

¿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

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

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

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



Hosting Patrocinado por
enREDados.com



[Home](#) | [Quienes Somos](#) | [Empleo](#) | [Foros](#) | [Tutoriales](#) | [Servicios Gratuitos](#) | [Contacte](#)

	<p>Tutorial desarrollado por: Roberto Canales Mora 2003-2005 Creador de AdictosAlTrabajo.com y</p> <p>Director General de Autentia S.L.</p> <p>Recuerda que me puedes contratar para echarte una mano:</p> <p>Desarrollo y arquitectura Java/J2EE Asesoramiento tecnológico Web Formación / consultoría integrados en tu proyecto</p> <p>No te cortes y contacta: 655 99 11 72 rcanales@autentia.com.</p>	 <p>autentia real business solutions</p>
---	--	--

Descargar este documento en formato PDF [patronesj2ee.pdf](#)

[Curso Web J2EE](#)

Curso Avanzado en Desarrollo Web con J2EE

[XML to PDF in Java & J2EE](#)

Output PDF, PCL5, HTML in Java J2EE Websphere, Weblogic, Tomcat, Jetty.

[diseño web Zaragoza](#)

Diseño y programación web a medida Excelente relación calidad/precio

[Master Java J2ee Oracle](#)

Prácticas laborales 100% aseguradas Nuevo temario de Struts. Trabaja ya

Anuncios Goooooogle

Anunciarse en este sitio

Patrones de Diseño J2EE

Para construir aplicaciones J2EE de un modo profesional, debemos conocer técnicas avanzadas de análisis, diseño y desarrollo (podéis ver otro tutorial donde hablábamos de [patrones de análisis](#)).

Sun tiene una obra grandiosa ([disponible vía Web](#) aunque es un libro que todos deberíamos tener en las estanterías) donde nos propone una serie de patrones para ayudarnos a la construcción de nuestras aplicaciones java. Sin su conocimiento, y el de algunas obras más, el diseño y desarrollo de aplicaciones puede ser nefasto...

La labor de arquitecto de software (lo que antes era el tecnólogo de las organizaciones) es muy duro y complejo, fundamentalmente por el entorno tan cambiante dentro del mundo Java y las nuevas tecnologías en general (yo no paro de estudiar y siempre me faltan cosas que probar).

Os ofrecemos un pequeño resumen de algunos de los patrones ofrecidos en la obra de Sun, con algunos toques personales. Es un requisito para entenderlos medianamente bien haber visto los [patrones GoF](#) (en este enlace podéis leer sobre patrones).

Intercepting Filter	<p>Cuando un cliente realiza una petición, se pueden realizar comprobaciones sobre ella de un modo transparente (por ejemplo: por si contienen código malicioso).</p> <p>La respuesta, también puede que nos interese tratarla (por ejemplo: para transformarla, comprimirla, etc..).</p> <p>La gracia es que estos filtros se pueden conectar en cascada y activar y desactivar sin afectar al código de nuestra aplicación (o justo lo contrario, que también nos puede interesar)</p> <p>Es sencillo implementar este patrón con el uso de filtros que podemos encontrar en últimas versiones de la especificación de servlets.</p> <p>El uno de nuestros tutoriales anteriores podéis ver la implementación de un filtro para comprobar la velocidad de las peticiones que nos hace un cliente.</p>
Front Controller	<p>Los sistemas requieren de unos servicios centralizados de validación de parámetros, invocación de acciones de negocio, invocación a la vista adecuada (incluso dependiendo del tipo de dispositivo que realiza la petición) , etc...</p> <p>Un controlador se encarga de recoger las peticiones y unificar el código repetido.</p> <p>Si recurrimos a los patrones de asignación de responsabilidad (GRASP) podemos identificar estos comportamientos desde la fase de análisis.</p> <p>Podemos ver el núcleo de la creación de un controlador para servlets en este otro tutorial.</p>

<p>View Help</p>	<p>Un Helper es una clase que se encarga de aglutinar cierto código común. Es aplicable tanto para las capas de negocio como para presentación.</p> <p>Cuando hablamos de View Helper, hacemos referencia a clases que, utilizadas desde la capa de presentación, encapsulan la complejidad del acceso a las verdaderas estructuras de datos.</p> <p>Cuando hablamos de JSPs, la manera más práctica de implementar un Helper es a través de etiquetas (custom tags). Podemos ver como crear estas etiquetas en el siguiente enlace.</p> <p>Otro modo de hacerlo es a través de JavaBeans pero trataremos de evitar lo máximo posible la inclusión de código Java. La relación entre TAGs y Beans la podéis encontrar en este enlace.</p>
<p>Composite View</p>	<p>Las aplicaciones Web reales agregan mucha información distinta (contenidos) en sus páginas, muchas veces poco relacionada con nuestro negocio (noticias, encuestas, estadísticas, cotizaciones, etc..).</p> <p>Normalmente, es necesario utilizar técnicas avanzadas de composición de estas páginas para simplificar su construcción y mantenibilidad.</p> <p>JSP (y sevlts) nos proporciona mecanismos sencillos para poder incluir porciones de una página en otra (de modo estático o dinámico). Estos mecanismos, normalmente son insuficientes por sí mismos por lo que tenemos que utilizar algunas técnicas más elaboradas en incluso recurrir a productos especializados como gestores de contenidos.</p>
<p>Service to Worker</p>	<p>De los patrones vistos anteriormente, parece de sentido común que un framework de trabajo debe combinar todos los mecanismos.</p> <p>Service to worker define un marco global donde separar las distintas responsabilidades a la hora de servir una petición.</p>
<p>Dispatcher View</p>	<p>Cuando en un controlador, debemos derivar la respuesta a un componente de presentación, normalmente deseamos separar su parte lógica de la física. Es decir, establecer un nivel más de indirección y poder intercambiar los elementos físicos de presentación de un modo transparente a la lógica de la aplicación.</p> <p>Esto además facilita la integración entre aplicaciones y la gestión centralizada de mensajes de error....</p> <p>El controlador delega sobre el dispatcher la presentación de un elemento lógico. El dispatcher se encarga de resolver la indirección y proporcionar el elemento físico.</p> <p>El patrón MVC con servlet y JSPs utiliza un dispatcher aunque podemos verlo de un modo más claro en aplicaciones basadas es struts.</p>
<p>Business Delegate</p>	<p>Si hemos decidido, en nuestras aplicaciones, la utilización de componentes complejos, (EJBs, JMS, etc) nos puede interesar simplificar su utilización (o abstraer la implementación para posteriormente poder cambiarla) a la capa de presentación.</p> <p>Podemos crear unas clases que hagan de proxy sobre las funciones reales remotas.</p> <p>Este nuevo nivel de indirección añade más trabajo pero mejora la flexibilidad.</p> <p>En algunos entornos de desarrollo integrado, la construcción de estas clases se realiza con un click de ratón.</p> <p>Este patrón nos puede ayudar, combinado con otros, a resolver situaciones particulares:</p> <ul style="list-style-type: none"> ● Cache de objetos ● Repetición de peticiones cuando un objeto remoto no está disponible (o hay un problema temporal de comunicaciones).
<p>Value Object</p>	<p>Una aplicación que utilice EJB de entidad, cada vez que accede a un atributo, está invocando a un método remoto. Además, existir multitud de mensajes para mantener la integridad del objeto.</p> <p>Un mecanismo sencillo, en multitud de aplicaciones, para solucionar el problema, podría ser retornar un objeto simple (Value Object). De este modo, el rendimiento del sistema se ve mejorado.</p> <p>En esta situación, podría adicionalmente existir un EJB de sesión con estado, que se encargase de realizar las sincronizaciones.</p>

	<p>Desde un punto de vista más simple, cuando invocamos un método una clase, podemos pasar los parámetros agrupados en un VO (Value Object)... incluso los valores en la sesión de un servlet, podría agruparse en un Value Object</p> <p>Aunque debemos no abusar. Este abuso se suele llamar Golden Hammer o martillo de oro que consiste en, una vez conocida una técnica, aplicarla indiscriminadamente en todas las situaciones (aún sin hacer realmente falta).</p>
Session Facade	<p>Las aplicaciones inician sencillas pero se van poco a poco complicando. Esto provoca que la capa de negocio se vea muchas veces condicionada por la capa de presentación.</p> <p>La capa de presentación es posible que, para controlar el flujo de ejecución, tenga que llamar a muchos objetos de la capa de negocio. Si la capa de negocio está implementada como EJBs, el problema se agrava (y más si accedemos directamente a EJBs de entidad).</p> <p>Podemos crear un EJB de sesión que centralice las peticiones entre la capa de presentación y negocio, que proporcione el correcto flujo de control para una aplicación en concreto, sin obligar a deformar nuestros objetos atómicos de negocio.</p>
Composite Entity	<p>Uno de los principales problemas de diseño a la hora de utilizar EJBs de entidad, consiste en definirlos con un nivel muy bajo de atomicidad (objetos muy pequeños).</p> <p>Si estos objetos ya existen y debemos construir una aplicación, podríamos construir uno que los unifique, de tal modo que a la capa de presentación (o al EJB que actúe de fachada según el patrón anterior), se le simplifiquen los accesos.</p> <p>Adicionalmente deberíamos analizar el libro de Anti-pones J2EE porque el exceso de aplicación de este patrón, también lo podemos pagar.</p>
Value Object Assembler	<p>Las aplicaciones J2EE es posible que utilicen un modelo de datos que esté compuesto por distintos tipo de objetos con distintos accesos (EJB de Entidad, JDO, JDBC, etc.). Para simplificar los accesos podemos crear un objeto que se encargue en ensamblar estos modelos.</p>
Value List Handler	<p>Una aplicación cliente puede requerir una lista de objetos que obtiene a través de la invocación de un servicio. Podemos crear un componente que, implementando un interfaz estándar, nos permita recuperar los valores e iterar por ellos (independientemente de la implementación) y realizar otras labores de valor añadido (como cachear los datos o reintentar su recuperación).</p> <p>Es conveniente revisar los interfaces de Java estándar para no repetir comportamiento ya existentes en los sistemas.</p>
Service Locator	<p>En nuestras aplicaciones tenemos que acceder a pilas de conexiones JDBC, a EJBs, a servicios asíncronos, etc..</p> <p>Todos estos servicios requieren código particular (creación de contexto, carga de clases, etc.) en función de la implementación particular. Esta parte específica que nos permite acceder a los servicios, la podemos centralizar en un componente denominado localizador de servicios.</p> <p>Este localizador de servicios se encarga de retornarnos los recursos listos para utilizar, independientemente de como se hayan obtenido.</p>
Data Access Object	<p>Los objetos de negocio requieren acceder a datos.</p> <p>Estos datos pueden estar almacenados de modos distintos (EJBs, JDO, LDAP, etc.). A la lógica de negocio le debería dar igual como estos datos estén almacenados y como se realicen los accesos.</p> <p>El patrón DAO nos enseña como podemos hacerlo de un modo sencillo y transparente. Este patrón realmente es una combinación de otros</p> <p>Creo que es el patrón que hay que estudiar con mayor profundidad porque, siendo el más obvio, es el que creo que más valor añadido aporta.</p>
Service Activator	<p>Muchas aplicaciones no se pueden comportar de un modo síncrono, es decir, realizar una petición y esperar una respuesta inmediata.</p> <p>Numerosas veces es necesario realizar aplicaciones que envíen una petición y dispongan de un mecanismo para atender adecuadamente a la respuesta, cuando se encuentre disponible.</p>

El patrón service activator nos proporciona un modelo simple que podemos implementar a través de servicios de mensajería asíncrona como JMS...

Estos patrones realmente son un punto de partida para hacernos pensar sobre problemas comunes en aplicaciones J2EE.

Solo el estudio de las técnicas de diseño y el esfuerzo de modelado (utilizando UML para discutir con compañeros en un lenguaje común) nos permitirá crecer como arquitectos ...

Sobre esta base y repasando la arquitectura sobre la que se encuentra construido Java y algunos de los Frameworks más importantes, podremos llegar a construir aplicaciones de gran valor arquitectónico (evitando errores comunes)... eso sí... hay que estudiar un poquito

[Sobre el Autor ..](#)

Si desea contratar formación, consultoría o desarrollo de piezas a medida puede contactar con

Formación en nuevas tecnologías

[Autentia S.L.](#) Somos expertos en:
J2EE, C++ , OOP, UML, Vignette, Creatividad ..
 y muchas otras cosas

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	
	<input type="button" value="Enviar"/>

Otros Tutoriales Recomendados ([También ver todos](#))

Nombre Corto	Descripción
Comunicación entre TAGs, Beans y JSPs	Os mostramos las posibilidades de comunicación entre JSPs, Bean y etiquetas de usuario.
Configuración y acceso a OpenLdap desde Java con JNDI	Con este tutorial, aprenderás como realizar la instalación de OpenLdap, así como la carga de un LDIF básico, y a configurar el entorno Java para acceder a la información.
JDO con OJB	Os mostramos como configurar el entorno OJB de apache para construir la primera aplicación JDO
Patrones de GRASP	Os presentamos una introducción a los patrones de asignación de responsabilidades y su relación con el proceso unificado.
Cachear porciones de JSPs	En este tutorial os enseñamos como incrementar increíblemente el rendimiento de vuestro Web basado en tecnología JSP con el Framework de cache OSCACHE
Aplicación de Patrones de Diseño en Java	En este tutorial os mostramos como las técnicas avanzadas de diseño (como patrones de diseño) contribuyen a la construcción de aplicaciones profesionales en Java.
CMMI. Modelo de Madurez Software	Os introducimos a CMMI o Capability Maturity Model Integration. CMMI es un modelo de calidad exigido por el gobierno americano a sus proveedores para el desarrollo de Software. Su conocimiento es esencial para reducir costes de desarrollo.
Introducción al UML	Este es el primer artículo sobre el diseño de proyectos orientados a objeto con UML, donde se describe los primeros diagramas a utilizar
Struts Jakarta	Cuando se ha trabajado creando aplicaciones Java poco a poco se va viendo la necesidad de normalizar los desarrollos. Uno de los Framework (entornos) más extendidos es Struts
TagLibs y JSPs	Os mostramos como crear librerías de etiquetas para vuestros JSP y así simplificar su construcción.

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.

[Patrocinados por enredados.com Hosting en Castellano con soporte Java/J2EE](#)



**¿Buscas un hospedaje de calidad
por sólo 2 € al mes?**

www.AdictosAlTrabajo.com Optimizado 800X600