Web3 (2.1 legacy document)

Table of contents

1 Comments.....6

Table of contents

1 About....3
2 Installing SAP R/3 (TM) JavaConnector....3
3 Configuring cocoon.xconf....3
3.1 Items....3
4 Implementing your own RFC's....4
4.1 Web3 rfc-Syntax....4
4.2 rfc:include....4
4.3 rfc:import....5
4.4 rfc:tables....5
4.5 rfc:structure....5
4.6 rfc:table....5
4.7 rfc:field....5
5 A more complex example....5
6 Setting up the sitemap....6
6.1 Global Configuration....6
6.2 Parametrized Configuration....6
7 Further questions....6

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. About

EFP Consulting Austria produced an open-source library called Web3 that allows you to integrate SAP R/3 seamlessly into Cocoon 2. With these components you are able to call Remote Function Calls via an easy to use XML-syntax. For most BAPIs and Remote enabled Function Calls you simply need a text editor.

This toolkit offers you ...

- ... synchron communication to any SAP system above release 3.1H.
- ... easy to use ABAP function calls from outside R/3 with a se37-like markup-syntax.

The following guide helps you to setup cocoon with your SAP R/3. Reasonably this guide cannot answer all questions dealing with your environment. For further questions be advised to contact your favorite SAP Consultant.

2. Installing SAP R/3 (TM) JavaConnector

Be sure to proper install the appropriate SAP Java-Connector from www.sap.com. To install the connector refer to the included documentations.

3. Configuring cocoon.xconf

With Web3 you have a flexible Toolkit where you can connect to multiple R/3 System within one single Cocoon- Instance. Enter your backend configuration into cocoon.xconf like this:

3.1. Items

Element	Description
web3	Declare your logger here.
pool	The pool element is the logical unit of all your SAP settings.
client	The systems client you log onto.
user, password, language	...
route	The route to your SAP system. Please refer to your SAP help to prepare the connection string.
system	The system-number of your SAP system you log onto.
gateway, program-id	Are mandatory and not used within Web3's szenario.
@trace	a boolean switch whether to use JCO's facility to trace the communication layer or not.
@level	if @trace is true set the trace-level to your prefered value. Please refer to the JCO's documentation about tracing.

@size	denotes the pool-size of your sap connection pool. Be aware that this is a hard-limited pool.

A configuration in your cocoon.xconf would look like this:

<web3 logger="core.web3"> <backend name="indy"> <pool level="0" size="10" trace="false"> <client>100</client> <user>user</user> <password>secret-passphrase</password> <language>DE</language> <route>indy</route> <system>00</system> <gateway>sapgw00</gateway> <program-id>USR-GR02</program-id> </pool> </backend> <backend name="hugo"> ... </web3>

4. Implementing your own RFC's

First of all you have to setup your markup with the right Namespace

<page xmlns:rfc="http://apache.org/cocoon/Web3-Rfc/1.0"> ... </page>

After doing so you can enter RFC-mappings with the following syntax. You will see this is a very easy task and you will enjoy using Web3 to do the stuff for you within your SAP environment :)

4.1. Web3 rfc-Syntax

Data definitions (metadata) are created and managed in the ABAP Dictionary. The ABAP Dictionary permits a central description of all the data used in the system without redundancies. New or modified information is automatically provided for all the system components. This ensures data integrity, data consistency and data security.

The syntax used within this markup builds upon these facts. So do not wonder if you find the corresponding logical units in the Cocoon frontend.

Element	Description
rfc:include	Starts a mapping for the specified RFC.
rfc:import	Container element for structures and fields.
rfc:tables	Container element for tables.
rfc:structure	Container element for fields.
rfc:table	Container element for structures.
rfc:field	The data-fields.

4.2. rfc:include

First when you start to map a RFC keep in mind to get the parameters from your SAP system. Therefore you may want to use se37 where all relevant parameters are listed. The @name has to be the name of the RFC you want to map.

Since many RFC's return hierarchical data split into tables you will encounter some problems in xslt when rendering these data to trees. So you may implement org.apache.cocoon.components.web3.Web3Streamer to get a proper XML to work with. You can set your own streamer in the @streamer.

Attribute	Description
@name	The name of the SAP RFC

@streamer	The class-name of your function streamer. Most of the time you won't need to implement it anyway.

4.3. rfc:import

When requesting a RFC you have to fill the import parameterlist. The import parameterlist lists all relevant parameters that have to be provided to get proper results from your R/3.

se37 can provide you with the relevant information.

4.4. rfc:tables

Some highly complex RFC's have also tables in their calling interface. You may need to read the SAP documentation of the BAPI you gotta call and when to make use of one of them.

4.5. rfc:structure

A structure comprises fields. A field can refer to an elementary type (via a data element or by directly specifying the data type and length in the structure definition), another structure or a table type. A structure can therefore be nested to any depth.

At the time the facility to use nested structures has not been tested!

4.6. rfc:table

A table consists of a collection of structures with same structure type. To keep it simple we can sloppily say a table is like a SQL-database-table.

4.7. rfc:field

Fields, also called data elements are the smallest indivisible units of the complex types and are used to specify the types in structures and columns of tables. A field describes either an elementary type or a reference type.

In ABAP exist predefined types and custom defined types. But don't worry about the horrible task of converting into the right type. This task is excellently done for you by the JavaConnector.

Keep in mind that the communication format for numbers and date is US. So supply Date information in format YYYY-MM-DD with the slashes and number information like DDDDD.DD where the dot is the comma!

If you want to check your RFC mappings for syntactically correctness you may want to use the enclosed dtd's.

5. A more complex example

The included example demonstrates a more complex function call. Keep in mind that the correct execution depends on the local customizing settings of the connected SAP System.

<page xmlns:rfc="http://apache.org/cocoon/Web3-Rfc/1.0"> <rfc:include name="XXX" streamer="XXX"> <rfc:import> <rfc:structure name="XXX"> <rfc:field name="XXX">YYY</rfc:field> <rfc:field name="XXX">YYY</rfc:field> <rfc:field name="XXX">YYY</rfc:field> </rfc:structure> <rfc:tables> <rfc:table name="XXX"> <rfc:structure name="1"> <rfc:field name="XXX">YYY</rfc:field> <rfc:field

name="XXX">YYY</rfc:field> <rfc:field name="XXX">YYY</rfc:field> </rfc:structure> </rfc:table> </rfc:tables> </rfc:import> </rfc:include> </page>

6. Setting up the sitemap

There are two ways to setup your Web3Transformer in the sitemap.

6.1. Global Configuration

If you have to deal only with one R/3 Instance you may configure your org.apache.cocoon.transformation.Web3RfcTransformer global like stated in the following sitemap snippet.

<map:transformer name="rfc" src="org.apache.cocoon.transformation.Web3RfcTransformer" logger="web3.transformation"> <system>indy</system> </map:transformer>

6.2. Parametrized Configuration

If you have setup several R/3 pools you can parametrize the org.apache.cocoon.transformation.Web3RfcTransformer as stated next

<map:transform type="rfc"> <map:parameter name="backend" value="indy"/> </map:transform>

7. Further questions

Enjoy using Web3. For further questions feel free to contact the author Michael Gerzabek

Built with Apache Cocoon

1. Comments

add your comments