Creating an Application Service
© Copyright 2005 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.

IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, and Informix are trademarks or registered trademarks of IBM Corporation in the United States and/or other countries.

Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.

Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®. World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressively prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP’s Support Services and may not be modified or altered in any way.
# Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example Text</strong></td>
<td>Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>Emphasized words or phrases in body text, graphic titles, and table titles.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.</td>
</tr>
<tr>
<td><strong>Example text</strong></td>
<td>Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.</td>
</tr>
<tr>
<td><strong>&lt;Example text&gt;</strong></td>
<td>Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.</td>
</tr>
<tr>
<td><strong>EXAMPLE TEXT</strong></td>
<td>Keys on the keyboard, for example, F2 or ENTER.</td>
</tr>
</tbody>
</table>

## Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>🚨</td>
<td>Caution</td>
</tr>
<tr>
<td>📄</td>
<td>Example</td>
</tr>
<tr>
<td>📝</td>
<td>Note</td>
</tr>
<tr>
<td>🕵️</td>
<td>Recommendation</td>
</tr>
<tr>
<td>💻</td>
<td>Syntax</td>
</tr>
</tbody>
</table>
**Contents**

- Requirements and Dependencies ................................................................. 2
- Applicable Releases ....................................................................................... 2
- Disclaimer ....................................................................................................... 2
- Scenario .......................................................................................................... 2
- Creating an Application Service ................................................................. 4
  - Implementation of created methods ......................................................... 12
- Service Browser: Testing the Application Service ....................................... 18
- Configuring UI Pattern for the Application Service ...................................... 21
Requirements and Dependencies

Before you start with this tutorial you should have installed the following Software:

- SAP Web Application Server Java 7.0
- SAP NetWeaver Developer Studio 7.0

This tutorial is based on the following guides:

- Create a local entity service with maintenance UI
- Create complex data types and Java Data Dictionary types

Additionally you should have configured CAF as described in the How-to Guide “Installation and Configuration Guide”.

If you do not have any experience with Entity Services and usage of complex data types already, we recommend that you to complete the above mentioned tutorials before starting with this one.

Applicable Releases

This tutorial is compatible with the following releases:

- SAP NetWeaver ’04s
- SAP Composite Application Framework (CAF) 7.0

Disclaimer

Any software coding and/or code lines / strings (“Code”) included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, except if such damages were caused by SAP intentionally or grossly negligent.

Scenario

The following tutorial shows you how to create an Application Service. Additionally you will learn how to test an Application Service with the Service Browser and configure UI patterns for application services.

Application services can use both entity and external services in an application. It is the most flexible of all services and contains the business logic of your application.

Here you also have the flexibility of being able to code for your specific application needs.

You use Application Services to present data to the user interface.
In this tutorial you will only create an Application Service for one of the Entity Services. In real life you would either need one for each Entity Service or one for all / several Entity Services.

The recommendation is to access the Entity Services only via Application Services.
Creating an Application Service

1) Open the carpool project in NetWeaver Development Studio. Create a new Application Service by selecting New from the context menu of Application Services in the Service Explorer.

2) Double click on the Application Service *TravelLocationApp* in the Service Explorer. Switch to the *Dependencies* tab. Select the *TravelLocation* Entity service from the *Service Catalog* on the left and click on the "Add" button. The *TravelLocation* will be added to the *Available Services* list.
3) Switch to the Operations tab. Click on the “Add” button to add a new operation. Select Access and Create, Read, Update, and Delete as the types of operation.

Click Next button.
4) Enter the following in the New Operation window.

Description: CRUD Methods
Permission Check: Disabled

Select TravelLocation from the tree.

Click Next button.

5) In the next window, select the attribute id and name. These are the input parameters for the create method.
Click Finish button. The new operations `createTravelLocation`, `readTravelLocation`, `updateTravelLocation` and `deleteTravelLocation` are added to the Operations list.
6) Mark the first (create) operation and from the Attributes/Type Repository, select Faults->caf.core->ServiceException and click on the Fault button. The ServiceException will appear in the right pane.

Repeat the same for every operation created.

<Short explanation for needed Service Exceptions -> when to use which one>

7) Additionally to these CRUD methods, you will define a Query for the entity TravelLocation.

Switch to the Operations tab and click on the Add button to add a new operation. Then select Query.
Click Next.

Now please enter the following data:

Name: findByld
Description: Find travel location by ID
Permission Check: Disabled
Select TravelLocation from the tree.

Click Next.
Now you can select the search attribute. In this case its *id*.

![Screenshot of selecting search attribute]

Click *Finish*.

**Note**: Don’t forget to add the exception *ServiceException* as mentioned in the step 6.

8) Now you will create another Query method for retrieving all available TravelLocations.

Therefore, please see the necessary steps under 7) and enter the following data:

- **Name**: `getAll`
- **Description**: Gets all TravelLocations
- **Permission Check**: Disabled

Select *TravelLocation* from the tree and click next.

In the next screen, do not select a search attribute. This means you get all locations.

Click *Finish*.

**Note**: Do not forget to add the exception *ServiceException* as mentioned in step 6.
Implementation of created methods

Switch to the Implementation tab. You have to write the code for the ‘CRUD’ methods (Create, Read, Update and Delete) as well as for the queries you just created.

1) But first you have to create a new helper method for generating unique Ids for the travel locations created. Add the following code between the custom code tags:

```java
public static synchronized long getUniqueId() {
    return System.currentTimeMillis();
}
```

Note: This is a very simple approach to creating a unique id for the purposes of this example.
2) Navigate to the `createTravelLocation` method in the try block between the `custom code` tags. Delete the generated instruction `retValue = null;` Add the following code instead:

```java
long uid = getUniqueId();
retValue = getTravelLocationService().create(uid);
retValue.setName(name);
getTravelLocationService().update(retValue);
```

3) In the `readTravelLocation` method
Delete the generated instruction `retValue = null;` Add the following code:

```java
retValue = getTravelLocationService().read(key);
```
4) In the `updateTravelLocation` method add the following code in the try block.
```
getTravelLocationService().update(dS0);
```

5) In the `deleteLocation` method add the following code in the try block.
```
getTravelLocationService().delete(dS0);
```
public void deleteTravelLocation(com.sap.cafsrv.travellocation.TravelLocation d3)
{
    // logging
    java.lang.String CAU_user = sessionContext.getCallerPrincipal().getName();
    java.lang.String CAU_methodHeader = TravelLocationAppBean.JAR_REQUEST + "del";
    Object[] CAU_parameters = new Object[] {d3};
    com.sap.cafsrv.util.CAFPublicLogger.entering(CAU_user, TravelLocationAppBean.JAR_REQUEST + "del" , CAU_methodHeader, CAU_parameters);
}

try {
    // custom code start - deleteTravelLocation(com.sap.cafsrv.travellocation.TravelLocation d3)
    getTravelLocationService().delete(d3);
}
finally {
    // custom code end - deleteTravelLocation(com.sap.cafsrv.travellocation.TravelLocation d3)
    com.sap.cafsrv.util.CAFPublicLogger.exiting(CAU_user, TravelLocationAppBean.JAR_REQUEST + "del", d3);
}
6) In the `getAll` method
Delete the generated instruction `retValue = null;`
Add the following code:
```
retValue = getTravelLocationService().getAll();
```

7) In the `findById` method
Delete the generated instruction `retValue = null;`
Add the following code in the try block.
```
retValue = getTravelLocationService().findById(id);
```
8) Now the Application Service is created. Save the metadata, generate the project code, build the DCs and deploy the project.
**Service Browser: Testing the Application Service**

The Service Browser is the standard method for testing application services.

You can either start with the URL:

http://<was_host>:<was_port>/webdynpro(dispatcher/sap~UI~servicebrowser/ServiceBrowser

Or, directly from the Netweaver Developer Studio, highlight the corresponding application service and select **Test** from the context menu.

1) First you will test the CRUD methods. To do this, navigate in the tree on the left side to your application service and select the references entity (TravelLocation).
2) You can find the operations, which are ‘representing’ your defined CRUD method on the right side.

Here you can create (New), update (Save), delete (Delete) and read (Find) travel locations.

3) Next you will test the queries (findBy methods) by selecting them in the tree on the left side.
Depending on the travel locations you created in the previous step, the screen for
the query `getAll` could look like this:
Configuring UI Pattern for the Application Service

You will use two UI patterns for testing the Application Service TravelLocationApp.

- **Object Editor** – for creating TravelLocation
- **Relation Tab** – for creating the address for the Travel Location

1) Launch the CAF Ui configuration Browser using the URL http://<was_host>:<was-port>/webdynpro/dispatcher/sap.com/caf~UI~configbrowser/ConfigBrowser

2) Select the Relation Tab from the left pane.

3) Click on **New Configuration** button. In the **Relation Tab Configurator** screen, select/enter the following.

   - **Configuration Name**: carpool_TravelLocation
   - **Service**: sap.com/carpool/TravelLocationService
   - **Header Aspect**: TravelLocation_address
   - **Max row count in table**: 1
   - **Button Title**: Edit Address
Click on the **Retrieve Metadata** button and select the fields *address, city, zip, country* (Press the *ctrl* key for multiple selections).

**Note**: Check the *Enabled* checkbox for these fields

4) Click on *Save + Return* button.

5) Select **Object Editor** from the Configuration Browser. Create a new Configuration with the following details.

*Configuration Name*: carpool_TravelLocationCreate

*Service*: sap.com/carpool/TravelLocationApp

*Header Aspect*: TravelLocation

Click on the **Retrieve Metadata** button

In "Buttons that should appear…", enter

*Button Name*: Submit

*Title of Object*: Create Travel Location
Object Editor Configuration

### Attributes to be shown in Object Editor Header

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreatedBy</td>
<td>CreatedBy</td>
</tr>
<tr>
<td>Name1</td>
<td>Name1</td>
</tr>
<tr>
<td>LastChangedBy</td>
<td>LastChangedBy</td>
</tr>
<tr>
<td>Name2</td>
<td>Name2</td>
</tr>
</tbody>
</table>

### Buttons that should appear as footer of Object Editor

<table>
<thead>
<tr>
<th>Operation</th>
<th>Name</th>
<th>Application Name</th>
<th>DC name</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Title of Object to appear in header (e.g Application Information): Create Travel Location
Object has own component implementation
Component: com.sap.corporate
Development Component: com.sap.corporate
Configuration name: com.sap.corporate
6) Now you need to add the Relation tab to the Object Editor. In the tab *Settings section* at the bottom of the screen, select the first row and enter/select

**Name to appear for tab:** Address  
**Associated aspect:** address

Click on the *Select Tab ConfigName* button at the end of the row and select `sap.com/caf~UI~ptn~objecteditor -> RelationTab`. Select the configuration name `carpool_TravelLocation` from the dropdown. Click OK.

Now the Object Editor screen will be populated as shown below.
7) Click the **Save + Return** button to save the Object Editor configuration.
8) Test the UI by launching the URL
http://<was_host>:<was_port>/webdynpro/dispatcher/sap.com/caf~UI~ptn~object editor/ObjectEditor?app.configName=carpool_TravelLocationCreate

You will see a window:

Click on the Edit Address button to enter a New Address.

9) This completes the configuration and testing of UI patterns for Application Services.
www.sap.com/netweaver