Building Mash-Ups and simplifying Application Integration with RESTful Web Services in SAP ABAP

Applies to:

SAP NetWeaver, SOA, Web 2.0, ABAP , Mash-Ups

Summary

RESTful services have not raised much attention in the SAP community until today. Althogh the SAP has concentrated on the SOAP type services to support the idea of an SOA, it is worth to give RESTful services a closer look. This article describes how RESTful services can be build with the SAP ABAP framework and shows some examples how to employ RESTful techniques to application development in a SAP universe. For some problems coming with distributed SAP infrastructures in larger companies these type of service seem to be a really simple and obvious solution.

Author: Marcus Schiffer

Company: Evonik Industries

Created on: 7. September 2007

Author Bio

Marcus Schiffer is working with the IT services department of Evonik Industries. He is involved in developing SOA strategies and works as a project manager for innovative IT solutions.

Table of Contents

RESTful Services

Service Oriented Architecture (SOA) is one of the most discussed topics in IT today. Promises of a brave new world with IT agility, business flexibility and lower TCO are buzzing around every meeting on IT strategy and architecture in these days. IT managers are attracted by the idea of having a large portfolio of granular business functionality available at a fingertip. Visions of fast and cheap business process changes with easy integration of multiple IT systems and connections to the outside world are growing fast in the IT organisations.

SAP has boosted this idea with the E-SOA initiative, where lots of web services will be created and available in SAP ERP standard for various business functionalities in a new highly standarized fashion. So as one example the use of common data types is promoted to avaoid confusion about say a data type for calendar dates. These services are build using SOAP type services, a widely accepted standard for Web Services.

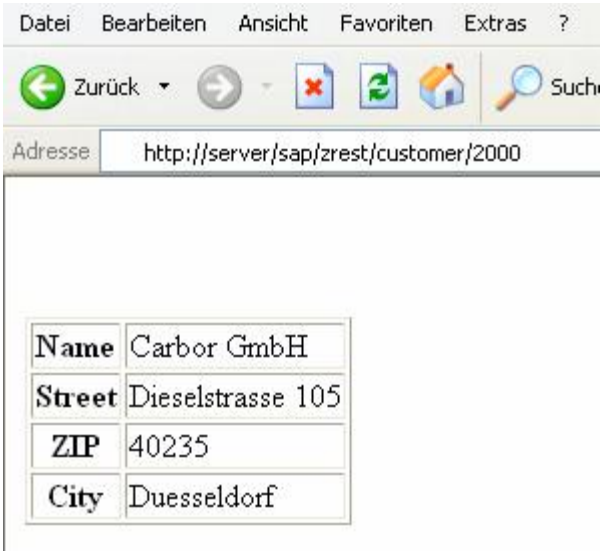
However, there is competition growing. Recently the Representational State Transfer (REST) type services are getting much attention. Also in SDN some pioneers addressed this topic See especially the article and blog of DJ Adams (Ref. [1], [2]), who followed this idea as early as 2004. Unfortunately, it seems there has not been much interest in RESTful services since then.

This article will pick up the topic again and try to show how REStful services can be used in a distributed environment for easy integration of resources from SAP systems into business applications. Also the aspect of application integration is briefly discussed.

Very short recap of REST

In the RESTful world everthing is a named resource. These named resources can be displayed and manipulated by addressing them with a simple URL containing the ID or name of the resource and the desired action. This action on the resource is determined by the well known http protocl methods (put for create, get for display, post for update , delete for delete). Ok, enough theory, you can find so much of the underlying philosophy in the internet or the original paper of Roy Fielding [Ref.3].

Lets try to make the idea clear using a simple example: Assume you want to see details of the customer identified by the customer number “2000” . Would it not be great to simply type in a corresponding URL in your browser identifying the resource of the customer and get all the desired information ? E.g. calling an URL like http://server/sap/zrest/customer/2000 would response like this



Baically this is the essence of a RESTful “get” service. The browser sends a get request to the server identifying the resource object, and the response is a representation of the object in html format. Note, that you do not need any more information (like a WSDL) about how to call the service or what kind of input fields are required.

Building RESTful services in SAP ABAP

Creating a RESTful service in SAP NetWeaver ABAP does not take much effort. Although the basic procedure has been well explained and discussed in detail already in Ref. [2], the proceeding is again shortly outlined.

Creating the handler and the service

The first step while building a new RESTful service in SAP NetWeaver ABAP is the creation of a new handler for the ICM framework. A new class needs to be created which inherits from the IF_HTTP_EXTENSION interface. Here the method IF_HTTP_EXTENSION~HANDLE_REQUEST has to be coded accordingly to deal with the REST service requests.

First in this method the URL of the request must be interpreted. This is done using the corresponding methods of the request object. The URL string of a request can be obtained using:

```
DATA: URL_INFO TYPE STRING.
URL_INFO = SERVER->REQUEST->GET_HEADER_FIELD( NAME = '~PATH_INFO' ).
```

In the example above the result for URL_INFO would be : “/2000”.

Now the http method is determined using another method of the request object:

```
DATA: HTTP_METHOD TYPE STRING.
CALL METHOD SERVER->REQUEST->GET_METHOD
                            RECEIVING
                             METHOD = HTTP_METHOD.
```

With this information the handler is now able to discriminate between the http methods PUT, GET, POST, DELETE. Furthermore the handler coding can react in the appropiate way by calling the corresponding function module to update, create, show or delete the resource.

For the current example it would be the “get” method and after the URL_INFO is shifted left by one character the actual customer ID resource name can be determined.

```
SHIFT URL_INFO LEFT BY 1 PLACES.
CUSTOMER_ID = URL_INFO.
```

Calling the BAPI for customer detail (BAPI_CUSTOMER_GETDETAIL) now provides detail information on e.g. the address data of the requested customer.

The only thing left is the wrapping of the resulting data from the BAPAI call in html code to form the body of the server response. (see Ref[2] for a coding example)

Finally the response is sent to the client by:

```
CALL METHOD SERVER->RESPONSE->SET_CDATA( DATA = W_BODY ).
```

Afte the handler is created, it can be used in a new service definition: Simply create a new service in the service tree of transaction SICF. If you do not use your own namespace a good place is somewhere under default_host/sap/. For our demonstartion, we choose /zrest/customers/ .

The restful service to show the details of customer 2000 can now be addressed with the URL http://server/sap/zrest/customers/2000 !

Output as XML

Typically web services are not only called from dialog transactions in a browser. Instead, they should also be able to support automatic communication between applications. The output representation needs to be more machine readable fur such a purpose. Of course, the representation of choice for such use cases is XML format. In the examplewe simply add a .xml to the REST request and interpret this in the handler with

```
SPLIT URL_INFO AT '.' INTO CUSTOMER EXTENSION.
```

The generation of XML output is now straight forward: Simply call the appropiate build-in XSLT transformation for the result table RESULT_TABLE of the forementioned BAPI call and transfer the result to the client:

```
CALL TRANSFORMATION ('ID')
SOURCE TAB = RESULT_TABLE
RESULT XML XML_OUT
OPTIONS XML_HEADER = 'WITHOUT_ENCODING'.
CALL METHOD SERVER->RESPONSE->SET_CDATA( DATA = XML_OUT ).
```

Calling the restful service by http://server/sap/zrest/customers/2000.xml will now treat the service request differently and provide XML output like:

POST, PUT, DELETE

The above example is dealing with the “get” requests from a typical browser. In priciple the SAP Netweaver Application server can deal with put, get, post and delete requests. However, in case of requests coming

from a browser we need to make some changes, since the browser typically only does send “get” or “post” requests. Usually hidden fields are used to simulate delete or put actions.

If the service is called from a proxy program instaed, of course all the methods are available by default.

For updating or creating a resource, the payload of the request also needs to interpreted. This can be done by calling the following method:

```
CALL METHOD SERVER->REQUEST->IF_HTTP_ENTITY~GET_CDATA
  RECEIVING
    DATA    = PAYLOAD
    .
```

The handler may now interpret the PAYLOAD data (e.g. XML data) and call an appropiate BAPI to create or update the resource.

Example applications and integration options

Now its time to show simple examples on how to integrate RESTful services into a business application.

REST and WebDynpro: Easy Mash-Ups

Assume your company has a CRM system running with multiple backend SAP systems connected. In the example scenario a sales agent logs in to a SAP CRM systems Web Client and gets a list of customer orders from an internet shop. The sales agent has to check the orders and in this process the agent needs further access to non replicated customer master data from the backend.

Here is a very simple demo of how this could be achieved with RESTful services:

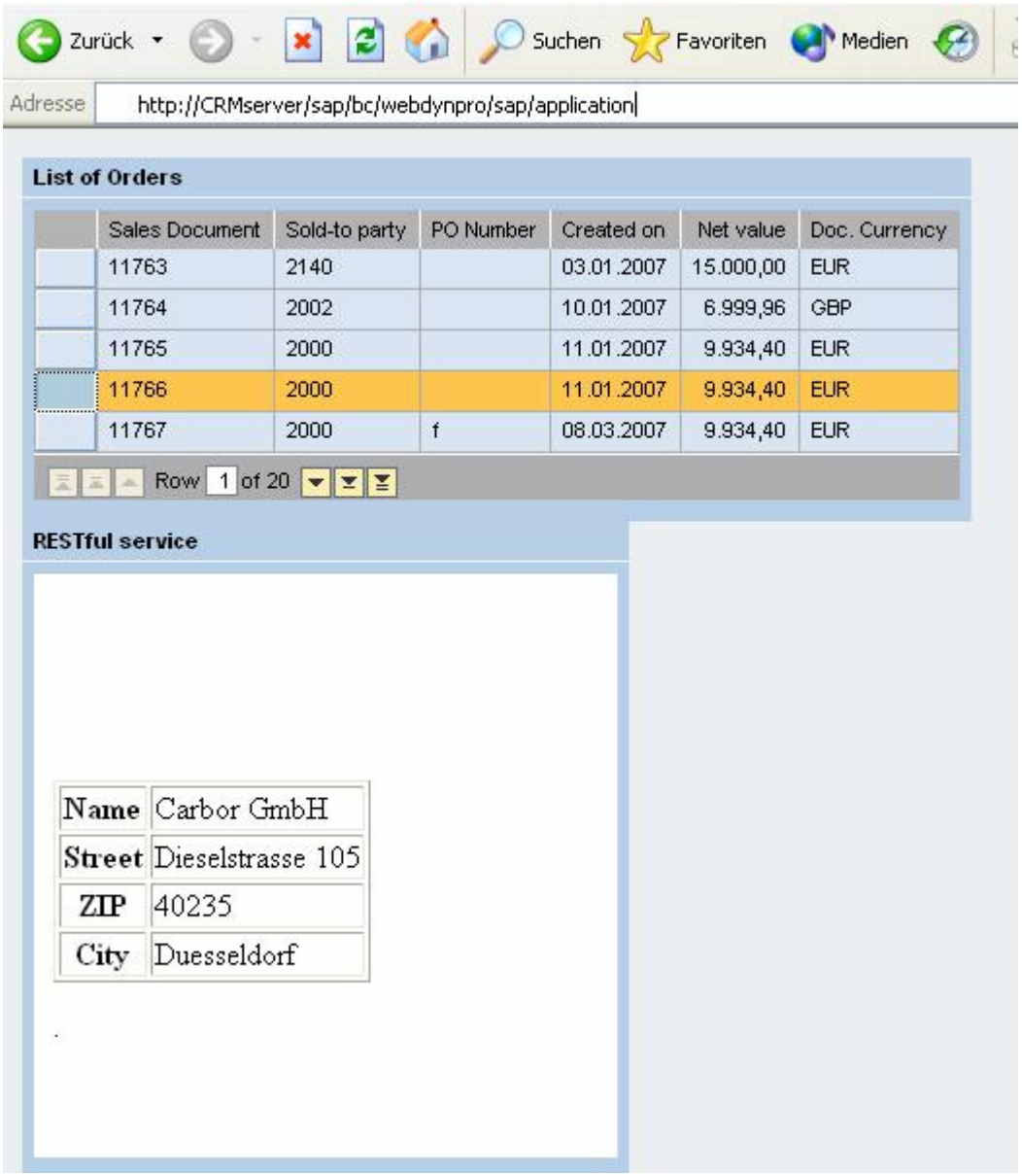
Using the WebDynpro for ABAP framework a WebDynpro application is build which simply creates a list of sales orders with some basic data using the tables element. In the WebDynpro view one additional element of type iframe is introduced which serves as container for the restful service.

In the very simple example, the table is populated with a list of customer order data on initialisation and an table row event is defined for “onSelect”.

With basic WebDynpro methods (see Ref.[4]), it is easy to extract the customer number from the table and in the implementation of the “onSelect” method, a context attribute “URL” of type string can be filled by

```
CONCATENATE ‘HTTP://SERVER/SAP/ZREST/CUSTOMERS/’  CUTOMERNO INTO URL
```

Now the attribute “source” of the iframe element can be bound to the context attribute “URL” filled with the url to a restful service for the respective customer resource. Each time the sales agent selects a row with customer order data the corresponding RESTful service displays the customer detail data in the iframe element.



In the terminology of Web 2.0 this is a Mash-Up. It is a web application mixing data from different sources into one coherent application framework. Using restful techniques makes the creation of such kind of applications very simple especially using SAP WebDynpro for ABAP. A next step could be e.g. the addition of google maps data for the customer location into the application.

Calling RESTful services from EXCEL

Many business applications need exchange of data between SAP and e.g. Microsoft Office applications. This is a use case for restful services with XML output in combination with VB for applications.

In an Excel macro the coding to call a SAP based restful service is very short. Using the current example an XML representation of the customer resource in a variable RESPONSETEXT can be retreived using:

```
DIM WINHTTP AS NEW WINHTTP.WINHTTPREQUEST
WINHTTP.OPEN "GET", "HTTP://SERVER/SAP/ZREST/CUSTOMER/2000.XML"
WINHTTP.SETCREDENTIALS "USER", "PASSWORD", 0
WINHTTP.SEND (DATA)
RESPONSETEXT = WINHTTP.RESPONSETEXT
```

(Other programming languages like JAVA or Ruby will support thhe same functionality in a similar way.)

Conclusion

It is amazing how simple it is to implement new technology ideas using the SAP NetWeaver ABAP framework. All the interfaces and methods needed are there at your fingertip. And all can be done by recycling the ABAP skills and experience already exsiting.

With RESTful services in SAP some classical requirements for application development can be solved with relatively low effort. Althogh the development of suitable handlers for the ICM framework will add individual concepts and non standard techniques to a restful project, it seems that the ease of using the resulting services is worth a compromise. Restful services enable the creation of lightweight Mash-Ups in a very intuitive way. As the Excel example shows, even for application – applicaton coupling there is no need for additional proxy components to call the restful services. Instaed of using WebServices toolkits, direct access to the resources is possible.

Restful services should be considered as possible solution to many problems in web application development for a distributed system landscape in larger organisations. The merging of information from various backend systems into portal applications, the feed of detail data to overview transactions on e.g. purchase order lists from multiple backend systems or the cross company transfer of logistical information could be potential use cases for RESTful services.

Related Content

- [1] Forget SOAP - build real web services with the ICF
- [2]Real Web Services with REST and ICF
- [3] Original Article of Roy Fielding on REST
- [4] WebDynpro Tutorials on help.sap.com

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.

