.core/faults`

Add a fault by selecting the `CAFServiceException` type and then clicking the **Add to Fault** button in the middle column.

6. Click **Add** button to create a new operation
7. Enter the name of the new operation as shown on the figure. Press Finish.

8. You should add input, output and fault parameters to the newly created operation. You can do this in the Operation Parameters section of the editor.

   In the left part of the section, expand the type tree caf.core/primitive.

   Add the following parameter by selecting the STRING type and then clicking the Add to Input button in the middle column:

<table>
<thead>
<tr>
<th>Parametername</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>emloyeeId</td>
<td>STRING</td>
</tr>
</tbody>
</table>

   In the left part of the section, expand the type tree caf.core/services

   Add the following parameter by selecting the Id type and then clicking the Add to Input button in the middle column:

<table>
<thead>
<tr>
<th>Parametername</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
teamId     Id

Add the following parameter by selecting the Id type and then clicking the Add to Output button in the middle column:

<table>
<thead>
<tr>
<th>Parametername</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>memberId</td>
<td>Id</td>
</tr>
</tbody>
</table>

In the left part of the section, expand the type tree ce350ex/modeled/data types/simple/role

Add the following parameter by selecting the Role type and then clicking the Add to Input button in the middle column:

<table>
<thead>
<tr>
<th>Parametername</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>Role</td>
</tr>
</tbody>
</table>

In the left part of the section, expand the type tree caf.core/faults

Add a fault by selecting the CAFServiceException type and then clicking the Add to Fault button in the middle column.
9. Go to the Dependency tab of the ProjectManagerService application service editor. In the Existing Objects section navigate to the Team business object node as shown on the figure and click the Add Dependency icon.

10. In the Existing Objects section navigate to the Member business object node as shown on the figure and click the Add Dependency icon.

11. In the Existing Objects section navigate to the Project business object node as shown on the figure and click the Add Dependency icon.

12. Save the application.
13. Generate the application.
Exercise 5: Import Enterprise Service from Service Registry

In this exercise you will learn how CAF application can reuse an Enterprise Service. This Service will be used to retrieve structure containing naming details of an employee given its employee Id.

1. In the Composite Application Explorer view, right click on node external and chose Import Web Service from the context menu.

2. Chose Service Registry and press Next.
3. In the search criteria input field, enter “EmployeeBasic*” and press Next.
4. Select the service definition that was found in the Found Service Definitions section of the wizard page.
5. Select the endpoint that appeared in the Available Endpoints section and press Finish.

6. Enter user name and password to connect to the registry.(demo/welcome). If prompted again, use the same user id and password.

7. Start the default mapping creation wizard for the newly imported external service by right-clicking on the service and selecting Default mappings menu.
8. From the input parameters select the only required one – the `simpleContent` as shown on the figure.

9. From the output parameters select the `Name` structure that contains naming details of the employee.
10. In the **Target Application Service** section of the wizard select **Create methods in existing Application Service** radio button and then select **ProjectManagerService** in the area below. Press OK.

11. Go the Datasource tab and confirm that the mappings have been created.
Exercise 6: Add Business Logic

In this exercise you will learn how to use all the model elements you created up to now in your code. You will see how you can create business object node instances, how to invoke the external web service, invoke methods of an EJB, access a business object node from dependent CAF application and use the classes from the library project that you declared dependency to.

In order to help yourself with the manual coding you can use the files that are provided in your local workspace.

12. Copy the *Utils.java* file from the `c:\sessioncontent\CE350\snippets` folder and paste it in the project.
13. As the file is currently in the default package of the project you should move it to the package com.sap.demo.ce350ex. To do this, right-click on the file and select Refactor -> Move ...
14. Select the package you want to move to as shown on the figure and press OK.

Hint: If prompted to save click ok.

15. In the Implementation tab of the ProjectManagerService editor click on the link to the implementation file of the service.

16. Select the content inside the ProjectManagerServiceBeanImpl class as shown on the figure.
17. Paste the following code:

```
@EJB(beanName="DevDepartmentUtils")
private DevelopmentDepartmentUtilsLocal devDepartmentUtils;

@com.sap.caf.dt.CAFOperation(name = "createTeam")
public java.lang.String createTeam(@javax.jws.WebParam(name="teamName")
        java.lang.String teamName, @javax.jws.WebParam(name="projectId")
        java.lang.String projectId) throws com.sap.caf.rt.exception.CAFServiceException {
    // Check if there is a project with such id
    if (!existsProject(projectId)) {
        throw new CAFServiceException("Project with ID "+projectId+" does not exist.");
    }

    TeamServiceLocal teamService = this.getTeamService();
    Team team = teamService.create();
    // set details
    team.setName(teamName);
    team.setProjectId(projectId);
    // reuse other Enterprise Application
    team.setDepartmentNumber(devDepartmentUtils.createNewDepartmentEntityNumber());
    // use lib code to set the abbreviation
    DoubleMetaphone doubleMetaphone = new DoubleMetaphone();
    team.setAbbreviation(doubleMetaphone.doubleMetaphone(teamName, true));
    // store the team instance into the local database
    teamService.update(team);
    // return the Id of the newly created team
    return team.getKey();
}

@com.sap.caf.dt.CAFOperation(name = "createMember")
public java.lang.String createMember(@javax.jws.WebParam(name="teamId")
        java.lang.String teamId, @javax.jws.WebParam(name="role")
        java.lang.String role, @javax.jws.WebParam(name="employeeId")
        java.lang.String employeeId) throws com.sap.caf.rt.exception.CAFServiceException {
    if (!canCreateMemberForEmployeeId(employeeId)) {
        throw new CAFServiceException("This employee has already being assined to a project.");
    }

    MemberServiceLocal memberService = this.getMemberService();
    PersonNameType employeeName = this.getEmployeeNameById(employeeId);
    Member member = memberService.create();

    // use the Util.concat method to combine the names of the employee in one string
    member.setName(Util.concat(new String[] {employeeName.getFormOfAddressName(), " ", employeeName.getGivenName(), " ", employeeName.getFamilyName()}));
    member.setRole(role);
    member.setEmployeeId(employeeId);
    memberService.update(member);
```
String memberId = member.getKey();

this.getTeamService().addMembers(teamId, memberId);

return memberId;

private boolean canCreateMemberForEmployeeId(String employeeId) throws CAFFindException {
    MemberServiceLocal memberService = this.getMemberService();
    QueryFilter queryFilter = QueryFilterFactory.createFilter(Condition.EQ, employeeId);
    Collection<Member> members = memberService.findByEmployeeId(queryFilter);
    return members == null || members.isEmpty();
}

private PersonNameType getEmployeeNameById(String employeeId) throws CAFServiceException {
    EmployeeBasicDataByEmployeeQueryMessageType inputMessage = new EmployeeBasicDataByEmployeeQueryMessageType();
    EmployeeBasicDataByEmployeeQueryMessageTypeCS1E data = new EmployeeBasicDataByEmployeeQueryMessageTypeCS1E();
    EmployeeIDType id = new EmployeeIDType();
    id.setSimpleContent(employeeId);
    data.setEmployeeID(id);
    inputMessage.setEmployeeBasicDataSelectionByEmployee(data);
    EmployeeBasicDataByEmployeeResponseMessageType outputMessage = this.EmployeeBasicDataByEmployeeQueryResponse_In(inputMessage);
    return outputMessage.getEmployee().getCommon().getName();
}

private boolean existsProject(String projectId) {
    ProjectServiceLocal projectService = this.getProjectService();
    Project project = null;
    try {
        project = projectService.read(projectId);
    } catch (Exception e) {
        project = null;
    }
    return project != null;
}

If there is an error displayed because of not imported data types correct this by organizing imports.
Right click in the source window and choose “Source->Organize Import” (Keystroke combination: Ctrl+Shift+o).
18. Click Next.

19. Click Finish. All the compiler errors should now be gone.
20. In the **Implementation** tab of the **Team** editor click on the link to the implementation file of the business object node.

21. By default the **TeamBeanImpl** class has an empty implementation, but you can add custom methods or override some of the existing ones.

Right-click somewhere in the editor and select **Source -> Override/Implement Methods**...
22. Click Deselect All button and Select update method to override.

23. Select the update method as shown on the figure.

24. Replace its content with the code shown below.
Navigate to method "update" and insert the following code:

```java
@override
public void update(Team Input) throws CAFUpdateException, CAFOptimisticLockException,
CAFPPessimisticLockException {
    String description = Input.getDescription();

    if (description == null || description.equals("")) {
        Input.setDescription(Utils.concat(new String[] {Input.getName(), ", department number: ", Input.getDepartmentNumber()}));
    }
    super.update(Input);
}
```

If there is an error displayed because of not imported data types correct this by organizing imports.
Right click in the source window and choose Source->Organize Import" (Keystroke combination: Ctrl+Shift+o).
You may need to do this several times while entering the code.
Exercise 7: Expose Application Service as Web Service

In this exercise you will learn how to expose methods of application service as a web service.

1. In the Composite Application Explorer view, right click on Application Service ProjectManagerService and chose Expose service as Web Service from the context menu.

2. Provide the webservice a name and select the methods you would like to expose as shown on the figure and press OK.
Save, Generate, Build and Deploy application

3. In the Composite Application Explorer view highlight the project node ce350ex and choose (in sequence):
   - Save application
   - Generate application
   - Build application
   - Deploy application

4. For Deployment, you may need to login with administrator credentials.
Exercise 8: Run the Application

In this exercise we will configure and run the composite application. Refer also recording Configure, test and run the scenario (3 min).

1. Launch the CAF Runtime Configuration by entering the URL http://<server>:<port>/caf. You may need to login with administrator credentials.

   Select Administrative tools

2. Select External Service Configuration

   Navigate External Services menu to demo.sap.com/ce250ex -> Web Services and select EmployeeBasicData service.

   On the Right Hand Side: Navigate Destinations menu to Web Services and select EMPLOYEE_DATA destination.

   Press Map button.

3. Select Service Registry

   Welcome Administrator! Business Entities

   Business Entities

   Name

   Business Entity Name
4. Launch the CAF Runtime Configuration by entering the URL `http://<server>:<port>/caf`. You may need to login with administrator credentials. Select Test tools -> Service Browser.

In the application that appeared navigate to demo.sap.com -> xsolce151 -> Project -> findAll. In this way you will execute a query for all projects that are available and the results will be presented in the right side of the application.

Choose one of the projects and copy its key to the clipboard. You will need it later to create a team.
5. Launch the Web Service Navigator by entering the URL http://<server>:<port>/wsnavigator. You may need to login with administrator credentials.

6. Enter “ce350ex” in the input field for filter criteria and press ENTER key.

7. Click on the appearing entry “…_ProjectManagerService”

8. Select createTeam operation to invoke.

9. Paste the project id that you searched previously and enter a name of the team as shown on the figure. Press Execute button to invoke the operation.

10. As result you will retrieve the id of the newly created team. Copy this id to the clipboard as you will need it later to create team members.
11. Go to Select Operation step and select createMember.

12. Paste the team id and enter the other fields with data as shown on the figure. Press Execute button. Use employee id (100264,100265)

13. In the Service Browser navigate to demo.sap.com -> ce350ex -> modelled.Team -> Team -> findAll. This will execute a query for all instances of Team and its associated Members and you will see the team and member you just created.
14. In the **WebService Navigator** go to step **Enter Input Parameters** to create another team member. Paste the team id and enter the other fields with data as shown on the figure. Press **Execute** button.

15. In the **Service Browser** select **findAll** again. You will see that the new member instance that we created has appeared.

16. In the **WebService Navigator** go to step **Enter Input Parameters** to create another team member with the same employeeId. Paste the team id and enter the other fields with data as shown on the figure. Press **Execute** button.
17. You will be notified that there is an employee with such id already assigned to this project.
Appendix

Setting up the Environment

Start Developer Studio by double clicking the shortcut on your desktop.

Developer Studio starts.

Before you start developing with SAP NetWeaver Developer Studio you have to make sure that the system settings (e.g. message server port) are set properly.

Navigate to the Welcome page of your local J2EE engine:

Start->Programs->SAP NetWeaver Composition Environment 7.1 -> Application Server ->Welcome Page

Click on icon System Information.
**Advanced Service Composition with SAP Composite Application Framework Capabilities in SAP NetWeaver CE 7.1**

Be aware of the *Message Server Port* (for example “3602” in the screenshot on the right)

In SAP NetWeaver Developer Studio, navigate to *Window/Preferences* and highlight node *SAP AS Java*.

Make sure the message server host and port are set properly.

**Working with Remote Projects**

In this exercise you will learn how to work with remote CAF projects. The idea of the scenario is to show concurrent work on the same project with two IDEs, which involves resolving model-level conflicts with CAF Merge Tool.

1. **IDE1 & IDE2**: Login to development configuration.
2. **IDE1 & IDE2**: Enter your password.

3. **IDE1**: Start new project wizard as shown on the figure.
4. **IDE1**: Select Development Component as type of your project. Press Next.
5. **IDE1**: Select *Composite Application* as type of your development component. Press *Next*.
6. **IDE1**: Select your development configuration and then the software component in which you would like to create your CAF DC. Press **Next**.
7. **IDE1**: Enter name and Support Component of your CAF DC as shown on the figure.

*Note:* If somebody has already created a project with the name shown on the figure, you will not be able to create it. So, please, add some random string to the name shown on the figure. Press Next.
8. **IDE1:** You should create a new activity for the project. Press *Create Activity* button to start the wizard.

9. **IDE1:** Enter name for the activity you would like to create. Press *OK.*
10. **IDE1:** You can see your newly created activity. Select it and press *Next.*
11. **IDE1:** You can see an overview of the projects that will be created for you. Press *Finish*. This will start the project creation process.
12. **IDE1**: In the newly created project expand `home.<your_project_name>`. Right-click on modelled package and select *New Application Service* from the popup menu.

13. **IDE1**: Enter a name of the application service as shown on the figure. Press *Finish*. 
14. **IDE1:** Save and generate your project. During generation a popup as the one shown on the figure will appear. It will ask you if the files in the list should be added to activity. Press OK.

15. **IDE1:** Open Development Infrastructure perspective and there navigate to Open Activities view. Expand the tree to your activity as shown on the figure. Right-click on the activity and select Checkin from the popup menu. Confirmation dialog will appear in which you should select OK.

16. **IDE1:** After the checkin process has finished you will be asked if you want to activate the activity. Press No.
17. **IDE2:** Open Development Infrastructure perspective and in the Component Browser view navigate to your newly created CAF project. If it does not appear you may need to refresh the containing software component as shown on the figure.

18. **IDE2:** Select the 5 projects that constitute your CAF application. Right-click on them and select *Sync / Create Project -> Create Inactive Project* from the popup menu.
19. **IDE2**: In the dialog that appears deselect the **Build DCs after sync** checkbox and press **OK**. After that you can open the CAF perspective and see the project you just synced.

20. **IDE1**: Open **TestService** editor and navigate to its **Operations** tab. Press **Add** button to create a new operation.
21. **IDE1**: Enter name of the new operation as shown on the figure and press *Finish*.

**Screenshot:**

![Image of the New Operation window](image)

**New Operation**

**Specify required fields**

- **Name**: op1
- **Transaction type**: REQUIRED
- **Permission check**: Disable
- **Implemented**: Yes

**Finish**  
**Cancel**

22. **IDE1**: You will be prompted to checkout the CAF model file. Press *OK*.

**Screenshot:**

![Image of the Select elements window](image)

**Select elements**

- **Select All**  
  - File: SAG_SPS_DN=home2/remotecat2/metadatasap.com/src/MF/caf_1.xml

**Cancel**
23. **IDE1**: Create new activity and press **OK**.

24. **IDE1**: Save and generate the project. During generation a popup as the one shown on the figure will appear. It will ask you if the file in the list should be checked out for edit. Press **OK**.
25. **IDE2:** Open the TestService editor and navigate to its Operations tab. Press Add button to create a new operation.

26. **IDE2:** Enter name of the new operation as shown on the figure and press Finish.
27. **IDE2**: You will be prompted to checkout the CAF model file. Press **OK**.

28. **IDE2**: Create new activity and press **OK**.
29. **IDE2**: Save and generate the project. During generation a popup as the one shown on the figure will appear. It will ask you if the file in the list should be checked out for edit. Press **OK**.

30. **IDE1**: Open Development Infrastructure perspective and there navigate to *Open Activities* view. Expand the tree to your activity as shown on the figure. Right-click on the activity and select **Checkin** from the popup menu. Confirmation dialog will appear in which you should select **OK**. After the checkin do not activate the activity.

31. **IDE2**: Open Development Infrastructure perspective and there navigate to *Open Activities* view. Expand the tree to your activity as shown on the figure. Right-click on the activity and select **Checkin** from the popup menu. Confirmation dialog will appear in which you should select **OK**.
32. **IDE2**: The checkin process should fail, saying you that you have to sync your project to latest revision.

33. **IDE2**: Open **Composite Application** perspective and right-click on your project in the **Composite Application Explorer** view. Select **DTR -> Sync** from the popup menu.

34. **IDE2**: Open **Development Infrastructure** perspective again. You will see that the files in your activity have conflicts that need to be resolved. Select the CAF model file – `ca_model.cafmm` – and right-click on it. In the popup menu that appears select **Resolve Conflict -> CAF Resolve**.
35. **IDE2:** The merge tool will appear. Expand the trees on both sides so that you see the changes (surrounded by rectangles). From the left side select op2 and press the *Accept local* icon ( ($.icon Ottawa Steakhouse)) Then select op1 from the right side and press the *Accept remote* icon ( ($.icon Ottawa Steakhouse)). With this you will get the changes that were made in IDE1.

After you have completed these tasks save your changes and close the merge tool editor.

36. **IDE2:** On the message that appears (see the figure). Press *Yes*.

37. **IDE2:** Open *Composite Application* perspective and open the *TestService* editor. Navigate to its *Operations* tab. You will see that both op1 and op2 are available there.
38. **IDE2: Open Development Infrastructure perspective.** Select the file with custom code (TestServiceBeanImpl.java), right-click on it and in the popup menu that appeared select Resolve Conflict -> Merge.

39. **IDE2: The java merger will appear.** Select the code in the red rectangle in the right side of the merger and press the accept icon (✓). Press Accept Changes button. On the message that appears select that you would like to accept the local version of the file.
40. **IDE2:** Open Composite Application perspective and Generate the CAF application. Then open Development Infrastructure perspective and there navigate to Open Activities view. Right-click on your activity and select Checkin from the popup menu. Confirmation dialog will appear in which you should select OK. After the checkin do not activate the activity.

41. **IDE1:** Open Composite Application perspective, right-click on your project and in the popup menu that appeared select DTR -> Sync.

42. **IDE1:** After the sync finishes you can open the TestService editor, navigate to its Operations tab and see that both op1 and op2 operations are available.