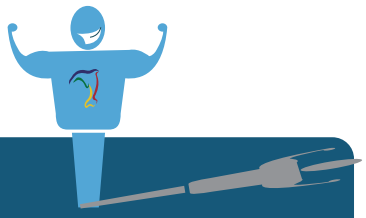
autentia

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...

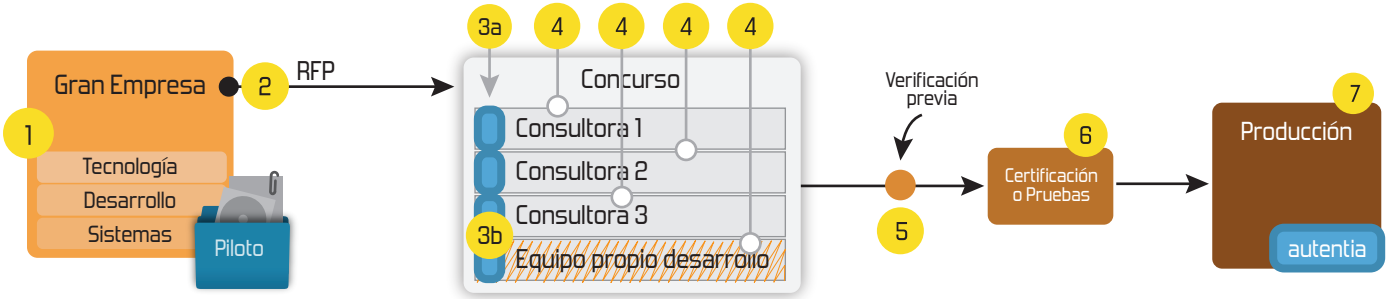
1. Desarrollo de componentes y proyectos a medida



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: Alejandro Perez García 2003-2004.

Si te gusta lo que ves, **puedes contratarme** para impartir **cursos presenciales** en tu empresa o para ayudarte en proyectos (Madrid).

Contacta: alejandropg@autentia.com.

Descargar este documento en formato PDF strutsxdoclet.pdf

Como manejar Struts con Xdoclet

1. Introducción

Struts es un proyecto de apache que podemos encontrar en http://jakarta.apache.org/struts/index.html

Es un framework de desarrollo de aplicaciones Web. Este framework se basa en el patron MVC (Modelo Vista Controlador), con lo que nos permite separar perfectamente al lógica de negocio de la lógica de presentación, de la lógica de control.

Se configura a partir del fichero struts-config.xml. En este fichero es donde describimos, entre otras cosas, los formularios de nuestra aplicación, así como las acciones que pueden invocarse.

Los formularios corresponden con una clase. Típicamente un POJO, VO, o Bean, es decir una clase que sólo tiene atributos y métodos get y set. Las acciones corresponden con clases que extienden de org.apache.struts.action.Action. Sólo con esto ya podemos ver que el trabajo de mantenimiento va a ser doble: por un lado tendremos que mantener las clases, y por otro lado tendremos que mantener el fichero de configuración struts-config.xml.

Además del problema del mantenimiento tenemos otro problema añadido: el fichero struts-config.xml suele ser único (podemos dividirlo en módulos, pero el problema sólo se atenúa) con lo que su mantenimiento va a resultar engorroso incluso con un sistem de control de versiones (como CVS) ya que todos los desarrolladores tendrán que tocar el mismo fichero.

Para solucionar estos problemas podemos usar Xdoclet. Xdoclet es una herramienta que nos permite poner una serie de tags en l comentarios de nuestras clases y métodos, de forma que al correr Xdoclet se parsean estos tags y se genera algún tipo de salida. En nuestro caso, pondremos comentarios en las clases de acción y las clases de formulario y obtendremos de forma automática e fichero struts-config.xml y el fichero validation.xml (es donde definimos las validaciones automáticas de los formularios).

Esto es sólo un ejemplo de las múltiples cosas que se pueden hacer con Xdoclet. os recomiendo que os paséis por su página http://xdoclet.sourceforge.net/xdoclet/index.html para echar un vistazo a todo su potencial.

2. Preparando Xdoclet para ejecutarlo con ant

Ant (http://ant.apache.org/) es una herramienta al estilo del make de C. Es decir nos va a permitir automatizar procesos de compilación, despliegue, copiado de ficheros, ... La forma ideal de usar Xdoclet es integrándolo dentro de nuestros scripts ant par automatizar todo el proceso.

Dentro de nuestro fichero build.xml (por defecto es el fichero que ejecuta ant) tendremos que crear un nuevo target:

```
<target name="webdoclet" depends="init">
   <taskdef name="webdoclet" classname="xdoclet.modules.web.WebDocletTask">
      <classpath>
         <path refid="xdoclet.classpath" />
      </classpath>
   </taskdef>

   <!-- Run the Ant build file with a parameter "-Dxdoclet.force=true" to force regeneration of files. -->
   <webdoclet destdir="${build.web.dir}/WEB-INF"
            force="${xdoclet.force}"
            mergedir="metadata">
      <fileset dir="src" />

      <strutsconfigxml version="1.1"
```

```
            xmlencoding="ISO-8859-1"
            validateXML="true"
            templateFile="metadata/struts/struts_config_xml.xdt"
            mergeDir="metadata/struts" />

        <strutsvalidationxml />
    </webdoclet>
</target>
```

Como se ve en el ejemplo, primero se define la tarea "webdoclet" con taskdef. En este punto es importante destacar el uso de "xdoclet.classpath", esta es una referencia al path donde están todos los .jar necesarios para que Xdoclet funcione.

Después ejecutamos la tarea que acabamos de crear (webdoclet). Con "destdir" indicamos el directorio donde se deben dejar los resultados. Con "fileset dir='src'" indicamos donde se deben buscar los ficheros en los que se buscaran los tags de Xdoclet.

Con "strutsconfigxml" estamos indicando a Xdoclet que queremos generar el fichero struts-config.xml. Le indicamos la versión de struts para la que queremos generar el fichero.

Con "templateFile" estamos indicando la plantilla que Xdoclet va a utilizar para generar el fichero struts-config.xml. Esto no deber ser necesario ya que Xdoclet trae una plantilla para hacer esto. Yo lo he puesto porque con la plantilla por defecto tuve algunos problemillas con las clases de formulario que no extendían de org.apache.struts.action.ActionForm, así que cambié la plantilla por defecto.

Con "mergedir" indicamos el directorio donde se encuentran los ficheros de "merge". Estos ficheros los utiliza Xdoclet para insertarlos dentro del fichero struts-config.xml que está generando. Esto sirve para definir ciertas partes del fichero struts-config.xml que son fijas (como la definición de los datasources).

Con "strutsvalidationxml" estamos indicado a Xdoclet que también queremos que genere el fichero validation.xml.

3. Ficheros de merge

En el punto anterior hemos hemos hablado del atributo "mergeDir", y decíamos que en este directorio tendríamos algunos fichero que Xdoclet iba a incluir en el fichero struts-config.xml.

A continuación vamos a enumerar estos ficheros, y a ver un ejemplo de su contenido:

- **struts-data-sources.xml**: Definimos los datasources de struts. Nótese que en este fichero tenemos que poner los tags <data-sources> y </data-sources>.

```
<data-sources>
    <!-- Ejemplo de definición de datasource
    <data-source type="org.apache.commons.dbcp.BasicDataSource">
        <set-property
          property="driverClassName"
          value="org.postgresql.Driver" />
        <set-property
          property="url"
          value="jdbc:postgresql://localhost/mydatabase" />
        <set-property
          property="username"
          value="me" />
        <set-property
          property="password"
          value="test" />
        <set-property
          property="maxActive"
          value="10" />
        <set-property
          property="maxWait"
          value="5000" />
        <set-property
          property="defaultAutoCommit"
          value="false" />
        <set-property
          property="defaultReadOnly"
          value="false" />
        <set-property
          property="validationQuery"
          value="SELECT COUNT(*) FROM market" />
    </data-source>
    -->
</data-sources>
```

- **struts-forms.xml**: Nos permite definir formularios que no se van a mantener con Xdcolet (tags en los comentarios de javadoc).

```
<!-- sample form bean descriptor for a DynaActionForm
   <form-bean name="logonForm"
      type="org.apache.struts.action.DynaActionForm">
      <form-property
         name="username"
         type="java.lang.String"/>
      <form-property
         name="password"
         type="java.lang.String"/>
  </form-bean>
end sample -->
```

- **global-exceptions.xml**: Para definir excepciones globales. Téngase en cuenta que tenemos que especificar los tags <global-exceptions> y </global-exceptions>

```
<global-exceptions>
   <!-- sample exception handler
   <exception
      key="expired.password"
      type="app.ExpiredPasswordException"
      path="/changePassword.jsp"/>
   end sample -->
</global-exceptions>
```

- **global-forwards.xml**: Para definir redirecciones globales (se pueden usar desde cualquier acción). Nótese que es necesa especificar los tags <global-forwards> y </global-forwards>

```
<global-forwards>
   <!-- Default forward to "Welcome" action
   <forward
      name="welcome"
      path="/Welcome.do"/>
   -->
</global-forwards>
```

- **struts-actions.xml**: Nos permite definir acciones que no se van a mantener con Xdcolet (tags en los comentarios de javadoc).

```
<!-- Ejemplo de acción para saltar a index.jsp
<action
   path="/index"
   forward="/index.jsp"
/>
```

- **struts-controller.xml**: Para definir el controller.

```
<controller processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>
```

- **struts-message-resources.xml**: Para definir los ficheros de recursos.

```
<message-resources parameter="MessageResources" />
```

- **struts-plugins.xml**: Para definir los plugins.

```
<!-- ========================================================== Tiles plugin -->
<plug-in className="org.apache.struts.tiles.TilesPlugin" >
   <!-- Path to XML definition file -->
   <set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml" />

   <!-- Set Module-awareness to true -->
   <set-property property="moduleAware" value="true" />
```

```
</plug-in>


<!-- ========================================================== Validator plugin -->
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">
   <set-property property="pathnames"
                value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
</plug-in>
```

4. Definiendo un Formulario

Supongamos que tenemos una clase del estilo:

```
/**
 * @author autentia.alex
 */
public class Contact {
   private String name = null;
   private String familyName = null;
   private String email = null;

   public Contact() {
   }

   public String getName() {
      return this.name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getFamilyName() {
      return this.familyName;
   }

   public void setFamilyName(String familyName) {
      this.familyName = familyName;
   }

   public String getEmail() {
      return this.email;
   }

   public void setEmail(String email) {
      this.email = email;
   }
}
```

Para definir un formulario de Struts con esta clase basta con añadir lo siguiente (lo que está en negrita):

```
/**
 * @author autentia.alex
 *
 * @struts.form name="contactForm"
 */
public class Contact extends ValidatorForm {
   private String name = null;
   private String familyName = null;
   private String email = null;

   public Contact() {
   }

   public String getName() {
      return this.name;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getFamilyName() {
      return this.familyName;
   }

   public void setFamilyName(String familyName) {
      this.familyName = familyName;
   }
```

```
    public String getEmail() {
        return this.email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

Esto nos generará la siguiente entrada en el struts-config.xml:

```
...
<form-beans>
    <form-bean
      name="contactForm"
      type="com.autentia.contacto.vo.Contact"
    />
</from-beans>
...
```

5. Definiendo las validaciones

Si queremos que sobre los campos del formulario se hagan validaciones (un campo obligatorio, que cumpla un determinado formato, ...) bastará con poner algunos tags (ver los cambios en negrita):

```
/**
 * @author autentia.alex
 *
 * @struts.form name="contactForm"
 */
public class Contact extends ValidatorForm {
    private String name = null;
    private String familyName = null;
    private String email = null;

    public Contact() {
    }

    public String getName() {
        return this.name;
    }

    /**
     * @struts.validator type="required"
     */
    public void setName(String name) {
        this.name = name;
    }

    public String getFamilyName() {
        return this.familyName;
    }

    public void setFamilyName(String familyName) {
        this.familyName = familyName;
    }

    public String getEmail() {
        return this.email;
    }

    /**
     * @struts.validator type="email"
     */
    public void setEmail(String email) {
        this.email = email;
    }
}
```

Con esto estamos indicando que el campo "name" es obligatorio, y que el campo email debe tener formato de correo electrónico. Nótese que los comentarios sobre validaciones se ponen en los métodos "set".

Esto generaría la siguiente entrada en el fichero de validaciones validation.xml:

```
...
<form name="contactForm">
    <field property="name" depends="required">
```

```
        <arg0 key="contactForm.name"/>
    </field>
    <field property="email" depends="email">
        <arg0 key="contactForm.email"/>
    </field>
</form>
...
```

Fíjese como aparecen las dos claves "contactForm.name" u contactForm.email". Estas se han generado automáticamente a partir del nombre del formulario y el nombre del campo. Tendremos que añadir estas claves a nuestro fichero de recursos para darles valor según el idioma.

6. Definiendo una acción

Ya sólo nos queda ver como se definen las acciones. Para seguir con el mismo ejemplo podríamos tener la siguiente acción:

```
/**
 * @author autentia.alex
 */
public class InsertContact extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
            HttpServletRequest request, HttpServletResponse response) throws Exception {
        Contact contact = (Contact) form;
        ContactManager cm = ContactManager.getInstance();
        cm.insert(contact);
        return mapping.findForward("ok");
    }
}
```

Para dar de alta esta acción en el fichero struts-config.xml basta con añadir las siguientes líneas (aparecen en negrita):

```
/**
 * @author autentia.alex
 *
 * @struts.action path="/insertContact"
 *                name="contactForm"
 *                input="/formContact.jsp?action=insert"
 * @struts.action-forward name="ok" path="/index.do"
 */
public class InsertContact extends Action {
    public ActionForward execute(ActionMapping mapping, ActionForm form,
            HttpServletRequest request, HttpServletResponse response) throws Exception {
        Contact contact = (Contact) form;
        ContactManager cm = ContactManager.getInstance();
        cm.insert(contact);
        return mapping.findForward("ok");
    }
}
```

Esto generará la siguiente entrada en el fichero struts-config.xml:

```
<action
  path="/insertContact"
  type="com.autentia.contacto.struts.action.InsertContact"
  name="contactForm"
  scope="request"
  input="/formContact.jsp?action=insert"
  unknown="false"
  validate="true"
>
  <forward
    name="ok"
    path="/index.do"
    redirect="false"
  />
</action>
```

7. Conclusiones

En todos nuestros proyectos siempre tenemos que intentar seguir la regla del beso seco. Esto en español queda un poco mal, peo en ingles sería DRY KISS, donde DRY = Don't repeat yourself (no te repitas) y KISS = Keep it simple, stupid (hazlo lo más sencillo que puedas).

En definitiva, es intentar siempre que el código sea lo más sencillo posible de mantener. Con el uso que hemos visto de Xdoclet en este tutorial hacemos un buen acercamiento a la regla DRY ya que evitamos el tener que mantener dos ficheros por separado (código fuente java, y fichero de configuración xml), y encima escribimos menos código.

Además también estamos consiguiendo que la interacción entre distintos desarrolladores sea más sencilla, lo cual también es una gran ganancia.

Espero que os sirva de ayuda, y sobre todo recordad siempre la regla del beso seco ;)

8. Sobre el autor

Alejandro Pérez García

Dir. Implantación y Rendimiento

mailto:alejandropg@autentia.com

Autentia Real Business Solutions S.L.

http://www.autentia.com

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.
AspectJ, Programación con Aspectos	Os mostramos como configurar AspectJ (extensión Java para la programación basada en aspectos) y un pequeño ejemplo para medir la velocidad de una función sin alterar su código.
Pool de conexiones a BBDD con struts	Os mostramos como configurar un pool de conexiones a base de datos en vuestras aplicaciones construidas con struts
Introducción a ANT	En el mundo Java, la compilación, verificación e instalación de aplicaciones se ha normalizado con este potente paquete llamado ANT.
Novedades en Java 1.5	Ya está disponible la versión Beta del J2SDK 1.5. Os mostramos algunas de las nuevas características introducidas en el lenguaje Java: Clases genéricas, enumeraciones, bucles simplificados, etc.
Generación .exe con ficheros Java	Os mostramos como, utilizando la herramienta gratuita JSmooth, podemos generar un fichero .exe a partir de nuestros programas Java
Patrones de diseño J2EE	Os mostramos una interpretación particular de los patrones de diseño J2EE

Consola de administración de Struts	En este tutorial aprenderemos a simplificar la gestión de Struts a través de una consola gráfica gratuita
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 expresiones	Os mostramos las novedades de JSP 2.0: Nuevas librerías estandar de etiquetas y el lenguaje de expresiones con ejemplos de acceso a base de datos, XML y XSL en JSP