SAP Composite Application Framework

Implementing Remote Persistency with CAF and RFCs
## Typographic Conventions

<table>
<thead>
<tr>
<th>Type Style</th>
<th>Represents</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Example Text</em></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>Example text</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>Example text</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>Example text</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><em>&lt;Example text&gt;</em></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

Business Scenario ................................................................. 2
Requirements and Dependencies ............................................... 2
  Applicable Releases .......................................................... 2
  Disclaimer ........................................................................... 2
Creating a Development Component Project ......................... 2
Business Scenario

In this tutorial we describe, how remote persistency can be achieved and which prerequisites the external operation (RFC’s in our case) must fulfill so that it works nicely together with the CAF entity services methods.

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

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.

Creating a Development Component Project

Create a new Development Component Project called “xrfc” (Type Composite Application Services).
Before starting with the development of the Entity, import the following external services:

- **Right click on External Services and choose Import**

- **Choose “RFC - ABAP**

- **Import them from a preconfigured SAP system, e.g. DEMO.**
Enter “CAF*” for function name and click Search. Select the following functions:

- CAFTESTBO_CHANGE
- CAFTESTBO_CREATE
- CAFTESTBO_DELETE
- CAFTESTBO_GET
- CAFTESTBO_GETLIST

Choose Finish.

Create a Person entity with the following attributes:

<table>
<thead>
<tr>
<th>Name</th>
<th>Data Type</th>
<th>Special Options</th>
<th>Cardinality</th>
</tr>
</thead>
<tbody>
<tr>
<td>citizenNo</td>
<td>shortText</td>
<td>Key Field</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>checked</td>
<td></td>
</tr>
<tr>
<td>name</td>
<td>shortText</td>
<td>Mandatory = true</td>
<td>1..1</td>
</tr>
<tr>
<td>age</td>
<td>int</td>
<td>Mandatory = true</td>
<td>1..1</td>
</tr>
<tr>
<td>description</td>
<td>longText</td>
<td>Mandatory = true</td>
<td>1..1</td>
</tr>
</tbody>
</table>

**IMPORTANT**: By making the fields mandatory, the generated create-method will have those mandatory fields as input parameters which can then be mapped against the input parameters of external create methods! Otherwise the CAF would create empty create methods which make not much sense if mapped against external services.
Next add a **findByCitizenNo** method:

![New Operation window](image)

Remove the **Local Persistency** checkmark on the **Persistency** tabstrip.

![Persistency tabstrip](image)

Assign the external service **CAFTESTBO_CREATE** to the create method and map the appropriate fields. The result looks like the one depicted here:
What does the implementation do?

It simply checks, whether this id already exists and if so it raises an exception. Otherwise it inserts the new entity into the appropriate table. The ABAP code looks like this:
For testing purposes no permission checks are needed. So remove the checkmark on the Permissions tabstrip.

Now generate all project code, build everything and deploy the project to the J2EE server. Next configure the external service CAFTESTBO_CREATE with the CAF External Service Configurator.

The configuration should look like this:
So next we will assign the update external service to the update call of the entity. Assign the external service **CAFTESTBO_CHANGE** to the update method and map the appropriate fields. The result looks like the one depicted here:

**What does the implementation do?**

It simply checks, whether this id already exists and if not it raises an exception. Otherwise it updates the entity within the appropriate table. The ABAP code looks like this:
FUNCTION CAFTESTBO_CHANGE.

**Local interface:**

**IMPORTING**

```
VALUE(ID) LIKE ZFSCAFTESTBO-ID
VALUE(NAME) LIKE ZFSCAFTESTBO-NAME
VALUE(DESCRIPTION) LIKE ZFSCAFTESTBO-DESCRIPTION
VALUE(CREATED) LIKE ZFSCAFTESTBO-CREATED
VALUE(AGE) LIKE ZFSCAFTESTBO-AGE
```

**EXCEPTIONS**

```
OBJECT_NOT_FOUND
```

*Tables ZFSCAFTESTBO.*

```
select single * from ZFSCAFTESTBO
where id = id.
if sy-subrc ne 0.
  raise object_not_found.
endif.

move:
  name to ZFSCAFTESTBO-name,
description to ZFSCAFTESTBO-description,
created to ZFSCAFTESTBO-created,
age to ZFSCAFTESTBO-age

* modify ZFSCAFTESTBO.
update ZFSCAFTESTBO.
```

Now generate all project code, build everything and deploy the project to the J2EE server. Next configure the external service CAFTESTBO_CHANGE with the CAF External Service Configurator.

The configuration should look like this:
Now the creation of a new entity should work. The update call goes through till the backend and the entry appears now in CAF's internal table:

<table>
<thead>
<tr>
<th>Object</th>
<th>Category</th>
<th>Created</th>
<th>LastChange</th>
<th>AdminUser</th>
<th>Access</th>
<th>Name</th>
<th>Age</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID 234</td>
<td>Engineer</td>
<td>01/01/2020</td>
<td>12/31/2020</td>
<td>admin</td>
<td>read</td>
<td>John</td>
<td>25</td>
<td>Engineer</td>
</tr>
</tbody>
</table>

The data itself is not saved as can be seen from the figure above only the primary key of the remote persistence service.

Next we want to map the final functions: read, delete and findBy. We start with the read function. To do this assign the external service `CAFTESTBO_GET` to the read method and map the appropriate fields. The result looks like the one depicted here:
What does the implementation do?

It simply reads the data out of the database table and if no data could be read an exception is thrown. If data is available it is moved to the return parameters. The ABAP code looks like this:

```abap
FUNCTION CAFTESTBO_GET.
**
** " Local interface:
** " IMPORTING
** " VALUE (ID) LIKE ZFSCAFTESTBO-ID.
** " EXPORTING
** " VALUE (NAME) LIKE ZFSCAFTESTBO-NAME
** " VALUE (DESCRIPTION) LIKE ZFSCAFTESTBO-DESCRIPTION
** " VALUE (CREATED) LIKE ZFSCAFTESTBO-CREATED
** " VALUE (AGE) LIKE ZFSCAFTESTBO-AGE
** " VALUE (ZID) LIKE ZFSCAFTESTBO-ID
** " EXCEPTIONS
** " OBJECT_NOT_FOUND
**
** " Tables ZFSCAFTESTBO

SELECT SINGLE * FROM ZFSCAFTESTBO WHERE ID = ID.

IF SY-SUBRC NE 0.
  RAISE OBJECT_NOT_FOUND.
ENDIF.

MOVE:
  ZFSCAFTESTBO-NAME TO NAME,
  ZFSCAFTESTBO-ID TO ZID,
  ZFSCAFTESTBO-DESCRIPTION TO DESCRIPTION,
  ZFSCAFTESTBO-CREATED TO CREATED,
  ZFSCAFTESTBO-AGE TO AGE.
```
Next we continue with the delete function. Assign the external service `CAFTESTBO_DELETE` to the delete method and map the appropriate fields. The result looks like the one depicted here:

![Entity service operations and External operations table]

What does the implementation do?

It simply checks whether the id exists and if not it raises an exception. Otherwise it deletes the entity within the appropriate table without. The ABAP code looks like this:

```abap
FUNCTION CAFTESTBO_DELETE.
    " Lokale Schnittstelle: "
    IMPORTING
    " VALUE(ID) LIKE ZFSCAFTESTBO-ID
    " EXCEPTIONS
    " OBJECT_NOT_FOUND
    " Tables ZFSCAFTESTBO.
    select single * from ZFSCAFTESTBO where id = id.
    if sy-subrc ne 0.
        raise object_not_found.
    endif.
    delete from ZFSCAFTESTBO where id = id
    "commit work.
ENDFUNCTION.
```
Finally we implement the `findBy` method. Assign the external service `CAFTESTBO_GETLIST` to the `findByCitizenNo` method and map the appropriate fields. The result looks like the one depicted here:

What does the implementation do?
It simply selects the entries from the database table into the return structure. The ABAP code looks like this:

```abap
FUNCTION CAFTESTBO_GETLIST.
**
**"Local interface:
** IMPORTING
** VALUE(id) LIKE ZFSCAFTESTBO-ID OPTIONAL
** TABLES
** LIST_OF_CUSTOMERS STRUCTURE ZFSCAFTESTBO
**
** Using ZFSCAFTESTBO.
if ( id eq space or id eq '').
    select * from ZFSCAFTESTBO into table LIST_OF_CUSTOMERS.
else.
    select * from ZFSCAFTESTBO into table LIST_OF_CUSTOMERS
    where id eq id.
endif.
ENDFUNCTION.
```
Now generate all project code, build everything and deploy the project to the J2EE server. Next configure the external service `CAFTESTBO_GET`, `CAFTESTBO_DELETE`, and `CAFTESTBO_GETLIST` with the CAF External Service Configurator as we did it before.

With this we have a fully functional entity with remote persistency.