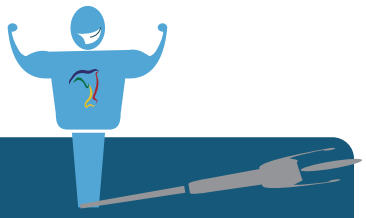
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

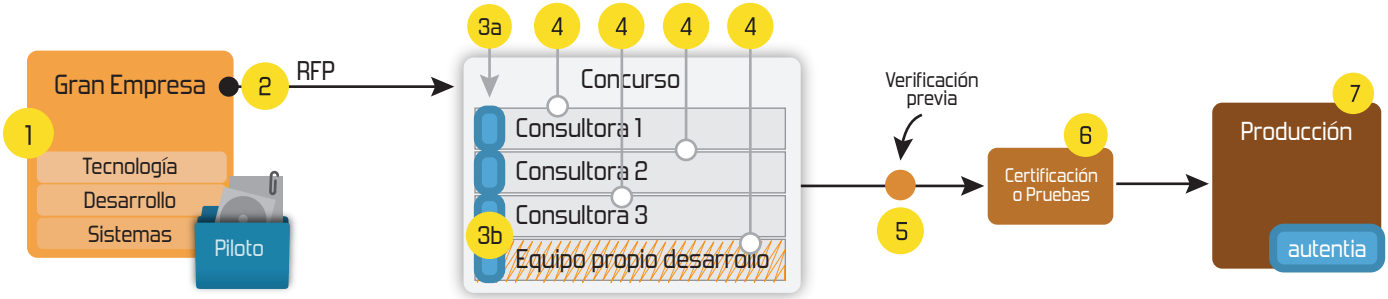
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Tutorial desarrollado por:
Roberto Canales Mora 2003-2005
Creador de AdictosAlTrabajo.com y

Director General de Autentia S.L.

Uso de XML y XSL en JSPs

En uno de nuestro anteriores tutoriales, os mostramos como, utilizando JSP 2.0, podemos combinar JSP y XSL para formatear documentos XML en el servidor (aunque también vimos que se podría hacer en el cliente, aunque no os lo aconsejo).

No siempre es posible (ni conveniente) utilizar lo último de lo último. Si os fijáis en las grandes organizaciones, normalmente estandarizan una versión de Java, JSP, EJB, etc... y hasta que no pasa un tiempo y se consolidan las tecnologías (y aparecen parches) no se cambia de versión.

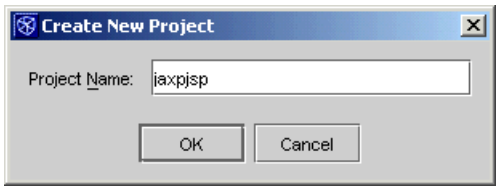
Entonces, es posible que tengáis que apañaros para hacer lo mismo ..... con versiones anteriores de JSP.

Veréis que es bastante sencillo y os vamos a mostrar como se hace paso a paso.... utilizando el patrón MVC.

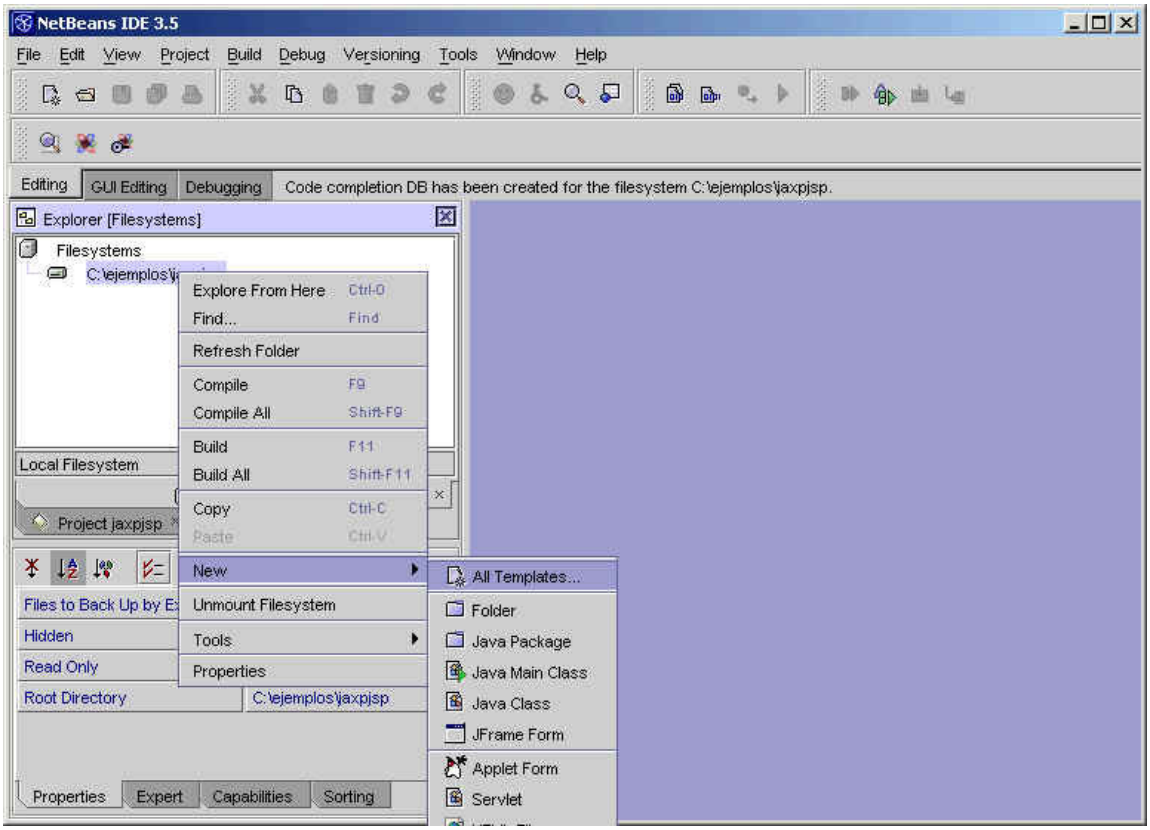
Pondremos los ejemplos con NetBeans. Mucha gente me pregunta por qué lo uso..... La razón es sencilla, me parece muy intuitivo... aunque hay otras opciones más potentes.

Crear el Proyecto

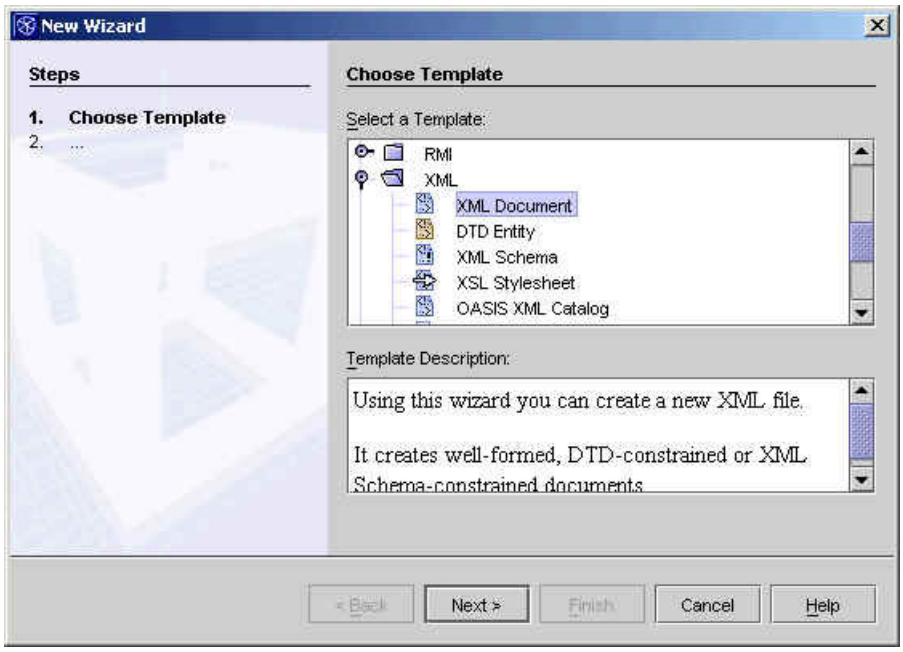
Creamos en NetBeans un proyecto



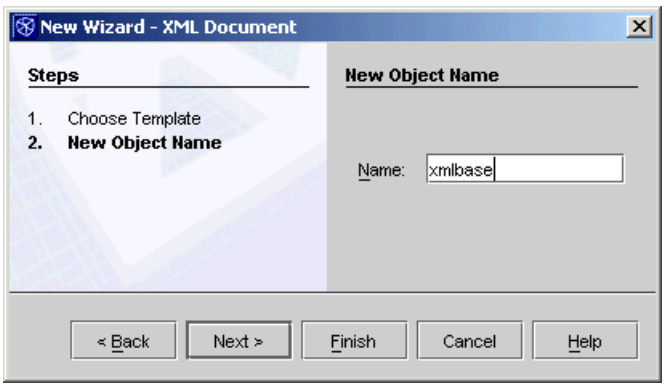
Vamos a crear unos documento XML y XSL con los generadores de código.



Seleccionamos dentro del grupo XML



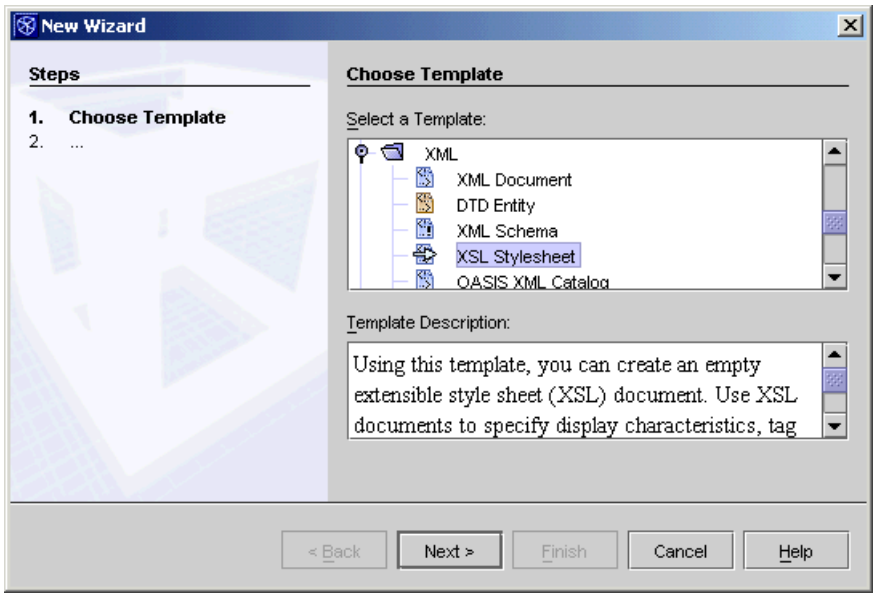
Asignamos un nombre



Vamos a utilizar este documento como base:

```
<?xml version="1.0" encoding="UTF-8"?>
<tutoriales>
  <tutorial>
    <autor>rcanales@autentia.com</autor>
      <nombre>JSP 2.0</nombre>
      <enlace>jspel</enlace>
      <descripcion>Nuevas caracteristicas de JSPs 2.0</descripcion>
  </tutorial>
  <tutorial>
      <autor>alejandropg@autentia.com</autor>
      <nombre>Struts y Eclipse</nombre>
      <enlace>struts</enlace>
      <descripcion>Configuración del entorno Struts en
Eclipse</descripcion>
    </tutorial>
</tutoriales>
```

Repetimos pero ahora seleccionamos un documento XSL



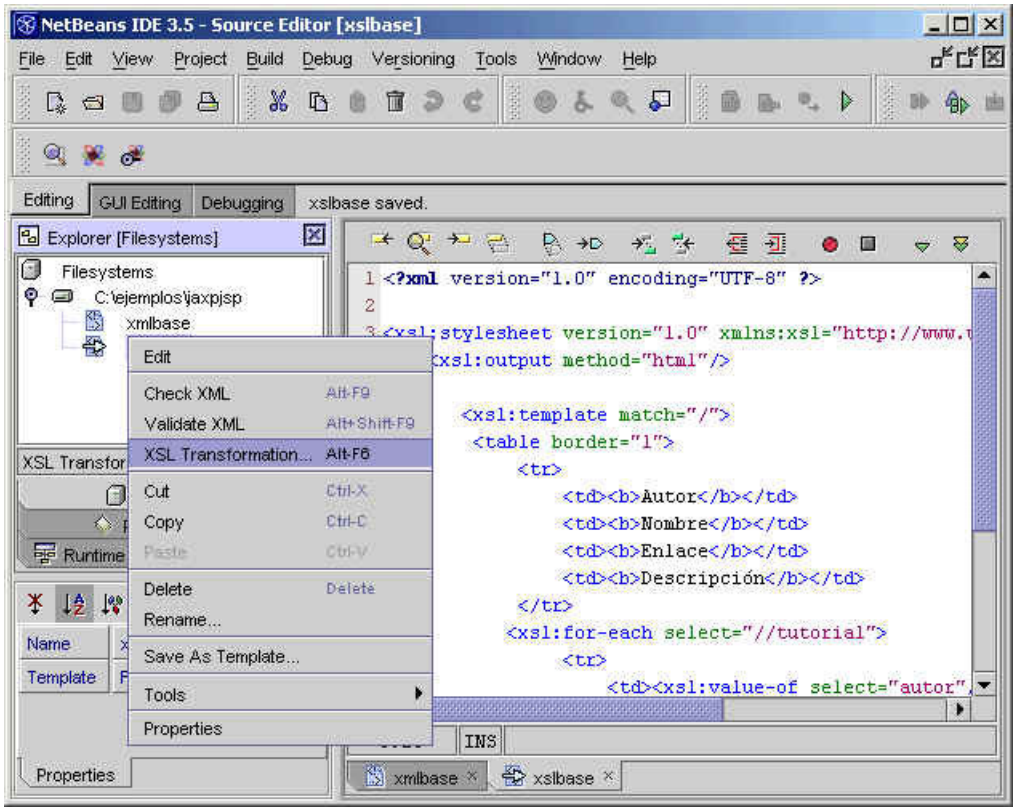
Escribimos nuestro XSL

```
<?xml version="1.0" encoding="UTF-8" ?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>

  <xsl:template match="/">
    <table border="1">
    <tr>
      <td><b>Autor</b></td>
      <td><b>Nombre</b></td>
      <td><b>Enlace</b></td>
      <td><b>Descripción</b></td>
    </tr>
    <xsl:for-each select="//tutorial">
      <tr>
        <td><xsl:value-of select="autor"/></td>
        <td><xsl:value-of select="nombre"/></td>
        <td><xsl:value-of select="enlace"/></td>
        <td><xsl:value-of select="descripcion"/></td>
      </tr>
    </xsl:for-each>
    </table>
  </xsl:template>
</xsl:stylesheet>
```

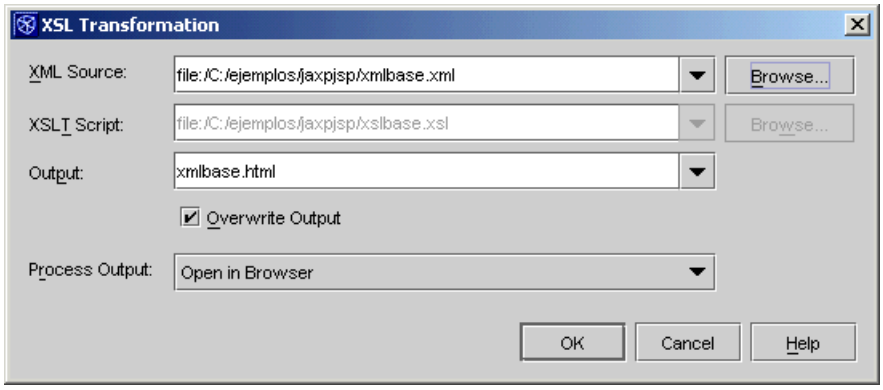
Probar la transformación en NetBeans

Vamos a usar las características de NetBeans para probar como quedaría....

Seleccionamos, pinchando el botón derecho sobre el XML o XSL

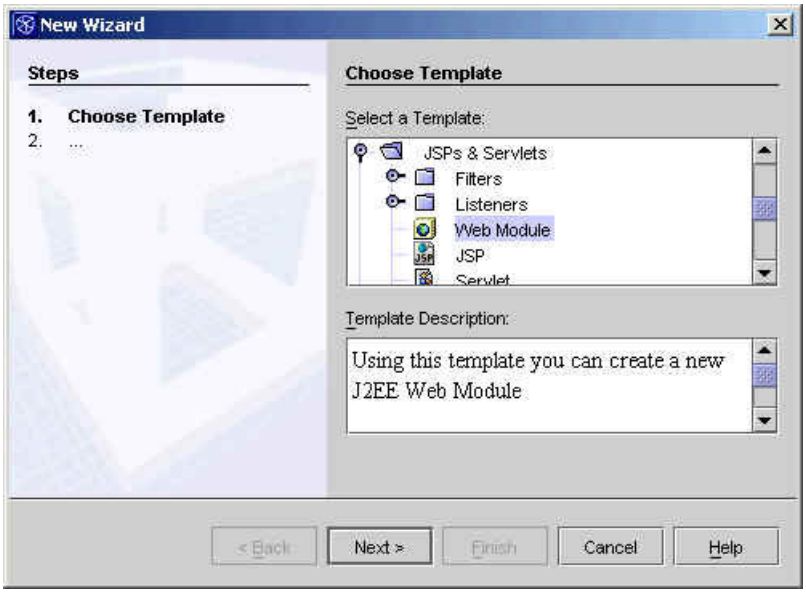


Seleccionamos los ficheros



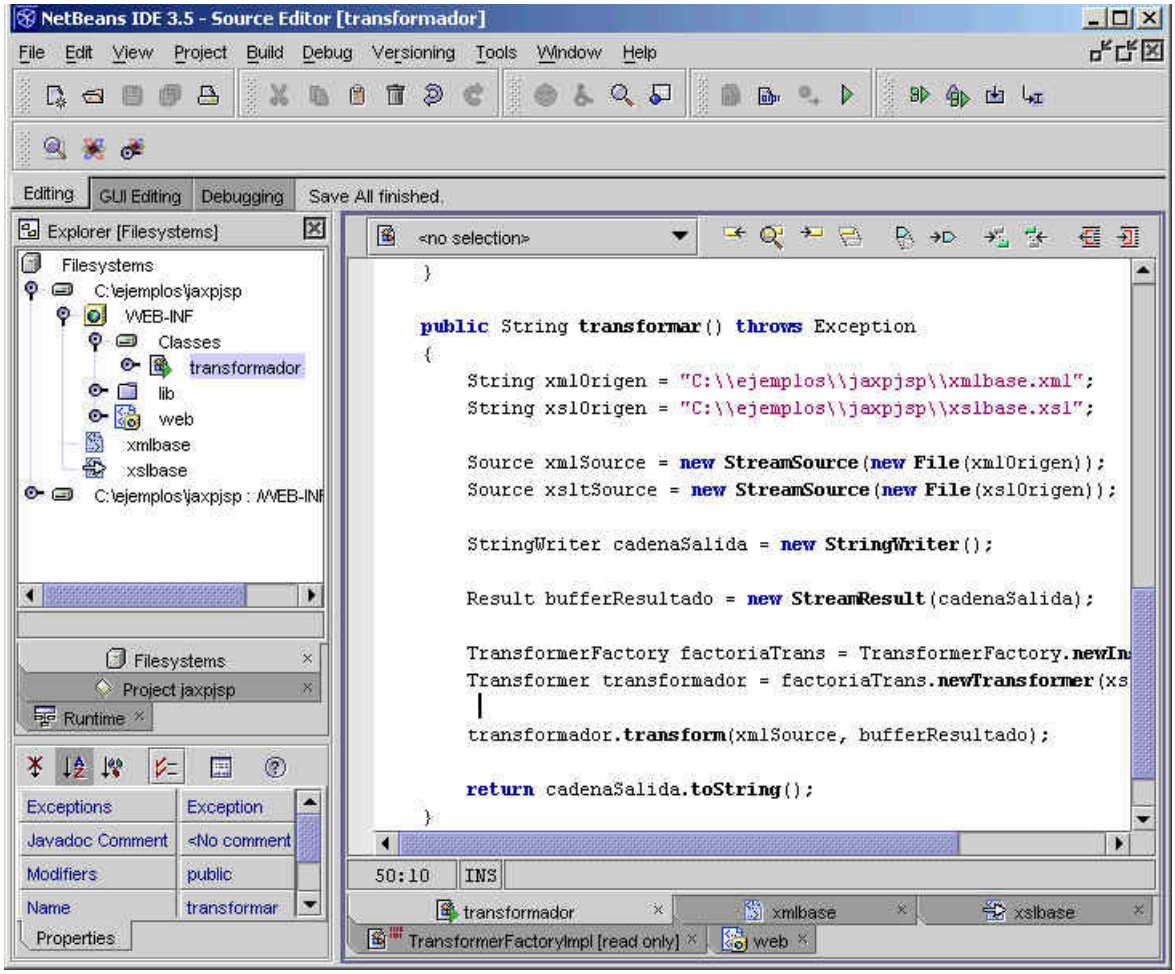
Y vemos el resultado

Activamos nuestro directorio como una WebApp



Introducir el código Java

Ahora, vamos a escribir una clase Java que sea capaz de hacer una transformación y retornárnosla como una cadena.



Escribir la clase Java de prueba

En este caso, vamos a utilizar un interfaz para motores de transformación denominado JAXP (ver documentación en SUN)

No tenemos que incluir nada extraordinario por la versión de Java que estamos usando.

```
import java.io.*;
import javax.xml.transform.*;
import javax.xml.transform.sax.*;
import javax.xml.transform.stream.*;
```

```
import org.xml.sax.*;
/**
 *
 * @author  Roberto Canales
 */
public class transformador
{

   void depura (String pCadena)
   {
      System.out.println("Mensaje: " + pCadena);
   }


   public static void main(String [] args) {
      transformador p = new transformador();

      try
      {
         p.depura("Comenzamos transformación");
         p.depura(p.transformar());
         p.depura("Terminamos");
      }
      catch(Exception e)
      {
         p.depura("Errores en aplicación");
         e.printStackTrace();
      }
   }

   public String transformar() throws Exception
   {
      String xmlOrigen = "C:\\ejemplos\\jaxpjsp\\xmlbase.xml";
      String xslOrigen = "C:\\ejemplos\\jaxpjsp\\xslbase.xsl";

      Source xmlSource = new StreamSource(new File(xmlOrigen));
      Source xsltSource = new StreamSource(new File(xslOrigen));

      StringWriter cadenaSalida = new StringWriter();

      Result bufferResultado = new StreamResult(cadenaSalida);

      TransformerFactory factoriaTrans = TransformerFactory.newInstance();
      Transformer transformador = factoriaTrans.newTransformer(xsltSource);

      transformador.transform(xmlSource, bufferResultado);

      return cadenaSalida.toString();
   }
}
```

Si ejecutamos este código, veremos en la pantalla de salida:

```
Mensaje: Comenzamos transformación
Mensaje: <table border="1">
<tr>
<td><b>Autor</b></td><td><b>Nombre</b></td><td><b>Enlace</b></td><td><b>Descripci&oacute;n</b></td>
</tr>
<tr>
<td>rcanales@autentia.com</td><td>JSP 2.0</td><td>jspel</td><td>Nuevas caracteristicas de JSPs 2.0</td>
</tr>
<tr>
<td>alejandropg@autentia.com</td><td>Struts y Eclipse</td><td>struts</td><td>Configuraci&oacute;n del entorno
Structs en Eclipse</td>
</tr>
</table>
Mensaje: Terminamos
```

Es decir, funciona.....

Crear el MVC

Ahora, vamos a crear un servlet, que generará un XML y lo pasará (en MVC) a un JSP. Vamos a introducir directamente el XML en una cadena de caracteres.... dando por supuesto que se obtendría de otro modo (por ejemplo usando una Base de Datos XML)

Este JSP, puede ejecutar el código de la transformación con distintas técnicas:

- Scriplet (Código Java en el JSP)
- Un Java Bean
- Un TAG de usuario
- Usando JSTL de JSP 2.0

Los distintos métodos de comunicación entre estos elementos (Servlets, Beans, JSPs y Tags los podéis ver en otro de nuestros tutoriales)

Os mostramos como hacerlo con un JSP con Scriplet... el resto de los métodos es muy sencillo (seguid los otros tutoriales).

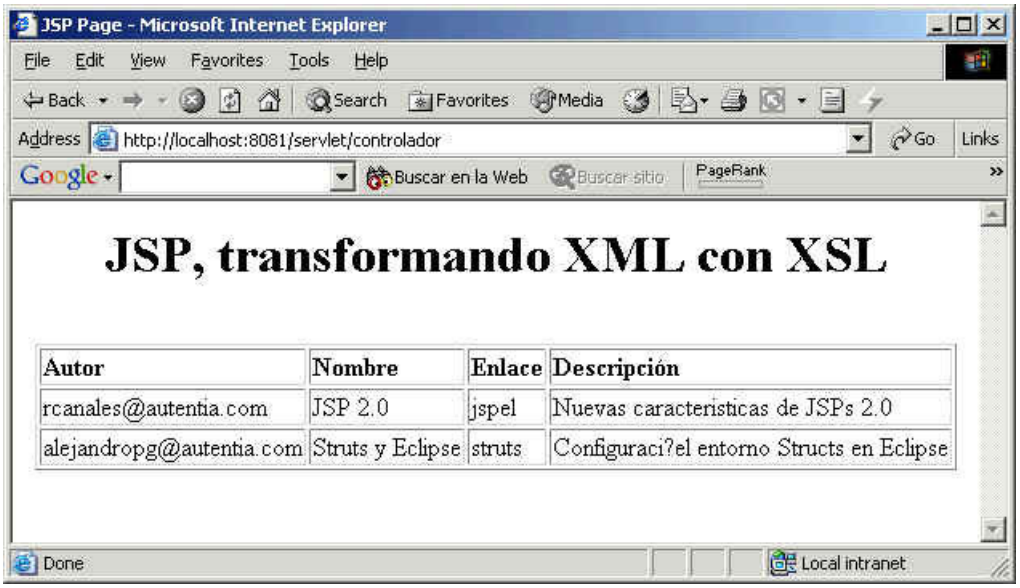
El servlet

```
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class controlador extends HttpServlet {

   protected void doGet(HttpServletRequest request, HttpServletResponse response)
   throws ServletException, IOException
   {
      try
      {
         String elXML = "<tutoriales><tutorial><autor>rcanales@autentia.com</autor>

<nombre>JSP 2.0</nombre><enlace>jspel</enlace><descripcion>Nuevas caracteristicas de JSPs 2.0</descripcion>

</tutorial><tutorial><autor>alejandropg@autentia.com</autor><nombre>Struts y Eclipse</nombre>

<enlace>struts</enlace><descripcion>Configuración del entorno Structs en Eclipse</descripcion>

</tutorial></tutoriales>";


         request.setAttribute ("xml",elXML);
         getServletConfig().getServletContext().getRequestDispatcher("/presentacion.jsp").forward(request, response);
      }
      catch (Exception ex)
      {
         ex.printStackTrace ();
      }
   }
}
```

EL JSP

```
<%@page contentType="text/html"
import="java.io.*,javax.xml.transform.*,javax.xml.transform.sax.*,javax.xml.transform.stream.*,org.xml.sax.*"%
>
<html>
<head><title>JSP Page</title></head>
<body>

<center>
<H1>JSP, transformando XML con XSL</h1>
<br>

<%
   String xmlOrigen = (String)request.getAttribute("xml");
   String xslOrigen = "C:\\ejemplos\\jaxpjsp\\xslbase.xsl";

   Source xmlSource = new StreamSource(new StringBufferInputStream(xmlOrigen));
   Source xsltSource = new StreamSource(new File(xslOrigen));

   StringWriter cadenaSalida = new StringWriter();

   Result bufferResultado = new StreamResult(cadenaSalida);

   TransformerFactory factoriaTrans = TransformerFactory.newInstance();
   Transformer transformador = factoriaTrans.newTransformer(xsltSource);

   transformador.transform(xmlSource, bufferResultado);
   out.print(cadenaSalida.toString());

%>
</center>

</body>
</html>
```

La salida es:



Bueno, ya sabéis como hacerlo ..... y con Struts.... sería igual de sencillo.

Como podéis comprobar, cada día podemos reutilizar y combinar mejor los conocimiento de nuestros tutoriales

Sobre el Autor ..

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	Enviar

Otros Tutoriales Recomendados (También ver todos)

Nombre Corto	Descripción
Reingeniería JDO con Druid	Os mostramos como crear vuestras clases y descriptores JDO, de tablas existentes, con la herramienta gratuita Druid.
Ejecutar JSPs almacenados en Base de Datos	Atendiendo una pregunta de nuestro foro, os mostramos como, con unos sencillos pasos, podemos ejecutar JSPs almacenandos en la base de datos. Esto puede ser una idea base para un gestor de contenidos construido en Java.
JDO con OJB	Os mostramos como configurar el entorno OJB de apache para construir la primera aplicación JDO
Cachear porciones de JSPs	En este tutorial os ensañamos como incrementar increiblemente el rendimiento de vuestro Web basado en tecnología JSP con el FrameWork de cache OSCACHE
JSP´s y Modelo-Vista-Controlador	En este tutorial os enseñamos como crear un JSP, su relación con los servlets y como crear un ejemplo MVC en Tomcat
Otra implementación JDO con TJDO	Os mostramos como montar un ejemplo simple de JDO, a través de la implementación gratuita TJDO
Gestión errores en JSPs	Os mostramos como realizar ciertas labores intermedias en JSPs: Comentarios, gestión de errores, formateo de fechas y precompilación de ficheros
Comunicación entre TAGs, Beans y JSPs	Os mostramos las posibilidades de comunicación entre JSPs, Bean y etiquetas de usuario.
Struts Jakarta	Cuando se ha trabajado creando aplicaciones Java poco a poco se va viendo la necesidad de normalizar los desarrollo. Uno de los Framework (entornos) más extendidos es Struts
JSP 2.0, JSTL y Lenguaje de	Os mostramos las novedades de JSP 2.0: Nuevas librerías estandar de etiquetas y el

Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores.

En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo.

Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.