"SAPping" Teknoda

Notas técnicas de SAP / ABAP - Tip en detalle Nro. 12

(Lo nuevo, lo escondido, o simplemente lo de siempre pero bien explicado)

Los nuevos escenarios de programación con SAP Netweaver (serie de varios tips)

" El modelo BSP (II): Un caso práctico con JavaScript"

Tema: Páginas Web, JavaScript, Netweaver, SAP Web AS, BSP, JAVA, Internet .

Descripción: El objetivo de esta serie de tips es **recorrer y ejemplificar el nuevo escenario de desarrollo que SAP ofrece a partir de Netweaver**. En este tip, explicamos los fundamentos de la codificación de BSP's con JavaScript, y explicamos paso a paso cómo llevar a cabo un ejemplo.

Nivel: Avanzado

Versión: 6.1, 6.2, 6.3

Fecha pub: Julio 2004

Próximos Tips:
El modelo BSP (III): Explotando BAPI's desde una BSP.
El modelo BSP (IV): Las extensiones BSP y el Model View Controller.
SAP como JAVA Server. Netweaver Development Studio.
Integración de JAVA y SAP a través del Java Connector.
Entendiendo las Web Dynpro. Cómo construir una Web Dynpro.
Web Services

NOTA: Para una correcta comprensión de este artículo, recomendamos la lectura de los tips anteriores "**Entendiendo el SAP Web Application Server desde el punto de vista del programador" y "El modelo BSP: Cómo construir las primeras BSP".** Consulte nuestro sitio www.teknoda.com/tipssap.htm para accederlos.

Tabla de contenido

III. Ciclos que se usan en Abap y sus equivalentes en JavaScript.

IV. Creación de una *BSP* Application con una página dinámica que despliega datos de una tabla perteneciente a un sistema SAP R/3.

V. Anexo: Conceptos adicionales.
Alcances de JavaScript.
Historia de JavaScript.

VII. Dónde obtener información adicional.

I. Introducción / Resumen Ejecutivo

Cómo viéramos en detalle en tips anteriores, a partir de la versión 6.1, **SAP sustituye el SAP BASIS System por el SAP Web Application Server.**

El SAP WAS, al igual que el BASIS que lo precede**, provee el cimiento tecnológico sobre el cual se apoya el R/3 Enterprise**, el sucesor de R/3, y la mayoría de los componentes de mySAP.com. En este rol, el SAP WAS funciona como **el nuevo R/3 Kernel**, implementando toda la arquitectura de conectividad, seguridad, acceso a bases de datos, y también la interfaz de programación.

Concebido para actuar en un mundo dominado por las aplicaciones "Web enabled", la principal innovación del SAP WAS es que **trae la funcionalidad Web completamente embebida**, tanto desde el punto de vista de los protocolos de comunicaciones como del servicio de aplicaciones. En su personalidad "Nativa", es decir, como kernel del R/3, el SAP WAS continúa soportando el universo ABAP de versiones anteriores, pero introduce nuevas herramientas para desarrollo de aplicaciones Web fuertemente integradas a R/3. Desde la primera versión del Web AS, (6.1) aparece el modelo de **Business Server Pages (BSP's)** como una potente opción para construir aplicaciones Web que accedan dinámicamente a los datos de SAP.

Paralelamente, el SAP WAS funciona como un servidor de aplicaciones JAVA (como Websphere, BEA Web Logic, etc.) , conformando enteramente el estándar J2EE. Se dice a menudo que el SAP Web AS es una plataforma de "doble personalidad" porque funciona como un Kernel para SAP R/3, brindando servicios como Web Application Server "nativo", y a la vez, es capaz de actuar como un servidor de aplicaciones JAVA.

En este tip continuamos explorando la personalidad nativa del WAS, y la implementación de BSP's (Business Server Pages) embebida dentro del R/3. Recomendamos la lectura de los tips anteriores "**Entendiendo el SAP Web Application Server desde el punto de vista del programador" y "El modelo BSP: Cómo construir las primeras BSP".** Consulte nuestro sitio www.teknoda.com/tipssap.htm para accederlos.

II. Entendiendo JavaScript y su inserción en el mundo SAP

Qué es JavaScript

Al igual que la mayoría de los lenguajes de "scripting", JavaScript es un lenguaje de programación concebido para crear **pequeñas porciones de código,** embebidas dentro del HTML, para implementar determinadas acciones, siempre en el ámbito de una página web. El código JavaScript va "intercalado" en los documentos HTML. Con JavaScript, se puede, por ejemplo, crear efectos especiales en las páginas, definir interactividades con el usuario, operar sobre strings o tablas, mostrar mensajes, etc.

En sus principios, el "scripting" dentro del HTML era resuelto por los navegadores. El browser se ocupaba de aislar e interpretar las instrucciones JavaScript embebidas en el HTML. Esta modalidad, aún vigente, se denomina *"Client-side Scripting"*. Casi todos los navegadores modernos soportan "scripting" en uno o más lenguajes, siendo JavaScript el más común.

Posteriormente, apareció también la posibilidad de que el código en JavaScript pueda ser derivado para su interpretación y ejecución del **lado del Web Server**, aumentando la confiabilidad y eficiencia de este recurso. Tal es el caso del SAP WAS 6.10, capaz de procesar los scripts en JavaScript y ABAP. Cuando realizamos una BSP embebida con código JavaScript o ABAP se interpreta en el SAP WAS 6.10 .

BSP 's y JavaScript

Las Business Server Pages (BSP's) son una de las formas en que pueden definirse páginas Web dentro del Web Application Server SAP, similares al concepto de JSP o ASP. Consisten en código HTML que define los componentes estáticos de la página, con código embebido ("server-side scripting") para que el server gestione los elementos dinámicos, tomados de R/3 u otras fuentes. Por supuesto, la primera opción para lenguaje de scripting es utilizar ABAP, lo que provee acceso directo a todas las componentes de R/3. Pero para aquellos desarrolladores no familiarizados con ABAP, o para los diseñadores de páginas Web, las BSP's soportan también la opción de "server-side scripting" en JavaScript.

El uso de JavaScript dentro de las BSP es básicamente un tema de preferencias sintácticas, dado que tanto en ABAP o en JavaScript, las estructuras de datos subyacentes son variables ABAP (campos, estructuras, tablas internas).

No debe confundirse el JavaScripting de las BSP's con la personalidad JAVA del Web AS; (ver detalle en apartado siguiente)**.** El uso de JavaScript en las BSP forma parte de la personalidad "nativa" del WAS, que viene dotado de un intérprete JavaScript integrado (Mozilla, JS 1.5). (Ver nota al final sobre cómo provee SAP el soporte de JavaScript). Tampoco debe confundirse el uso de JavaScript en las BSP's con el denominado "Client-Side Scripting". Este último no es resuelto por el servidor sino simplemente por el Browser.

¿ Porqué JavaScript en lugar de ABAP ?

Después de la enorme conquista que representan las BSP's para la comunidad SAP, en el sentido que nos permiten codificar Web Applications en el conocido y confortable ABAP, ¿ por qué habría alguien de reemplazarlo por JavaScript ?.

En muchos casos el desarrollo de páginas Web queda en manos de diseñadores, compañías externas especializadas en esta tarea, más que en los programadores de aplicaciones tradicionales. JavaScript es un lenguaje ampliamente conocido por diseñadores de páginas Web, y puede resultar una mejor opción para hacer comprensible la lógica de la aplicación.

En otros casos, pueden aparecer ventajas en las funcionalidades ofrecidas por JavaScript, difíciles de implementar en ABAP. Por ejemplo, desde JavaScript es muy fácil invocar otros lenguajes como Perl, por ejemplo. También hay tareas como procesar tablas en paralelo, o manejar "strings" que resultan muy laboriosas en ABAP, y bastante más simples en JavaScript.

De todos modos, una Web Application no tiene porqué resolverse usando un único lenguaje de Scripting. BSP's basadas en ABAP perfectamente pueden convivir con otras basadas en JavaScript.

Diferencias entre Java y JavaScript

JavaScript no guarda una relación directa con Java, aunque es comúnmente asociado por la semejanza en la denominación. Desde el punto de vista de sus arquitecturas, son bien distintos, y no guardan entre sí un vínculo mayor que la sintaxis idéntica .

A continuación revisamos las diferencias entre ambos lenguajes.

- ✓ **Compilador:** para programar en Java se necesita un kit de desarrollo (JDK) y un compilador. Sin embargo, JavaScript no es un lenguaje que necesite que sus programas se compilen, sino que éstos se interpretan por parte del navegador cuando éste lee la página o se interpretan del lado del server (por Ejemplo: SAP WAS 6.10).

- ✓ **Orientado a objetos:** Java es una lenguaje orientado a objetos. JavaScript no lo es.

- ✓ **Propósito:** Java es mucho más potente que JavaScript, esto es debido a que Java no es sólo lenguaje de propósito general sino que también es una plataforma, con la que se pueden hacer variadas aplicaciones. Sin embargo, con JavaScript sólo podemos escribir programas para que se ejecuten en páginas web.

- ✓ **Estructuras fuertes:** Java es un lenguaje de programación fuertemente tipificado, esto quiere decir que al declarar una variable tendremos que indicar su tipo y no podrá cambiar de un tipo a otro automáticamente. Por su parte JavaScript no tiene esta característica, y podemos almacenar en una variable la información que deseemos, independientemente del tipo de ésta. Además, se puede cambiar el tipo de información de una variable en el momento que se necesite.

Como vemos, Java Script no pretende ni necesita ser tan potente, robusto y seguro como Java, dado que está pensado con otro propósito. Justamente, por esta razón, debido a sus características, WAS 6.30 implementa la plataforma JAVA equiparándola con la plataforma ABAP. El WAS 6.30 posee una JVM (Java Virtual Machine) y una AVM (Abap Virtual Machine).

BSP con JavaScript vs. JSP

Una de las más grandes diferencias entre JSP Y BSP con JavaScript, es que las JSP's justamente, se apoyan sobre la plataforma JAVA, con todas las implicancias del entorno JAVA que se describieron en la sección anterior. Las JSP's requieren de la personalidad J2EE del SAP Web AS, y no pueden resolverse dentro del contexto de R/3.

Funcionalidades	BSP con JavaScript	JSP
Manejo de variables	Las BSP's manejan variables definidas en el "type definitions ", que pueden ser gestionadas fácilmente utilizando JavaScript. Por ejemplo: para obtener la fecha del sistema, existen 2 formas : uno es declarar una variable de tipo **sy –datum** en el "type definitions", que puede ser invocada en el componente layout dentro de una sentencia escrita en JavaScript o directamente escribir en el layout: **< % = sy .datum >**	Las JSP's permiten a través del uso de expresiones del tipo **<% = expresión de java %>** acceder a paquetes de clases propias de java. Estos paquetes de clases permiten acceder a las funcionalidades que ofrece java para manejo de cadenas, acceso de archivos, utilización de fechas ,etc. En una JSP la obtención de la hora y la fecha del sistema se haría utilizando los paquetes de clase propios de java. El paquete sería *java.util.date*. Por lo tanto en una JSP se escribiría: **<% = new java.util.date()>**
Manejo de requerimientos y respuestas	Las BSP deben : manejar los requerimientos y respuestas de http , las sesiones, guardar el nombre de la página , el contexto, etc. Para estas necesidades, utiliza objetos definidos en ABAP Object. Estos objetos se manejan en la componente Event Handler y se codifican en ABAP. Los ABAP Object más utilizados son: *request, runtime, response, navigation, event id, pages, messages, application*, entre otros.	Las JSP's al igual que las BSP manejan requerimientos , respuestas de http, sesiones, almacenamiento del contexto. Tienen ocho variables predefinidas. Estas son: *request , response, out , session , application, config , pageContext* y *page* . La coincidencia en los nombres de los objetos no significa que tengan exactamente los mismos métodos o iguales funcionalidades. Por ejemplo: para el manejo de la sesión, en el caso de las JSP's, utiliza directamente un objeto *session* mientras que las BSP manejan el objeto *runtime* que contiene el método *session_id.*
Etiquetas	Las BSP's permiten la creación propia de etiquetas en la versión del WAS 6.20 (BSP Extensions).	Las JSP's tienen su propia generación de etiquetas.
Redireccionamiento de página y pasaje de parámetros	El redireccionamiento o *forward* de la JSP, lo hace la BSP a través del ABAP Object Navigation con el método *goto_page*. El pasaje de parámetros se realiza de forma similar a las JSP.	Las JSP's tienen atributos propios como: *set property* para el manejo de los beans; *param* para pasar parámetros; *include* para incluir archivos (*include* también existe en las BSP's en forma de directiva); y *forward* que redirecciona la petición del usuario.

III. Ciclos que se usan en Abap y sus equivalentes en JavaScript

Antes de comenzar con el ejemplo práctico, reunimos en esta tabla la sintaxis de los ciclos más comúnmente utilizados. Nuevamente se demuestra que la elección entre los 2 lenguajes (ABAP y JavaScript) depende de una cuestión de gusto sintáctico. Sin embargo debemos saber los alcances y particularidades que presenta JavaScript (se verá detalladamente en la sección **V. Conceptos adicionales**).

Ciclo	JavaScript	ABAP
Implementación de ciclos WHILE	<%**do{**%> <% sentencia 1**;** %> <% sentencia 2**;** %> <% sentencia n**;** %> <%**}while(**condición**)** %> Ejecuta las sentencias 1 a n veces (según se cumpla la condición dada). <%**while(**condición**){** %> <% sentencia 1**;** %> <% sentencia 2**;** %> <% sentencia n**;** %> <%**}**%> Puede ejecutar las setencias 0 o n veces (según se cumpla la condición dada).	<%**WHILE** variable **IS INITIAL**%> <% sentencia 1 %> <% sentencia 2 %> <% sentencia n %> **<%ENDWHILE** %>
Recorrido de tablas	<%**For(**int i =0;i < tabla1.length; i++**){** %> <% = tabla1[i].campo_a %> <% = tabla1[i].campo_b %> <% = tabla1[i].campo_c %> <% **}** %>	<%**LOOP at** tabla **into** tabla1%> <% = tabla1-campo_a %> <% = tabla1-campo_b %> <% = tabla1-campo_c %> <%**END LOOP**%>
Implementación de ciclos FOR	<%**For(**int i =0;i < 10; i++**){** %> <% sentencia 1 **;** %> <% sentencia 2 **;** %> <% sentencia n **;** %> <% **}** %>	<%**DO** 10 **TIMES** %> <% sentencia 1 %> <% sentencia 2 %> <% sentencia n %> <%**ENDDO** %>

Sentencia IF	<% **if (**condición**){** %> <% **}** %>	<% **IF (**condición**) .**%> <% **ENDIF.**%>
Implementación de sentencias CASE	**switch (**variable**){** **case** opción1 **:** sentencia**;** **break;** **case** opción2 **:** sentencia**;** **break;** default **:** **}**	**CASE** variable**.** **WHEN** 'opción1'**.** sentencia **WHEN** 'opción2'**.** sentencia **ENDCASE**.

IV. Creación de una BSP Application con una página dinámica que despliega datos de una tabla perteneciente a un sistema SAP R/3

Para construir y testear las BSP's, se trabaja con un nuevo componente del ABAP Workbench denominado **Web Application Builder**, accesible desde la transacción **SE80**. El Web Application Builder es una colección integrada de herramientas que incluye, entre otras cosas, un BSP editor y un debugger para los scripts ABAP y JAVA.

Para definir una BSP es necesario definir primero una BSP Application que la contenga. Una BSP Application es un conjunto de páginas HTML. Las BSP's y las BSP's Applications forman parte del respositorio de SAP.

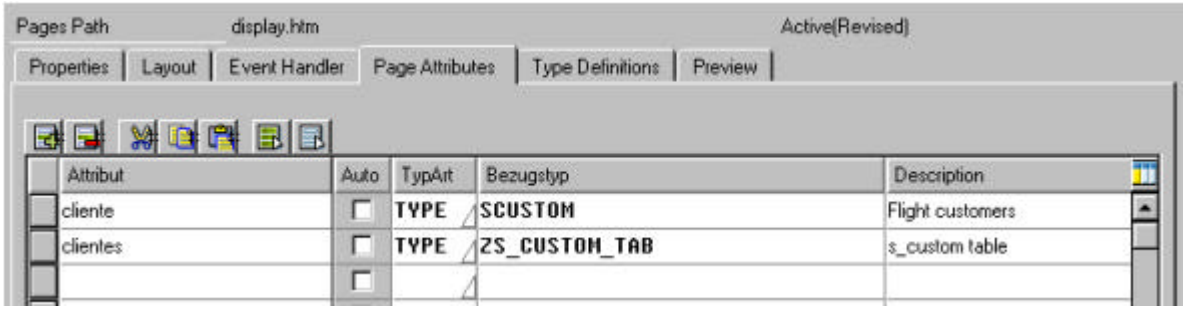
En el siguiente ejemplo, se muestra cómo se despliega en un Web Browser el contenido de la tabla SCUSTOM (contiene datos de clientes) perteneciente al sistema SAP R/3. El objetivo es recuperar y usar sus datos en una BSP, utilizando ABAP. La BSP a crear trabajará con sólo 3 componentes: el *Layout*, el *Page Attribute* (almacena los datos de los clientes) y el *Event Handler* (para realizar la consulta y poder obtener los datos a desplegar).

Para empezar, crear una BSP Application con nombre **"z_bsp_desplegar_scustom"** y luego una página con extensión **.htm**. Activar la BSP Application y la nueva página. (*Recordar los pasos a seguir para la creación de la BSP Application, ya detallados en el tip anterior)***.** Una vez completados estos pasos, continuar con los siguientes:

1. Seleccionar el tab **Page Attributes** en la ventana del editor.
2. Colocar en la columna de **Attribute** el nuevo parámetro llamado "*clientes"* (observar la "s" final).
3. Seleccionar en la columna de **TypArt** la opción *" TYPE"*.
4. Colocar en la columna de **Associated Type** el tipo de dato ABAP *"zs_custom_tab"*, el cual es una estructura de la tabla SCUSTOM. Esta estructura debe ser creada previamente en el *data dictionary*. El parámetro "*clientes"* será usado para almacenar la lista de clientes recuperada desde la tabla SCUSTOM durante el runtime.

Adición de un segundo parámetro desde **Page Attributes**:

5. Completar el segundo renglón con los siguientes datos: **Attribute**: *"cliente"* (sin "s"), **TypArt**: "TYPE" y **Associated Type**: "SCUSTOM" (tipo de dato ABAP, es una tabla standard de SAP). La siguiente imagen muestra los dos atributos ya incorporados:



6. Seleccionar el tab **Layout** en la ventana del editor.

7. Verificar la existencia de la directiva < %@page language="abap"%> como primera en el layout. Esta directiva indica que se trabaja con ABAP.

8. Incorporar en el layout el código HTML como se muestra en la siguiente imagen:

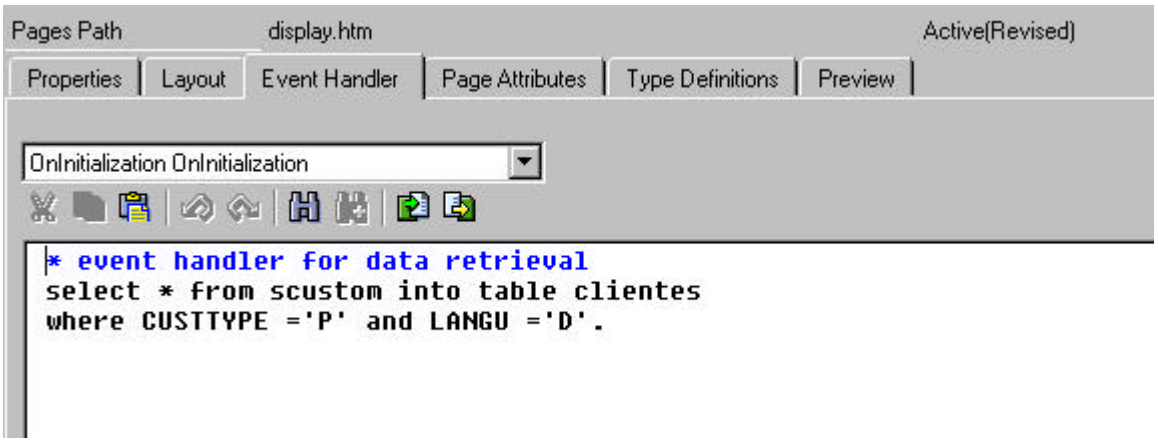
Layout

```
<%@page language="abap"%>
<html>
 <head><title> Fligth connection </title>
 </head>
<h2>Esta es una tabla extraída del R/3</h2>
 <body>
 <table border = 1>
 <tr><th>id</th>
    <th>name</th>
    <th>form</th>
    <th>street</th>
    <th>city</th>
    <th>country</th>
    <th>region</th>
 <% for (i =0 ; i<clientes.length;i++){ %>
 </tr><td><% =clientes[i].id %></td>
 <td><% =clientes[i].name %></td>
 <td><% =clientes[i].form %></td>
 <td><% =clientes[i].street %></td>
 <td><% =clientes[i].city %></td>
 <td><% =clientes[i].country %></td>
 <td><% =clientes[i].region %></td>
 <% } %>
 </table>
 </body>
</html>
```

Recordar que los datos contenidos en la tabla interna "*clientes*" necesitan ser "loopeados" y mostrados registro por registro.

Adición de **Event handlers**:

9. Seleccionar el tab **Event Handlers**.
10. Seleccionar de la lista desplegable "**On Inicialization**".
11. Copiar la siguiente consulta :



12. Guardar las tareas realizadas.
13. Activar la BSP Application.
14. Ejecutar la BSP Application. Se desplegará automáticamente la página en el Web Browser:

id	name	form	street	city	country	region
00000002	Andreas Lotz	Herr	Pimpinellenweg 9	Walldorf	DE	
00000003	Hans Tillinger	Herr	Veilchenstraße 12	Heidelberg	DE	
00000004	Bindewald	Frau	Kaiserstr. 55	Heidelberg	DE	
00000010	Raupp	Herr	Glockenblumenweg 5	Zuerich	CH	
00000011	Ballmann	Herr	Ballmanns Leiden 7	Heidelberg	DE	
00000012	Radetzky	Frau	Mozartstr. 102	Wien	A	
00000014	Genee	Herr	Albrecht-Duerer-streetse 4	Koeln	DE	

Pueden seguirse los mismos pasos anteriores para crear una BSP que defina un formulario de entrada de datos. En este caso, el Event Handler a utilizar sería **OnInputProcessing.**

V. Conceptos adicionales

Alcances de JavaScript

Como sabemos las tablas internas de datos ABAP permite almacenar y manejar gran cantidad de objetos del mismo tipo. ABAP soporta 3 formas básicas de tablas externas: standard , sorted y hashed (donde los datos son accedidos a través de un código hash). Sin embargo, JavaScript sólo maneja las tablas en forma de *array*, por lo tanto es razonable que las tablas tipo *sort*ed ó *hashed* no se puedan "bindear" a JavaScript.

Otros de los problemas que sucede con los Objetos ABAP es que el identificador "machee" con una palabra reservada en JavaScript tal como *byte* or *null*.

La solución a este problema es conocer las pocas palabras reservadas que tiene JavaScript. En el siguiente link se encuentra información útil sobre Javascript
http://devedge.netscape.com/library/manuals/2000/javascript/1.5/reference/keywords.html.

Entre las utilidades que brinda el Object Navigator (SE80), existe la posibilidad de crear y ejecutar programas en JavaScript **desde cualquier programa escrito en ABAP**, a través de llamados a métodos de la clase **CL_JAVA_SCRIPT**. Además, en este contexto, los programas JavaScript pueden ser depurados.

Por otro lado, debido a que los scripts de Java se almacenan en formato Bytecode, sólo necesitan ser compilados una vez. De esta forma se obtiene una reducción de tiempo en las próximas ejecuciones, ya que no es necesario volverlos a compilar.

Historia de JavaScript

Como ya sabemos, HTML no resulta suficiente para realizar todas las acciones que pueden llegar a necesitarse en una página web. Esto es debido a que a medida que fue creciendo la web y sus distintos usos, las páginas se fueron complicando y también las acciones que se querían realizar a través de ellas.

El primer ayudante para cubrir las necesidades que estaban surgiendo fue Java, a través de la tecnología de las *applets* (pequeñas aplicaciones que se incluyen en las páginas web y que pueden realizar las acciones asociadas a la Aplicación Web). La programación de applets fue un gran avance y Netscape (por aquel entonces el navegador más popular), había roto la primera barrera del HTML al hacer posible la programación dentro de las páginas web. No cabe duda que la aparición de las applets supuso un gran avance en la historia de la web, pero no ha sido una tecnología definitiva y muchas otras herramientas han seguido implementando el camino que comenzó con aquellas applets.

Netscape, después de hacer sus navegadores compatibles con los applets, comenzó a desarrollar un lenguaje de programación al que llamó LiveScript, el cual permitió crear pequeñas aplicaciones dentro de las páginas Web y fue mucho más sencillo de utilizar que Java. De modo que el primer JavaScript se llamo LiveScript. Este nombre no duró mucho, pués antes de lanzar la primera versión del producto se forjó una alianza con Sun MicroSystems, creador de Java, para desarrollar en conjunto un nuevo lenguaje.

La alianza hizo que JavaScript se diseñara como una ramificación de Java, solamente útil dentro de las páginas web y mucho más fácil de utilizar, de modo que cualquier persona, sin conocimientos de programación pudiese adentrarse en el lenguaje y utilizarlo sin mayores complicaciones. Además, para programar JavaScript no es necesario que instalemos un kit de desarrollo (JDK), ni compilar los scripts, ni realizarlos en archivos externos al código HTML, como ocurría con los applets.

VI. Dónde obtener información adicional

SAP Developers Network www.sdn.sap.com

SAP NetWeaver - http://www.sap.com/solutions/netweaver/

SAP NetWeaver Software – http://service.sap.com/netweaver

IMPORTANTE

Dudas o consultas: ***sapping@teknoda.com***