



Struts 2 – Spring – JPA



- Es otro framework que también implementa MVC
 - También implementa otros aspectos de JEE
- Para integrarlo con Struts2 sólo se considerará el manejo de dependencias de objetos
- Una aplicación Java consiste en un conjunto de objetos
 - Estos objetos cooperan entre ellos para resolver los problemas en la aplicación
 - Entre los objetos podemos tener actions, interceptores y otros como *PortfolioService*
 - ¿Como son instanciados?



- Instancias de objetos
 - Algunos objetos son instanciados por el framework
 - Struts2 instancia la clase que tiene el action correspondiente para la ejecución de una solicitud
 - Como desarrolladores solo proveemos el código fuente del action
 - Nunca creamos el action por nuestra cuenta
 - Objetos como PortfolioService no son creados por el framework
 - Muchos actions dependen de este objeto para hacer su trabajo
 - Los actions deben obtener la referencia de ese objeto para poder utilizarlo



- Instancia de objetos
 - Los actions crean los objetos manualmente
 - Utilizan el operador “new”
 - Esto crear una relación estrecha entre el action y PortfolioService
- Spring es una tecnología popular en el manejo de creación de objetos Java
 - Puede ayudar a tener mejor disociación entre objetos
- → Revisar chapter8 y chapter9
- Ya no se crean por nuestra cuenta los objetos
 - Spring lo crea e inyecta en el método setter
 - Los detalles de creación del objeto quedan en un archivo XML



- Una vez que el método setter no realiza “new”
 - Es posible usar una interfaz para aislar la implementación
 - Ante cualquier cambio en la implementación no es necesario tocar los action
 - Solo se intervienen las clases que implementan la interfaz
 - Para el caso de PortfolioService se pueden agregar implementaciones con Hibernate, JPA, jdbc, etc
- Agregando Spring a Struts2
 - Descargar y agregar el plug-in de Spring a la aplicación
 - El plug-in provee las extensiones de Spring al objeto ObjectFactory



- Integración con struts2
 - En el archivo web.xml se debe agregar:

```
<listener>  
<listener-class>org.springframework.web.context.ContextLoaderListener  
</listener-class>  
</listener>
```

- Además se debe indicar cuales objetos son los que debe manejar
 - Se declaran como Spring Beans en la configuración de Spring
 - Se busca por metadata en el archivo /WEB-INF/applicationContext.xml



- Manejando la creación de Login y la utilización de PortfolioService:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

<bean id="portfolioService"
      class="manning.chapterNine.utils.PortfolioServiceJPAImpl"/>

<bean id="springManagedLoginAction"
      class="manning.chapterNine.Login" scope="prototype">
  <property name="portfolioService" ref="portfolioService"/>
</bean>

</beans>
```



- Es necesario agregar todas las bibliotecas .jar al path de la aplicación
- Spring – JPA
 - Se debe agregar la configuración necesaria en applicationContext.xml
 - Existe un BeanPostProcessor que revisa que todas las anotaciones de los beans manejados por Spring
 - Todas las anotaciones relacionadas con persistencia
 - Tales como las que indican que método setter debería ser inyectado con el EntityManager
 - En la configuración se define el EntityManager
 - Gestiona todas las entidades persistentes
 - Se utiliza para leer, escribir y actualizar los objetos



- Spring – JPA
 - En la configuración se debe indicar como se crea el EntityManagerFactory
 - Se indica cual es el proveedor de la JPA que se utilizará
 - El DataSource asociado al “factory” también se define como un bean de Spring
 - Se especifican los parámetros de conexión a la base de datos
 - Se debe definir el TransactionManager
 - Se asegura que todo ocurra dentro de los límites de transacciones
 - Se asocia al EntityManagerFactory
 - Finalmente se indica a Spring que se utilizará anotaciones para indicar las transacciones



- Revisar:
 - `manning.utils.User`
 - `manning.utils.PortfolioService`
- Definiendo el `EntityManager`
 - Se crea un nuevo objeto para los servicios: `PortfolioServiceJPAImpl`
 - Se debe utilizar la anotación `@PersistenceContext` para indicar a Spring que en ese lugar debe insertar un `EntityManager`
- Revisar `PortfolioServiceJPAImpl`
 - Con la anotación `@Transactional` se indica que todos los métodos en la clase serán transaccionales



- Revisar
 - <http://java.sun.com/javaee/5/docs/api/index.html>
 - <http://java.sun.com/developer/technicalArticles/J2EE/jp>