

¿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](#) | [Tutoriales](#) | [Contacte](#)



CoNcepT

Nuevo

TNTConcept versión 0.2 (26/04/2007)

Autentia da un paso más en su evolución, hemos lanzado una nueva versión con más de 50 mejoras. Ponemos a vuestra disposición el software que hemos construido (100% gratuito y sin restricciones funcionales) para nuestra gestión interna, llamado TNTConcept (auTeNTia).

Construida con las últimas tecnologías de desarrollo Java/J2EE (Spring, JSF, Hibernate, Maven, Subversion, etc.) y disponible en licencia GPL, seguro que a muchos profesionales independientes y PYMES os ayudará a organizar mejor vuestra operativa.

Las cosas grandes empiezan siendo algo pequeño Saber más en: <http://tntconcept.sourceforge.net/>

	Tutorial desarrollado por: Javier Antoniucci Puedes encontrarme en Autentia Somos expertos en Java/J2EE Contacta en javier.antoniucci@autentia.com	www.adictosaltrabajo.com es el Web de difusión de conocimiento de www.autentia.com  autentia real business solutions Catálogo de cursos
---	--	---

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

[Firma en nuestro libro de Visitas](#) <-----> [Asociarme al grupo AdictosAlTrabajo en eConozco](#)

Manual Gratis de Java

Aprende Java totalmente gratis.
 JAVA-Programación Orientada
 Objetos
www.mundojava.net

Master Experto Java

100% alumnos trabajando Nuevo
 temario de Struts + Ajax
www.grupoatrium.com

Central Telefonica

Última Generación en Digitales,
 Analógicas y más. Servicio 24 hrs
www.avancemgrup.com

Prestamos Bajo Interes

En Agencia Negociadora queremos
 que tus sueños se hagan realidad
www.agencianegociadora.com/pres

Anuncios Google

Spring: definición dinámica de Beans

A medida que aplicamos Spring a nuestros proyectos, nos acostumbramos a su inyección de dependencias y un día nos damos cuenta de que no podríamos volver a trabajar como antes. Más o menos como nos pasó cuando descubrimos Struts o cualquier MVC. Y cuando más avanzamos, más queremos y Spring es casi inagotable.

Cuando iniciamos el IoC de Spring, el contexto se carga con la configuración XML que hayamos indicado. Una característica poco conocida y menos documentada es la capacidad de modificar dicho contexto programáticamente.

A priori puede parecer medio "traído de los pelos" querer hacer una cosa semejante. ¿para qué vamos a querer hacer una cosa así? Si incluso Spring nos permite cargar el contexto desde varios XMLs. Bueno, resulta muy útil en las siguientes situaciones:

- Carga dinámica de plugins
- Configuración de controladores MVC para flujos de WebFlow
- Automatización de la configuración de Spring mediante anotaciones
- ...y muchos etcéteras donde no se quiera/pueda incluir la configuración en los XML

A fines de simplificar la explicación, vamos a organizar el desarrollo del tutorial en tres secciones:

- Los ApplicationContext
- Los BeanFactory
- Juntando todo

Por último os recomendamos algunos links y nuestras conclusiones.

Los ApplicationContext

Habitualmente, el IoC es responsable por la "interconexión" de los objetos y no debíamos preocuparnos por acceder al contexto; pero en ciertos casos es inevitable acceder al mismo para buscar una referencia a una instancia que necesitamos. En el caso de necesitar acceder desde un Action, éste deberá implementar ApplicationContextAware como se muestra a continuación. En este ejemplo hemos definido una clase abstracta ya que el código se reutilizará en todos los actions:

```
public abstract class AbstractAction extends FormAction implements ApplicationContextAware {
```

...

```

protected MessageSourceAccessor messageSourceAccessor;

protected ApplicationContext applicationContext;...

...

public void setApplicationContext(ApplicationContext context) throws BeansException {

    if (context == null) {

        // reset internal context state

        this.applicationContext = null;

        this.messageSourceAccessor = null;

    }

    if (this.applicationContext == null) {

        // initialize with passed-in context

        if (!(ApplicationContext.class).isInstance(context)) {

            throw new ApplicationContextException(

                "Invalid application context: needs to be of type [" + (ApplicationContext.class).getName() + "]);

        }

        this.applicationContext = context;

        this.messageSourceAccessor = new MessageSourceAccessor(context);

    }

    else {

        // ignore reinitialization if same context passed in

        if (this.applicationContext != context) {

            throw new ApplicationContextException(

                "Cannot reinitialize with different application context: current one is [" +

                this.applicationContext + "], passed-in one is [" + context + "]);

        }

    }

}

...

}

```

El messageSourceAccessor no es imprescindible en todo los casos pero nos resultará muy útil cuando querramos mostrar un mensaje en el idioma del usuario y éste es el lugar/momento de obtener una referencia al mismo.

Para modificar el contexto, debemos castear el ApplicationContext en un ConfigurableApplicationContext:

```
ConfigurableApplicationContext configContext = (ConfigurableApplicationContext) this.applicationContext;
```

Ahora podremos modificar la configuración del contexto.

Los BeanFactory

Para poder añadir Beans, primero debemos crear una definición del bean y para ello Spring nos provee de diversos BeanDefinitions:

```
RootBeanDefinition bean = new RootBeanDefinition(org.springframework.webflow.executor.mvc.FlowController.class)
```

Obviamente podemos añadirle propiedades de la siguiente forma:

```
MutablePropertyValues propertyValues = new MutablePropertyValues();
```

```
propertyValues.addPropertyValue("flowExecutor", flowExecutor);
```

```
propertyValues.addPropertyValue("defaultFlowId", nombreControlador + "-flow");
```

```
bean.setPropertyValues(propertyValues);
```

Una vez completada la definición del Bean, la añadiremos al contexto utilizando un BeanFactory como el siguiente:

```
DefaultListableBeanFactory beanFactory = (DefaultListableBeanFactory) configContext.getBeanFactory();
```

Con este BeanFactory añadimos el bean mediante:

```
beanFactory.registerBeanDefinition(nombreControlador, bean);
```

Y básicamente esta es la solución.

Juntando todo

Básicamente, porque en la práctica nos encontraremos con variadas situaciones y soluciones alternativas para cada una de ellas. Revisando la documentación se puede descubrir que los BeanDefinition, BeanFactory, etc. cuentan con varias implementaciones específicas.

Una de las primeras dificultades que nos encontraremos consiste en intentar modificar el contexto mientras se está cargando, por ejemplo durante la inicialización de un bean que pertenece al contexto. Obtendremos un ConcurrentModificationException. Esta excepción se debe a que el contenedor ve modificada la iteración sobre la configuración y no puede continuar con la misma.

Una forma de evitarlo consiste en implementar en dicho bean una interfaz ApplicationListener con la que nuestro Bean será notificado de los eventos de la aplicación. Entonces, para el evento concreto de finalización de la carga o actualización del contexto (ContextRefreshedEvent) podemos implementar la carga dinámica de Beans sin peligro de que surja la mencionada excepción.

Un ejemplo:

```
public void onApplicationEvent(ApplicationEvent event) {

    if (event instanceof ContextRefreshedEvent) {

        // Configura los beans de los controladores y los carga en el contexto

        ConfigurableApplicationContext configContext = (ConfigurableApplicationContext) this.applicationContext;

        DefaultListableBeanFactory beanFactory = (DefaultListableBeanFactory) configContext.getBeanFactory();

        final Enumeration keys = this.mapeosIdiomas.keys();

        FlowExecutor flowExecutor = (FlowExecutor) this.getApplicationContext().getBean("flowExecutor");

        while (keys.hasMoreElements()) {

            final String clave = (String) keys.nextElement();

            final String[] valores = clave.split("_", 3);

            final String nombreControlador = valores[1];

            // Define el controlador

            RootBeanDefinition bean = new RootBeanDefinition(

                org.springframework.webflow.executor.mvc.FlowController.class);

            MutablePropertyValues propertyValues = new MutablePropertyValues();

            propertyValues.addPropertyValue("flowExecutor", flowExecutor);

            propertyValues.addPropertyValue("defaultFlowId", nombreControlador + "-flow");

            bean.setPropertyValues(propertyValues);

            beanFactory.registerBeanDefinition(nombreControlador, bean);

        }

        this.setDefaultHandler(this.getApplicationContext().getBean(this.defaultHandlerId));

    }

}
```

Algunos links interesantes

- Documentación oficial de la API de Spring, <http://www.springframework.org/docs/api/>
- Documentación oficial de Referencia / Chapter 3. The IoC Container, <http://www.springframework.org/docs/reference/beans.html>

Conclusiones

La modificación dinámica de los beans del contexto puede resultar muy útil para abordar problemáticas donde el dominio cambia dinámicamente según lo indica el usuario o para simplificar la configuración de Spring. Requiere conocimientos avanzados y una buena dosis de autodidacta ya que la documentación no es muy extensa. Los foros oficiales cuentan con algunos threads que nos pueden orientar.

Desde Autentia contamos con los conocimientos y experiencia para ayudarle a sacar la máxima ventaja de las tecnologías más innovadoras y mejorar la calidad de sus desarrollos software.

No dude en contactarse con nosotros mediante www.autentia.com.



This work is licensed under a [Creative Commons Attribution-NonCommercial-No Derivative Works 2.5 License](http://creativecommons.org/licenses/by-nc-nd/2.5/).
[Puedes opinar sobre este tutorial aquí](#)



Recuerda

que el personal de [Autentia](http://www.autentia.com) te regala la mayoría del conocimiento aquí compartido ([Ver todos los tutoriales](#))

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?

¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

info@autentia.com

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos

Autentia = Soporte a Desarrollo & Formación

Creatividad Internet

[Autentia S.L.](http://www.autentia.com) Somos expertos en:

J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ..
 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	
	Enviar

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

Nombre Corto

[MBeans y JBoss](#)

[URLs amigables con Spring MVC](#)

[XMBeans y JBoss](#)

[Introducción a Spring Web Flow](#)

[Comunicación entre TAGs, Beans y JSPs](#)

[CMP Entity Beans y MySql](#)

[Desarrollo de Entity Beans](#)

[JOX Mapping entre JavaBeans y XML](#)

Descripción

Este tutorial tiene como finalidad familiarizarse con el servidor de aplicaciones JBoss y su API JMX

En este tutorial se va a hacer un ejemplo práctico utilizando Spring MVC para la configuración de URLs amigables de nuestra aplicación

Este tutorial es una continuación de una anterior llamado MBeans y JBoss

Spring Web Flow es un módulo de extensión del framework Spring, que facilita la implementación del flujo de páginas de una aplicación web

Os mostramos las posibilidades de comunicación entre JSPs, Bean y etiquetas de usuario.

Os mostramos como crear un Entity Bean con persistencia controlada por el servidor, configurado para usar MySql

Os mostramos como construir un Entity Bean básico y desplegarlo en el servidor J2EE de referencia. Lo usaremos como base de buenas prácticas J2EE

En este tutorial se realiza una presentación de una de las muchas APIS que nos facilitan esta tarea, de mapeo entre un documento XML y un JavaBean

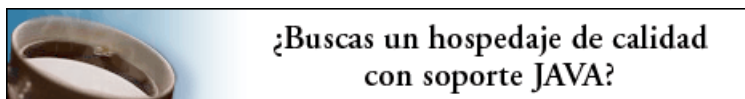
Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

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](#)



www.AdictosAlTrabajo.com Optimizado 800X600