How to Develop Web Services in WebAS
Part 1: Making Sense of Business Server Pages

By Axel Angeli, logosworld.com

***Editor's Note**: Reading about hot new SAP® technologies like Web Application Server and Business Server Pages is one thing, understanding how they can help your company build a more web-friendly SAP architecture is another. Maybe it's because SAP technical guru Axel Angeli is based in Germany (SAP AG headquarters), or maybe it's just his passion for all things SAP, but Axel seems to have a better handle on SAP's new technology platform (and how to explain it) than anyone we know. We're very excited to have Axel on board for a series of articles on developing web services using SAP's Web Application Server. In his brilliant debut article, Axel not only lays the groundwork for a conceptual understanding of SAP web services, he also helps us with the simple questions we needed to know, like the differences between WebAS versions 6.1, 6.2, and 6.3, and why the Web Application Server is such a technical breakthrough in the first place.*

Technical Overview: Business Server Pages (BSP) are SAP's WebAS implementation of dynamic, server-side HTML content that add a new personality to the SAP kernel. BSPs resemble the common concept of server pages from other Web servers, such as the Microsoft Active Server Pages (ASP) and the Java Server Pages (JSP), but BSPs also add additional support for the ABAP/Objects language and full transparent access to the SAP WebAS repository. Hence, they are pretty easy to comprehend if one is somehow familiar with Web server programming. In this white paper, we'll break down all these new SAP technical concepts, and walk the reader through the creation of sample BSP pages.

SAP Enterprise Edition: A New Personality for SAP

The most (and probably only) significant change between SAP R/3 4.6 and SAP 4.7 (which received the fashionable name *SAP Enterprise Edition*) has been the addition of the *Web Application Server* (WebAS) to the kernel. This adds another personality to the SAP ERP suite by allowing it to act as a genuine Web server. While all traditional dynpros can still be accessed through well-known transactions via the SAPGUI, in 4.7, there now exists the possibility to develop dynamic Web pages that can be called from any HTTP client.

SAP Web Application Server is indeed a further development of the SAP application server technology. Based on the highly scalable SAP multi-tasking kernel, the SAP WebAS extends the technical capabilities of the SAP framework by adding the following main features:

- Adding the HTTP protocol to the SAP kernel, allowing the WebAS to appear as a common HTTP server to the outer world.
- WebAS can act as a simple HTTP client to request and accept HTTP messages from the Internet.
- WebAS allows users to develop dynamic Web pages within the SAP framework.
- WebAS allows users to develop Web pages in ABAP/Objects.
- Dynamic web pages allow for easy creation of Web services.
- Starting with WebAS Version 6.2, the WebAS will incorporate the SAP J2EE engine, thus allowing users to develop Web applications both in Java and ABAP. (See Figure 1.)

In traditional R/3, the CPIC-based RFC protocol has been the only supported external protocol. Although SAP provides for a large number of programming libraries on most common platforms and operating systems, it remains a proprietary solution, which is naturally not suitable for a universal EAI infrastructure. The RFC protocol must be converted to the generic protocol of the communication partner, e.g., COM for Windows or CORBA for Java.

The HTTP protocol, however, has matured as the generic application protocol for IP networks and is understood and spoken by all modern runtime frameworks. Therefore, the HTTP protocol allows a direct non-solicited conversation between different applications. With the WebAS personality, the SAP kernel is hence compatible with the countless HTTP applications existing inside and outside the enterprise—though this does not mean that the traditional ways to develop in ABAP and to access R/3 are no longer needed.

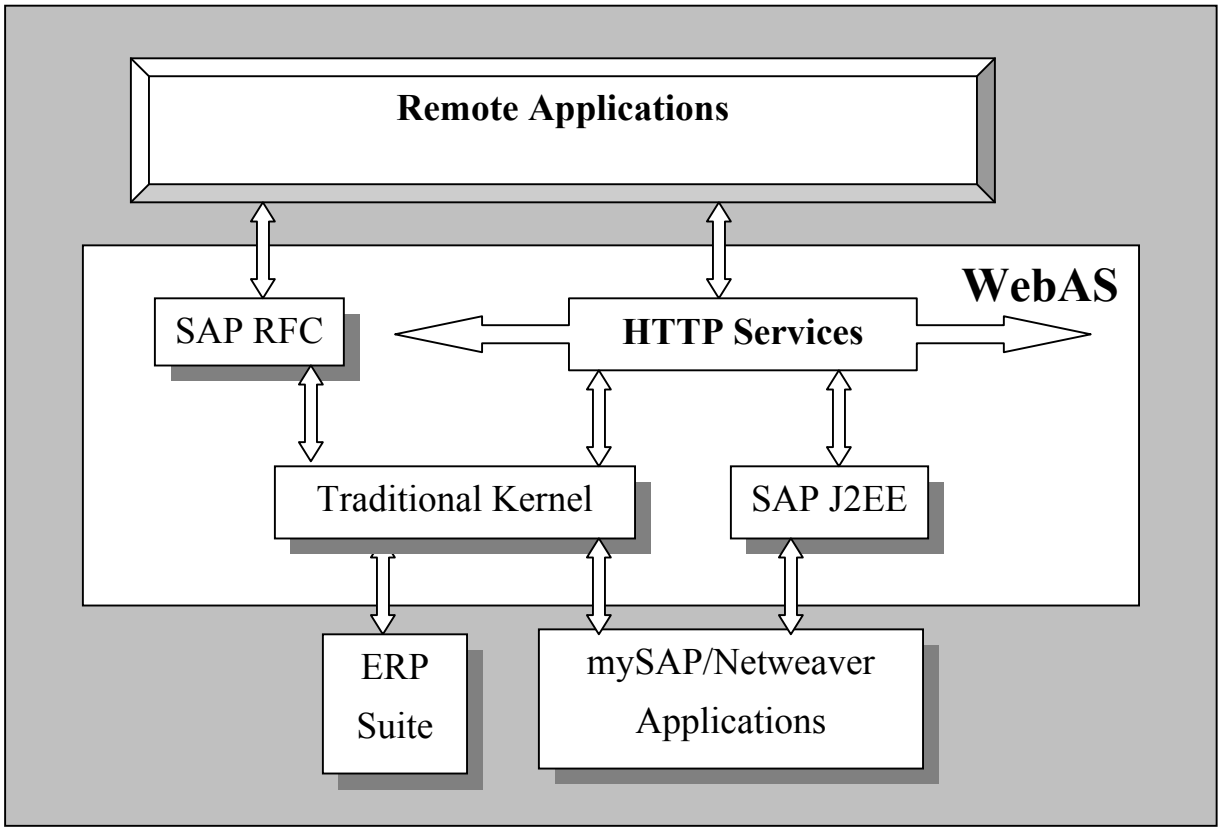

Figure 1: WebAS Integrates HTTP Support in the Traditional R/3 Kernel

WebAS

WebAS achieves two main goals: first, it allows the SAP ERP suite to be accessed via the ubiquitous HTTP protocol, and secondly, it establishes the SAP kernel as a professional development platform that allows for the development of Web-enabled applications in ABAP by relying on the proven and highly-scalable SAP infrastructure. As HTTP Web services are going to dominate distributed computing and client-server development, this new platform will be of essential importance to the future success of the SAP framework.

The current version of WebAS is SAP 6.1, which is identical with the one that underlies SAP Enterprise. There are already First Customer Shipments (FCS, in SAP's marketing Babylon now also known as *ramp-up customers*) for releases 6.2 and 6.3. The main evolutionary steps of these releases are:

- Release 6.2 adds the addition of the SAP J2EE engine (Java 2 Enterprise Edition) according to the standard defined by SUN Microsystems (http://www.java.sun.com/j2ee),

- Release 6.3 adds WebDynpro™, a new high-performance presentation server that easily creates browser-aware, interactive Web surfaces.

- Release 6.4 has also been announced, and will probably be the release that adds support for Microsoft.NET on Windows.NET platforms.

WebAS comes with three different usage targets

WebAS' new design and features present themselves in three main flavours of usage:

- WebAS as a development and runtime framework (also known as Virtual Machine)
  This is probably the most important use, and the most surprising novelty of the WebAS concept: it positions WebAS as an equivalent peer with the other two mainstream development frameworks: Java J2EE and Microsoft.NET.

- WebAS as HTTP server.
  Being a fully-equipped Web server, WebAS now plays in the same league as the other popular HTTP-based application server packages, such as *IBM WebSphere, Apache Tomcat, Microsoft.NET, Netscape Fasttrack, BEA, and Borland Enterprise Server,* to name a few.

- WebAS as enhanced kernel for R/3.
  WebAS also embeds the traditional and bulletproof R/3 kernel, with all its unique advantages like the Transport Management System with version control, load-balanced application server structure, OS-independent transaction control, and many more.

WebAS as a development and runtime framework

Despite its name, it has to be understood that WebAS is primarily a development and runtime platform. By adapting the HTTP protocol, the WebAS complies with the de facto industry standards in TCP/IP networking for message exchange, without forsaking any of the existing features of the SAP kernel. With the enormous amount of sophisticated development support that always existed in R/3, WebAS is hence designed to be a development framework for the creation of all kinds of high-performance, enterprise-wide applications. It targets the same area as J2EE and Microsoft.NET do.

WebAS executes its ABAP code on the ABAP runtime, which is simply a virtual machine. The ABAP runtime kernel that makes up this virtual machine has proven its power on many R/3 installations over the last ten years. The WebAS kernel is sizable, flexible, and stable under many different business contexts. The WebAS kernel has many innovative features, such as the compile-on-demand feature, which indeed checks code on demand for modifications, automatically compiling if the generated load is older than the source. This helpful WebAS feature is currently unique on the market. There are also that sort of build tools for Java and .NET, but they are not yet fully transparent to the user of the program and its developer as they are in the ABAP runtime.

In SAP 4.7, users are not required to use only WebAS tools for development. Programmers can still take advantage of mainstays like the ABAP Development Workbench. The ABAP

Development Workbench is one of the strongest interactive development environments (IDE) that can be found on the market. The ABAP debugger is already legendary, and allows easy and cost-effective maintenance of existing code. Other smart features of the ABAP Workbench, like the real-time cross-reference of code and data dictionary, automatic versioning and seamless integration into the transport system, are still features not often found in non-SAP environments.

WebAS as an HTTP Server

One interesting aspect of WebAS is the development of Web pages. In order to be a true content-driven HTTP server, WebAS has to provide a technology that allows for the production of dynamic page content. This can either be achieved

- by defining a Web service in SAP
- or by creating a Business Server Page (BSP) as an intelligent template for dynamic Web content in WebAS.

A Web service is basically a program that returns a text string as a result, while exposing a calling interface that is understood by a calling HTTP client. Practically speaking, a Web service in ABAP/Objects is defined by extending the ABAP object template class CL_HTTP_EXTENSION. The new class extension can then inspect the HTTP parameters that came along with the HTTP call to control the method's execution flow, and will provide a text string that is understood by the requestor as a result.

While a Web service via R/3 is a code-driven approach, a Business Server Page defines a template result page and inserts executable script code in appropriate places, therefore putting the design aspect of a Web page in the foreground.

Both strategies are well known from other Web servers. The new SAP Enterprise technology platform is a kind of "best-of," attempting to find a balance between resembling well-adopted standards and providing seamless integration into SAP's ABAP/Objects engine.

Two points about SAP HTTP and BSP to keep in mind:

- SAP HTTP services are called *servlets* in Java and .NET, which in turn copy the traditional CGI standard.
- BSPs are SAP's direct correspondence to Java Server Pages (JSP) and Active Server Pages (ASP)

SAP J2EE

In 2001, SAP's CEO Hasso Plattner caused some fury when he casually mentioned that SAP was going to support Java in future releases. A large part of the SAP community understood Hasso's remarks to mean: "SAP is going to drop ABAP and replace it with Java." In reality, Hasso never said this, and the idea has never even been considered by SAP's key decision makers. Instead, the plan was to support the new Java standard J2EE as an integral part of the SAP kernel. In future releases, it will be possible to make developments for the WebAS alternatively in the traditional SAP fashion using ABAP/Objects, or in Java, while abiding by the J2EE standard. Currently, the SAP J2EE only supports parts of the R/3 kernel, so development in J2EE on

WebAS is not yet equivalent to development in ABAP. Many important features like integration to R/3 messaging are still to be implemented.

J2EE is a mature version of the Java Virtual Machine (JVM). J2EE is a runtime and development platform for secure, reliable, transaction-based, Web-oriented software. As a platform, J2EE does primarily the same thing as the SAP kernel and the ABAP runtime do. It exposes services for transaction control, database access, and message-based communication. The J2EE platform supports component-based development, and so does SAP WebAS. J2EE technology and its component-based model simplify enterprise development and deployment for the Java world. The J2EE platform manages the infrastructure and supports the Web services to enable development of secure, robust, and interoperable business applications.

For ABAP developers, all the features and promises of J2EE/SAP compatibility must appear as the “Emperor’s New Clothes,” and indeed, SAP R/3 does not always support all of the features of J2EE. The main difference between the traditional SAP runtime and the J2EE platform is the programming language used. While SUN’s J2EE bases all developments on Java, the principle language of SAP is ABAP and ABAP Objects. Despite this difference, it has not proven too difficult for the SAP WebAS to support the J2EE platform as well as ABAP. Therefore, there are no obvious reasons why ABAP developers should even consider developing in Java on J2EE unless they want to develop applications for the WebAS and a non-WebAS platform like IBM’s WebSphere simultaneously. Otherwise, ABAP programmers can develop J2EE-compliant Web services from within SAP's WebAS environment.

SAP WebDynpro™

WebDynpro will be the major new feature of WebAS release 6.3. SAP already tends to market release 6.3 by the name WebDynpro only. The duty of WebDynpro will be to dynamically create HTML code based on the detected features of the requesting Web client or browser. WebDynpro will put a new meta layer between the WebAS HTTP platform and the browser, and is designed to free the WebAS developer from recurring and costly issues like adapting the generated HTML code according to a browser. See Figure 2.

With WebDynpro, the developer will only code a meta HTML page with as little code as possible. When the Web page is requested by a client, the meta page is processed and an actual proper HTML page is generated, depending on the capabilities of the requesting browser. Those familiar with Microsoft FrontPage™ or NetObjects Fusion™ know the basic idea from the concept of Web-bots. However, while FrontPage generates the pages and publishes them as static pages to a Web server, WebDynpro generates the result page in real-time, every time when a page is requested.

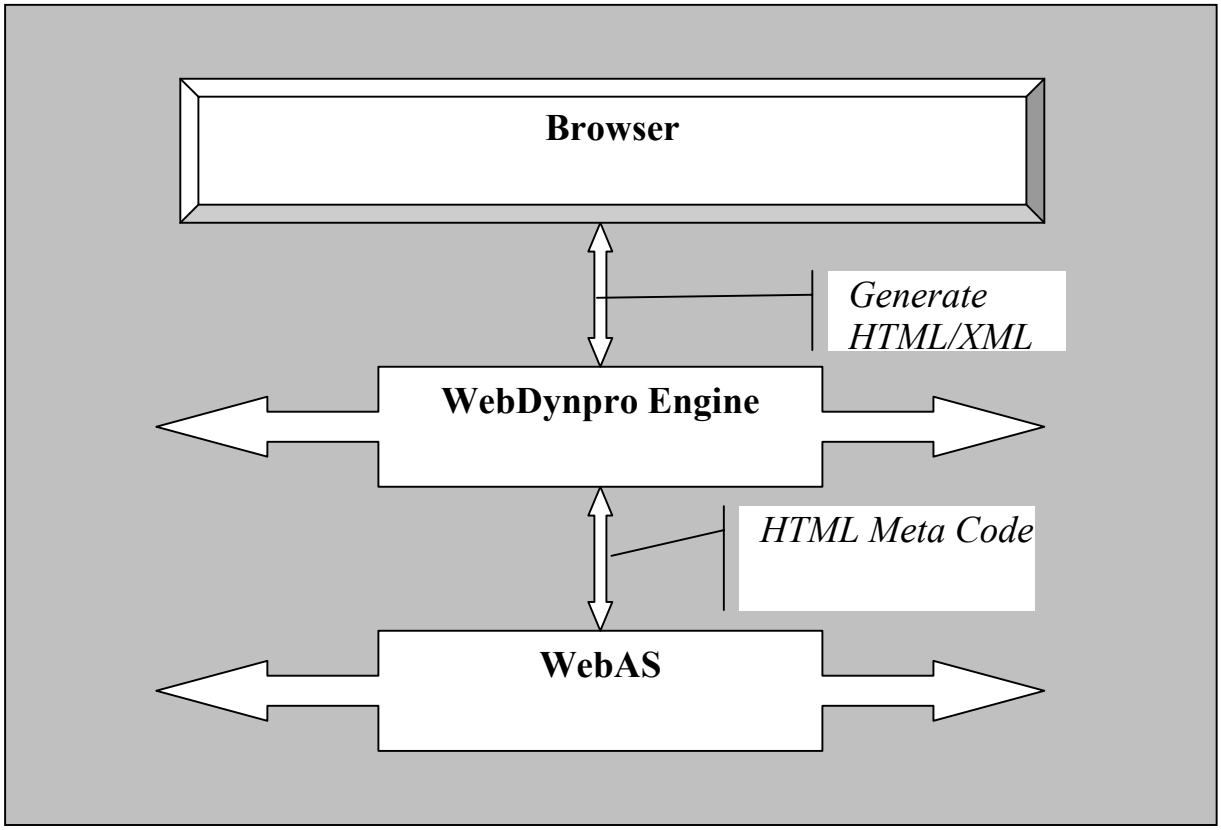

Figure 2: 3-Tier Layer Model of WebDynpro

WebAS 6.3 will probably be delivered in the middle of 2003 by the name WebDynpro™ 1.0. In its first release, WebDynpro will only support the J2EE engine. Plans are to support the ABAP runtime a little bit later. For those who are curious to see the idea behind WebDynpro, I recommend having a look on the CASABAC GUI Server http://www.casabac.com/.

The makers of WebDynpro have certainly been aware of CASABAC and may be inspired to some degree by the ideas of its inventor, former SAP engineer Bjoern Mueller.

The CASABAC GUI Server solution is a presentation server that has the same roots as WebDynpro, but CASABAC's developers looked beyond SAP, deciding to create a universal tool for more than the SAP framework only. Downloading the trial version from http://www.casabac.com/ will allow you to get an idea of where WebDynpro is headed: flicker-free page update, browser independent XML-based result pages, slim clients, data caching on the presentation server, and many more features.

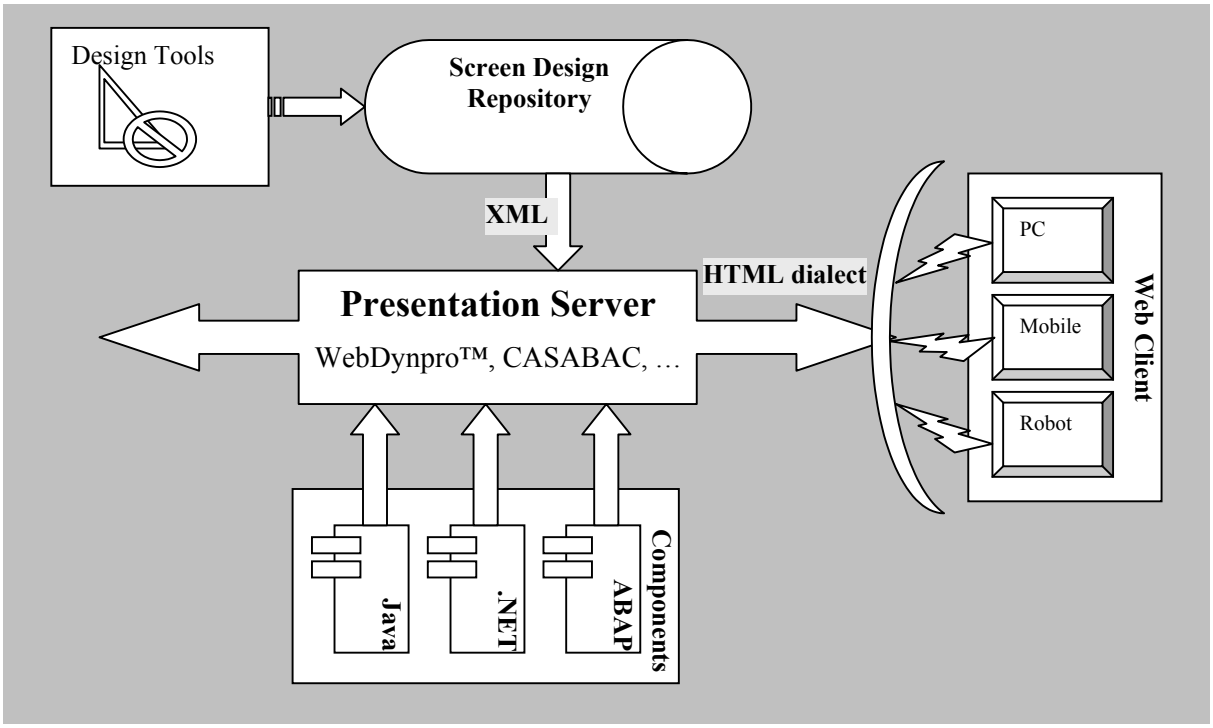

Figure 3: Work Principle of Presentation Server Like WebDynpro™ and CASABAC

Dynamic Server Pages

Before we discuss BSP in detail, we shall have a thorough look at how dynamic Web pages are realized in general. The common idea here is to dynamically construct the resulting Web page instead of storing a dump static page. This also allows reaction to parameters that are sent along with the received URI or in the body of the HTTP request. Typically, there are two kinds of dynamic Web pages:

- Template pages
  pages that define a result page with placeholder, which are filled when the page is actually requested

- Server pages
  pages which define templates for the result page and allow the insertion of script code that can be executed when the page is requested

Template Pages

Template pages are HTML pages with placeholders—pages that are replaced by dynamic content when the page is requested. This principle is used by SAP's Internet Transaction Server (ITS) and adopts the principle known by many mail merge programs. There is a program that determines the values of the placeholders when the page is requested. The found values are then merged into the template, and the resulting HTML page is returned. This is a simple approach and fine for many, browser-related solutions. The major drawback is that template pages still have a rigid layout that cannot be easily modified by the program.

Server Pages

Server pages - also known as active pages - are programs that produce an HTML or XML response page. Server pages are found in Microsoft's *Internet Information Server (IIS)* as *Active Server Pages (ASP),* in Java/J2EE-compliant Web servers as *Java Server Pages (JSP),* and in SAP's *Web Application Server (WebAS)* as *Business Server Pages (BSP).*
Server pages are similar to template pages, with the essential difference being that program language statements are inserted instead of placeholders. Server pages present a rough schema of the result page and allow the insertion of sections of programming code that serve the dynamic content. Most Web servers support server pages in a similar fashion. Because the programming can include conditional statements, the layout of the result page can also be controlled by selecting different template sections, via a properly positioned IF … THEN … ELSE section.

Servlets and CGI Scripts

Servlets are pure programs that return the response page as a string. The full result page must be constructed by the called program or method. This means that the program has full control over the output, but it also needs to know how the result pages must be properly prepared. Especially when dealing with plain HTTP output, it can be pretty tedious to construct an HTML output by means of string concatenation. Helper classes that allow constructing an XML or HTML document tree are available in many flavours to assist this work.

Nevertheless, servlets are especially good for the development of non-HTTP solutions, for example Web services, where the expected result is not necessarily browser-viewable code, because the caller is not necessarily a browser, but could be another program, for example, a database query. Common Gateway Interface (CGI) scripts are a historical form of servlets and refer to executable programs that are called according to a strict calling convention.

Create Your Own BSP

After all the introductory theory, let us now explore the principle of Dynamic Server Pages and, how SAP's BSPs work. This should give you a jumpstart to create your own BSPs right away.

A BSP adopts the same coding and design model that is already known from many other Web servers. Such dynamic Web pages are key features of any serious Web server. A BSP can be characterized as follows:

- BSPs are very close to Java Server Pages and Active Server Pages.

  JSP and ASP have positioned themselves as de facto standards in template-based Web design, so it is not a surprise that SAP adopts their syntax and working principle, which is indeed simple enough to be called straightforward.

- BSPs can contain code either in ABAP/Objects or in JavaScript.

  Please keep in mind that BSPs support server-side JavaScript—not Java! If you choose ABAP, and I see no reason why you would want to use JavaScript instead, you are still limited to the somewhat restricted ABAP Object design, compared to plain ABAP. So have a read through ABAP Object's limitations in SAP's help pages before you get started.

- BSPs are executed on top of the R/3 kernel.

  Every BSP is actually an ABAP program and executes on top of the R/3 kernel. Therefore, all features of the SAP kernel which are accessible through ABAP Objects are also within the reach of a BSP, including transaction control and message or workflow creation.

- BSPs allow seamless RFC calls.

  Through the HTTP interface, it is extremely easy to call external applications from a BSP, and BSPs are easily called from other applications as a Web service.

- BSPs can make seamless use of the SAP J2EE (release 6.2 and later) engine.

  In release 6.2, the BSP will be able to execute programs written in Java on top of the new SAP J2EE engine. However, the J2EE will run as a sidecar to the SAP kernel, thus not making full use of the SAP kernel services.

- BSPs provide an easy mechanism to internationalize and translate pages.

  There is a new Online Translation Repository introduced with BSP. With this, you can mark up keywords during Web design and translate them by means of the SAP translation repository (transaction SE61).

- BSPs allow an event-driven page design.

  BSPs are event-driven, which facilitates the design of complex and interactive BSP applications. The events allow dedicated code sections that are executed only when a certain event is fired. This avoids the need to poll the current state of a request by the program itself, and shifts these recurring tasks to the dispatcher.

A "Hello World" BSP

For our first example, we shall define a simple HTML page in the WebAS that outputs a “Hello World” page without any dynamic content. A Web page in WebAS is created with the well-known ABAP Workbench in transaction SE80. The Workbench defines a BSP application, which can contain one or more Business Server Pages. We then start defining our Web pages in a manner similar to creating function modules for a function group.

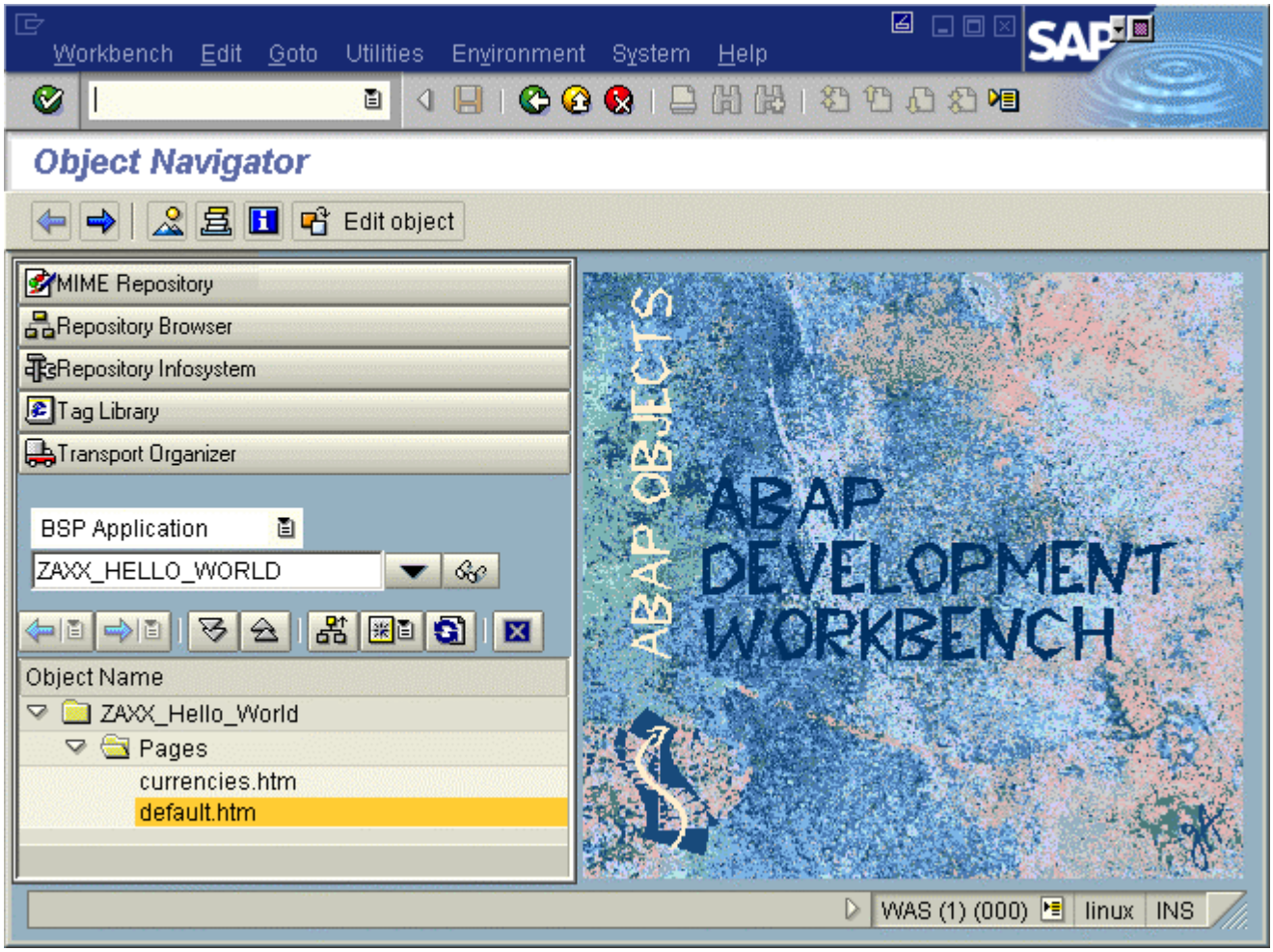

Figure 4: Create a BSP Application as Container for a Group of Business Server Pages

The “Hello World” example contains no dynamic code. In Figure 4, we display its greeting and a link to another BSP page with the name “currencies.htm”.In Figure 5 we have illustrated the “back end” of the “Hello World” BSP creation. Notice there is no major coding in this example.

```
<%@page language="abap"%>
<html>
  <head>
    <link rel="stylesheet"
        href="../../sap/public/bc/bsp/styles/sapbsp.css">
    <title> Hello World BSP Page </title>
  </head>

  <body class="bspBody1">
    <H1>Hello World from Your BSP</H1>
    Show me current exchange rates with
    <a href="currencies.htm">
    BAPI BAPI_EXCHRATE_GETCURRENTRATES</a>
  </body>
</html>
```

Figure 5: Hello World BSP

The BSP is then called like any other Web page by entering its URL in the browser. The shown path assignment can be customized, and the BSP can be organized in an arbitrary tree-like structure below the BSP root directory. Like with other Web servers, you can also define alias names for subdirectories, so that you won’t have to specify the full physical path all the time. The port number of the WebAS can also be set to any meaningful value during kernel setup. In Figure 6, you can see the abbreviated version of the URL appearing in a developer's property page for SE80. The full URL of a BSP is displayed in the Properties tab strip below in Figure 7.

```
http://192.168.69.111:8080/sap/bc/bsp/sap/ZAXX_HELLO_WORLD/default.htm
```

Figure 6: URL to Call the BSP via HTTP

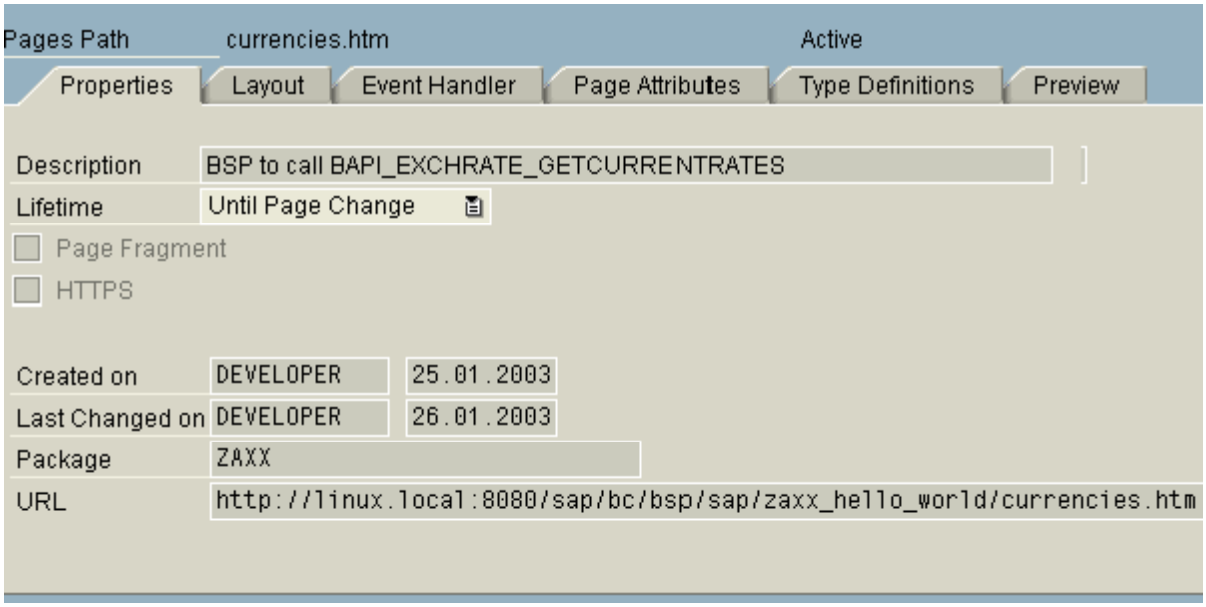

Figure 7: Properties Defined in SE80 for the BSP Currencies.htm

Calling a BAPI from a BSP

In our next example, we will define a BSP that actually contains programming code. The page "currencies.htm" will call the BAPI FUNCTION "BAPI_EXCHRATE_GETCURRENTRATES." This BAPI reads the actual currency exchange rates from R/3. The resulting table is then looped over and every line of the table is output. The ABAP program code is inserted within a matching pair of brackets of the form <% … %>. The programming language between the brackets depends on what the Web server understands. The only supported languages for BSP are currently ABAP and server-side JavaScript. To view the programming code for our example, see Figure 8. To view our successful output, see Figure 9 below.

```
<%@page language="abap"%>
<html>
  <head>
    <link rel="stylesheet"
        href="../../sap/public/bc/bsp/styles/sapbsp.css">
    <title> Hello World BSP Page </title>
<%
DATA: exch_rate_list type bapi1093_0.
DATA: tfrom_curr_range TYPE STANDARD TABLE OF bapi1093_3.
DATA: tto_currncy_range TYPE STANDARD TABLE OF bapi1093_4.
DATA: texch_rate_list TYPE STANDARD TABLE OF bapi1093_0.
DATA: treturn TYPE STANDARD TABLE OF bapiret1.

CALL FUNCTION 'BAPI_EXCHRATE_GETCURRENTRATES'
  DESTINATION 'NONE'
  EXPORTING
    date                   = sy-datum
  TABLES
    from_curr_range        = tfrom_curr_range
    to_currncy_range       = tto_currncy_range
    exch_rate_list         = texch_rate_list
    return                 = treturn.
%>
  </head>

  <body class="bspBody1">
    <H1><otr>Listing of exchange rates from
    BAPI_EXCHRATE_GETCURRENTRATES
    through BSP</otr></H1>
     <table border="1" width = "100%">
        <tr>
        <th width="20%">From Currency</td>
        <th width="20%">To Currency</td>
        <th width="60%">Exchange Rate</td>
        </tr>
        <% LOOP AT texch_rate_list into exch_rate_list. %>
          <tr>
             <td > <%= exch_rate_list-from_curr %> </td>
             <td > <%= exch_rate_list-to_currncy %> </td>
             <td > <%= exch_rate_list-exch_rate %> </td>
          </tr>
        <% ENDLOOP. %>
     </table>
  </body>
</html>
```

Figure 8: BSP Calling a BAPI from ABAP Personality

From Currency	To Currency	Exchange Rate
AED	JPY	30.55026
AED	USD	0.27241
ARS	BRL	1.20900
ARS	CAD	1.52630
ARS	CLP	471.66000
ARS	COP	1548.00000
ARS	JPY	112.15000
ARS	MXN	9.82810
ARS	PEN	3.16130
ARS	USD	1.00000
ARS	VEB	564.88000
AUD	JPY	69.36541
ZAR	JPY	19.06762
ZAR	USD	0.17002

Figure 9: Sample result page for BAPI_EXCHRATE_GETCURRENTRATES

Localization of Texts with the Online Text Repository

A special goodie is presented by the WebAS when it comes to translating the texts found on a BSP into a different language. Most Web servers do not support localization, with the consequence that the strings and labels of the Web page must be inserted as variables, and the variables in turn have to be determined by the driving program.

WebAS parses the BSP and stores all strings, along with the references to it, in the Online Text Repository (OTR). When the BSP is to be displayed in a different language, the WebAS consults the OTR and displays the translated strings at the proper place. This mechanism is an enormous facilitation of the development work. Using the OTR, you can avoid a situation where every developer has to code his/her own determination of the language texts. This allows the developer to design the BSP in a natural WYSIWYG manner in the language of choice by the developer.

Translation of the OTR texts is handled in the same way as other translations in R/3, via the transaction SE63. This demonstrates another advantage of the OTR, as it takes the tedious translation task away from the development team, letting it be done by a natural speaker and language professional of the target language.

How to Develop Web Services in WebAS
Part 1: Making Sense of Business Server Pages

Summary

With WebAS, which is available stand-alone and as part of the SAP 4.7 Enterprise Edition, the SAP technology has matured to a fully Web-compliant application server. Business Server Pages close the gap between the traditional ERP world, with its proprietary user interfaces, and the modern world of browser-based Web applications. The strength of BSP is the underlying traditional SAP kernel that allows for the development of Web-compliant applications, while relying on the approved, stable, transaction-based SAP application server technology. With BSP, you design HTML or XML templates, and insert ABAP object code in places where dynamic content is required. The inserted ABAP code allows for the calling of an arbitrary ABAP statement or functioning module within the integrated ERP system. All in all, using BSP is the easiest way to create browser or “Web services compatible” dynamic Web pages that can be called via HTTP.

In part two of this white paper, to be published in conjunction with the August 2003 edition of SAPtips, we will discuss the principles and usage of the WebAS HTTP extensions, which allow for the development of servlets on the WebAS and the creation of arbitrary Web services.

Axel Angeli is a senior SAP and EAI advisor and principal of logosworld.com, a German-based enterprise specializing in coaching SAP and EAI project teams and advising IT management on implementation issues. Axel has been in the IT business since 1984, and throughout his career, he has always worked with cutting-edge technologies. Axel's SAP experience stems back from the good old R/2 days, and he is an expert on SAP’s Netweaver technology and any kind of ABAP development. A speaker of several languages, Axel specializes in coaching and leading large multinational teams on complex projects with heterogeneous platforms and international rollouts. Known for his intensive and successful trouble-shooting experience, Axel has been nicknamed by his colleagues as the “Red Adair” of SAP projects. He is the author of the best-selling tutorial “The SAP R/3 Guide to EDI, IDocs, ALE and Interfaces."