

¿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)

AdictosAlTrabajo

Terrakas 1x04
¡¡Ya está en la web!! ;-)
terrakas.com

autentia
Soporte a desarrollo informático
Hosting patrocinado por
enREDados
Entra en Adictos a través de  E-mail Contraseña

Entrar

Deseo registrarme
Olvidé mi contraseña

Inicio

Quiénes somos

Formación

Comparador de salarios

Nuestro libro

Más 

» Estás en: Inicio » Tutoriales » Creación de una base de datos embebida en memoria con el soporte de Spring.



Jose Manuel Sánchez Suárez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en **Autentia**: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/J2EE



Ver todos los tutoriales del autor

Catálogo de servicios Autentia

entre otras muchas más que encontrarás en...



Fecha de publicación del tutorial: 2012-09-28

Tutorial visitado 5 veces [Descargar en PDF](#)

Creación de una base de datos embebida en memoria con el soporte de Spring.

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Configuración.
- 4. Creación de la base de datos embebida.
- 5. Creación de la estructura de tablas con el soporte de Hibernate y carga de datos iniciales.
- 6. Conclusiones.

1. Introducción

A partir de la versión 3.0, Spring proporciona un espacio de nombres específico en el esquema de configuración xml, para crear una base de datos embebida y en memoria en el arranque de nuestra aplicación java.

Hasta ahora podíamos crearla fácilmente publicando un bean de tipo DataSource y configurando los parámetros de la conexión, ahora es mucho más simple y, además, nos permite procesar ficheros sql después de su creación para crear la estructura de las tablas o realizar una carga de datos inicial.

Por defecto, Spring proporciona soporte para:

- HSQL (por defecto), <http://hsqldb.org/>
- H2, <http://www.h2database.com/html/main.html>
- Derby, <http://db.apache.org/derby/>

Usamos una base de datos embebida y en memoria para ejecutar nuestros tests de integración contra una base de datos real, aunque no física; para que la ejecución de los mismos no dependan de su estado; de tal modo que se crea y se destruye en el ámbito de ejecución de los tests.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 15' (2.4 GHz Intel Core i7, 8GB DDR3 SDRAM).
- Sistema Operativo: Mac OS X Lion 10.7.4
- Spring 3.1.

3. Configuración.

Lo primero, como siempre, configurar las dependencias de las librerías necesarias, con el soporte de Maven:

```

1 <dependency>
2   <groupId>org.springframework</groupId>
3   <artifactId>spring-core</artifactId>
4   <version>3.1.0.RELEASE</version>
5 </dependency>
6 <dependency>
7   <groupId>org.springframework</groupId>
8   <artifactId>spring-beans</artifactId>

```



Síguenos a través de:



Últimas Noticias

» ¡¡¡Terrakas 1x04 recién salido del horno!!!

» Estreno Terrakas 1x04: "Terraka por un día"

» Nuevos cursos de gestión de la configuración en IOS y Android

» La regla del Boy Scout y la Oxidación del Software

» Autentia conquista los Alpes

[Histórico de noticias](#)

Últimos Tutoriales

» Muro de Facebook: cómo publicarlo en tu web

» Jugando con JSON en Java y la librería GSON. Parte 2

» Introducción a Drools.

» Jugando con JSON en Java y la librería Gson

» Trabajando en Android con Maven

```

9      <version>3.1.0.RELEASE</version>
10     </dependency>
11     <dependency>
12       <groupId>org.springframework</groupId>
13       <artifactId>spring-jdbc</artifactId>
14       <version>3.1.0.RELEASE</version>
15     </dependency>
16   </dependency>
17   <dependency>
18     <groupId>com.h2database</groupId>
19     <artifactId>h2</artifactId>
20     <version>1.3.154</version>
21     <scope>test</scope>
22 </dependency>

```

4. Creación de la base de datos embebida.

Sin el soporte del espacio de nombres, la creación de la base de datos pasaba por configurar un bean como el siguiente en nuestro applicationContext.xml:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xsi:schemaLocation="
4         http://www.springframework.org/schema/beans http://www.springframework.org/schem
5     >
6   <bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource"
7     <property name="driverClassName" value="org.h2.Driver" />
8     <property name="url" value="jdbc:h2:mem:test" />
9     <property name="username" value="sa" />
10    <property name="password" value="" />
11  </bean>
12 </beans>
13

```

Ahora con el espacio de nombres, la configuración se reduce sensiblemente puesto que solo hay que incluir lo siguiente:

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans" xmlns:jdbc="http://www.springf
3     xsi:schemaLocation="
4         http://www.springframework.org/schema/beans http://www.springframework.org/schem
5         http://www.springframework.org/schema/jdbc http://www.springframework.org/schem
6     >
7   <jdbc:embedded-database id="dataSource" type="H2" />
8 </beans>
9

```

Muy simple y, además, nos permite configurar referencias a ficheros sql que se procesarán después de crear la base de datos, para crear la estructura de tablas o poblarlas con datos iniciales.

```

1 <jdbc:embedded-database id="dataSource" type="H2">
2   <jdbc:script location="classpath:ddl.sql"/>
3   <jdbc:script location="classpath:dml.sql"/>
4 </jdbc:embedded-database>

```

5. Creación de la estructura de tablas con el soporte de Hibernate y carga de datos iniciales.

En el entorno de test, haciendo uso de un ORM como Hibernate o cualquiera de las implementaciones de JPA (el propio EntityManager de Hibernate) podemos crear la estructura de las tablas de base de datos (de esta que hemos configurado embebida y en memoria) en función de la configuración de nuestras clases de entidad.

Para ello, no tenemos más que incluir una propiedad específica para configurarlo en la definición de la factoría de sesiones:

```

1 <jdbc:embedded-database id="dataSource" type="H2" />
2
3 <bean id="sessionFactory" class="org.springframework.orm.hibernate3.annotation.Annotation
4   <property name="dataSource" ref="dataSource" />
5   <property name="configurationClass" value="org.hibernate.cfg.AnnotationConfigurati
6   <property name="configLocation">
7     <value>classpath:hibernate.cfg.xml</value>
8 </property>
9   <property name="hibernateProperties">
10    <props>
11      <prop key="hibernate.hbm2ddl.auto">true</prop>
12    </props>
13 </property>
14   <property name="packagesToScan" value="com.autentia.training.**.*" />
15 </bean>

```

Si, además de lo anterior, tuviéramos la necesidad de poblar la base de datos con una carga inicial, podríamos añadir la siguiente configuración:

```

1 <jdbc:initialize-database data-source="dataSource">
2   <jdbc:script location="data.sql" />
3 </jdbc:initialize-database>

```

Con ello, después de crear la base de datos, se ejecuta el contenido del script sql y tendremos los datos listos para nuestro entorno de tests.

No debemos olvidar que los test, aunque sean de integración, deben ser atómicos y deben dejar el estado de la base de datos consistente, de modo que la base de datos, después de la ejecución del test se mantenga en su estado inicial.

Para ello, también son el soporte de Spring y haciendo uso de la gestión declarativa de transacciones, debemos marcar todos los métodos de los test como transaccionales y, con la estrategia por defecto de defaultRollBack, Spring se encargará de no confirmar los cambios tras la salida de cada método.

```

1 @RunWith(SpringJUnit4ClassRunner.class)

```

Últimos Tutoriales del Autor

» Double Opt-In y autologin con el soporte de Spring MVC y Spring Security.

» Posicionamiento de componentes en HTML con el soporte de CSS.

» Test de integración con Solr y el soporte de EmbeddedSolrServer.

» Arrancar Solr desde un proyecto Maven con el soporte de Jetty.

» Lectura y tratamiento de ficheros Excel con Talend (II): filtros y splits

Últimas ofertas de empleo

2011-09-08

Comercial - Ventas - MADRID.

2011-09-03

Comercial - Ventas - VALENCIA.

2011-08-19

Comercial - Compras - ALICANTE.

2011-07-12

Otras Sin catalogar - MADRID.

2011-07-06

Otras Sin catalogar - LUGO.



Jose Manuel Sánchez
sanchezsuarezj

sanchezsuarezj @imvif echa un vistazo a este sobre como consumir un servicio web Axis con Android - kcy.me/b9hd yesterday · reply · retweet · favorite

sanchezsuarezj y la semana próxima un nuevo curso de formación. Revisando material y preparando ejercicios. yesterday · reply · retweet · favorite

sanchezsuarezj @fjpereda gracias a vuestra buena predisposición!!! y a los conocimientos previos, hemos podido abarcar tanto en tan poco tiempo. 3 days ago · reply · retweet · favorite

sanchezsuarezj curso finalizado, asistentes contenidos por el



Join the conversation

```
2 | @ContextConfiguration(locations =
3 | { "classpath:applicationContext-test.xml" })
4 | @Transactional
5 | public class CatalogoDaoTest {
6 |
7 |     @Test
8 |     public void should...
9 |
10 | }
```

Con la anotación en la cabecera de la clase de tests, después de la ejecución de cada método se producirá un rollback.

6. Conclusiones.

Si no tenemos el soporte de una herramienta como [liquibase](#), la configuración que hemos visto en este tutorial puede ser una buena alternativa.

Un saludo.

Jose

jmsanchez@autentia.com

A continuación puedes evaluarlo:

[Regístrate para evaluarlo](#)



Por favor, vota +1 o compártelo si te pareció interesante



Ánimate y coméntanos lo que pienses sobre este **TUTORIAL**:

» [Regístrate](#) y accede a esta y otras ventajas «



Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

Copyright 2003-2012 © All Rights Reserved | [Texto legal y condiciones de uso](#) | [Banners](#) | [Powered by Autentia](#) | [Contacto](#)

