Spring Framework

Primeros pasos con Spring Boot

En definitiva, un proyecto Spring Boot es solo un proyecto regular de Spring que sucede para aprovechar

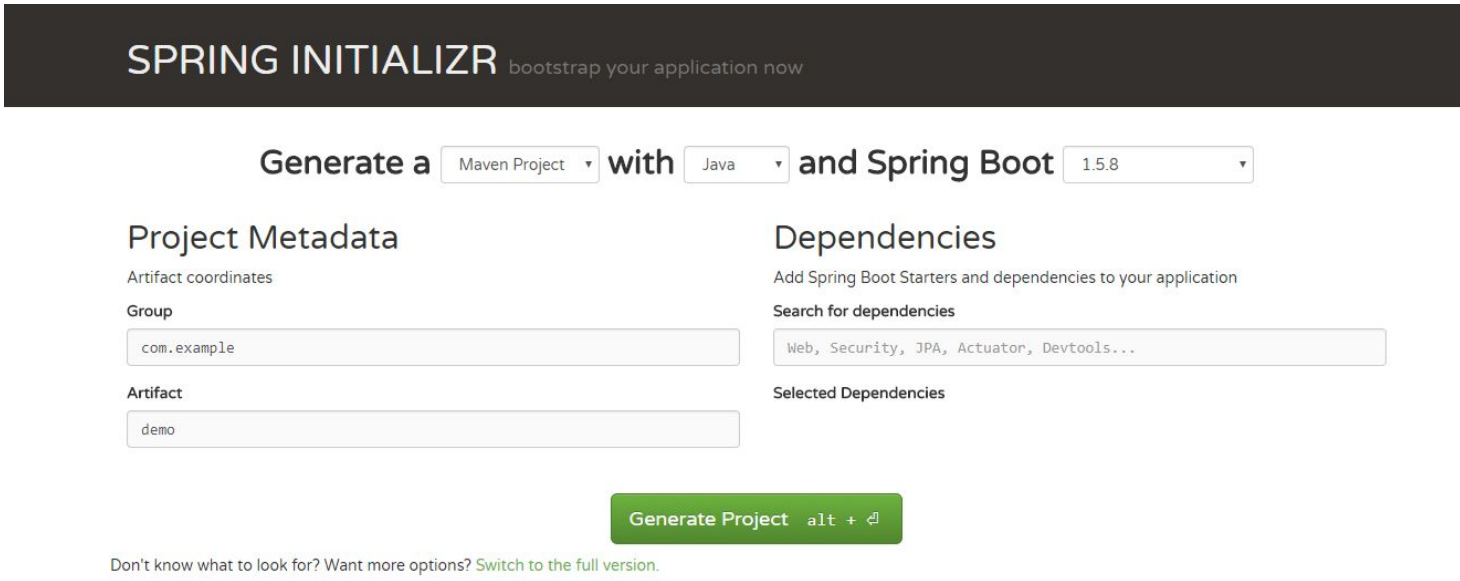
Spring Boot configuraciones automáticas. Por lo tanto, cualquier técnica o herramienta que ya se conozca con la creación de un proyecto Spring desde cero se puede aplicar a un proyecto Spring Boot. Sin embargo, existen algunas opciones convenientes disponibles para comenzar un proyecto con Spring Boot.

La forma más rápida de comenzar con Spring Boot es instalar la CLI de Spring Boot para que se pueda comenzar a escribir código. Para instalar Spring CLI se puede ver los pasos en este link:

https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html

Inicializando un proyecto Spring Boot con Spring Initializr

A veces comenzar es la parte más difícil de un proyecto. Se necesita configurar una estructura de directorio para varios artefactos de proyecto, crear un archivo de compilación y colocar en el archivo compilador dependencias. La CLI Spring Boot elimina gran parte de este trabajo de configuración, pero si favorece una estructura de proyecto Java más tradicional… Entonces tenemos otras opciones como: Spring Initializr.



http://start.spring.io/

Spring Initializr es en última instancia, una aplicación web que puede generar un proyecto, Inicia la estructura del proyecto por nosotros. No genera ningún código para la aplicación, pero nos dará una estructura de proyecto básica y una especificación de construcción: Maven ó Gradle.

Spring Initializr se puede usar de varias maneras:

- A través de una interfaz basada en web.
- Vía Spring Tool Suite.
- Vía IntelliJ IDEA.
- Usando el Spring Boot CLI

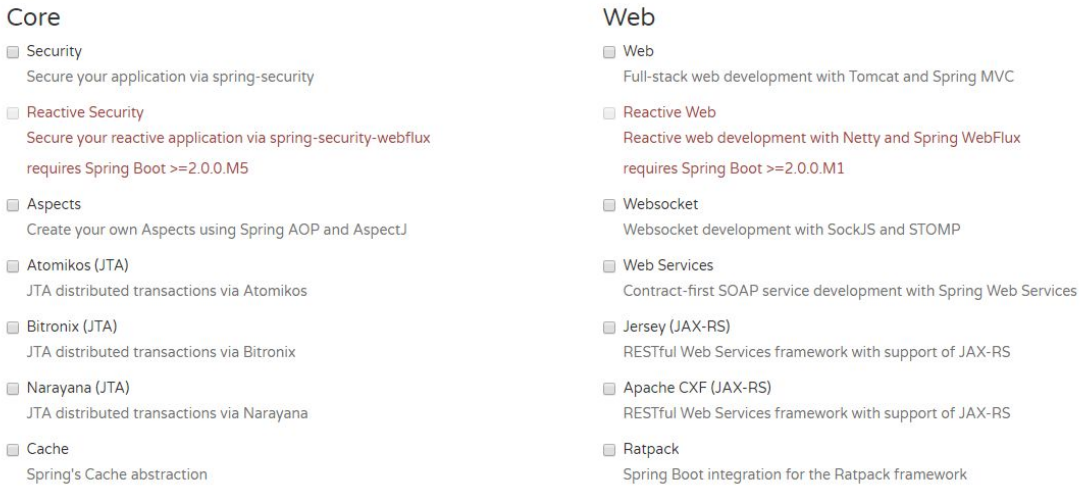
Utilizando la opción web Spring Initializr

La manera más directa de usar Spring Initializr es apuntar su navegador web a http://start.spring.io. Y debería ver un formulario similar al que mostramos en la figura anterior. Las dos primeras cosas que el formulario pregunta es si desea construir su proyecto con: Maven o Gradle y qué versión de Spring Boot usar. Asume por defecto un proyecto Maven utilizando la versión más reciente de Spring Boot, pero puedes elegir uno diferente.

En el lado izquierdo del formulario, se le pide que especifique algunos metadatos del proyecto. Como mínimo, debe proporcionar el grupo y el artefacto del proyecto. Pero si hace clic en "Cambiar a la versión completa ", puede especificar metadatos adicionales como la versión y el paquete base nombre. Estos metadatos se usan para rellenar el archivo Maven pom.xml generado (ó Archivo Gradle build.gradle).

En el lado derecho del formulario, se le pide que especifique las dependencias del proyecto. La forma más fácil de hacerlo es escribir el nombre de una dependencia en el cuadro de texto. A medida que se escribe, un valor aparecerá una lista de dependencias coincidentes. Seleccione el (los) que desea utilizar y serán agregado(s) al proyecto. Si no ve lo que está buscando, haga clic en "Cambiar a versión completa "para obtener una lista completa de las dependencias

disponibles.

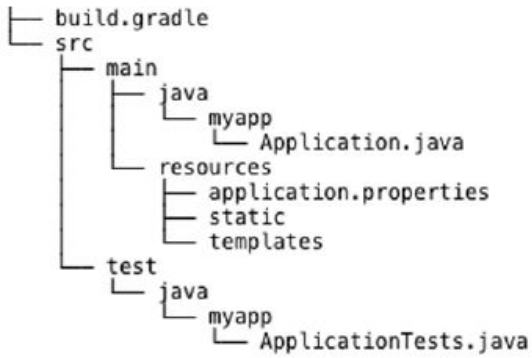


Si miras, reconocerás que las dependencias ofrecidas corresponden a las dependencias iniciales de Spring Boot. De hecho, al seleccionar cualquiera de estas dependencias, le está diciendo a la Initializr que agregue los iniciadores como dependencias para el archivo de compilación del proyecto.

Una vez que haya rellenado el formulario y haya realizado las selecciones de dependencia, haga clic en

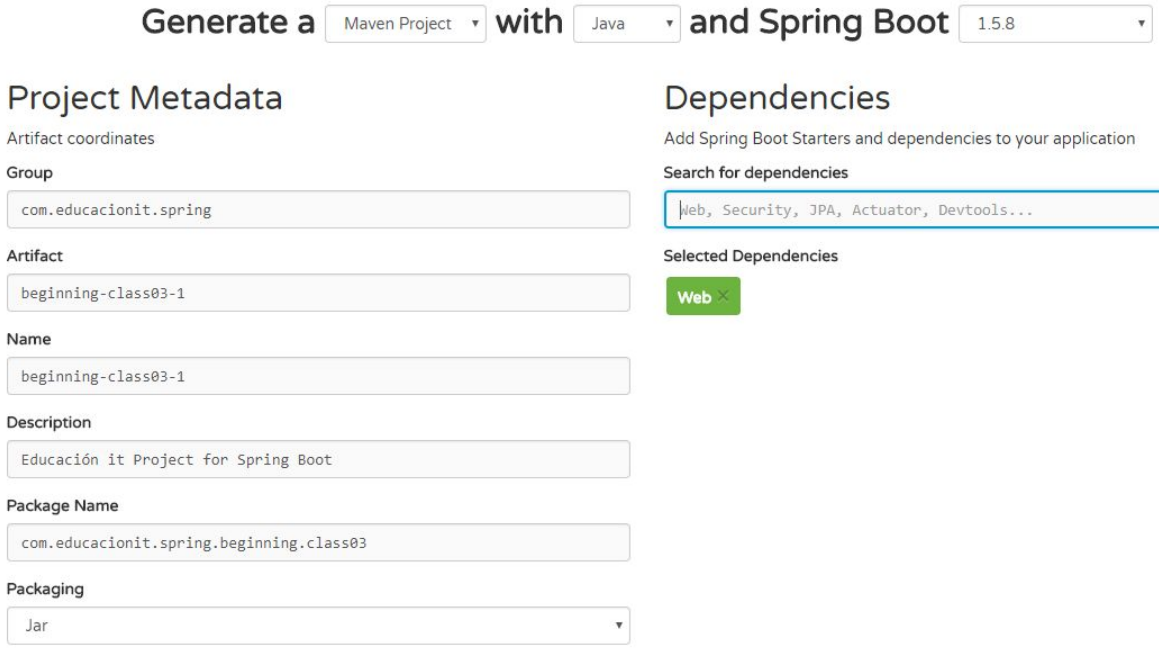
haga clic en el botón Generar proyecto para que Spring Initializr genere el proyecto indicado. El proyecto generado se le presentará como un archivo zip (cuyo nombre está determinado por el valor en el campo Artefacto) que su navegador descarga.

El contenido del zip variará ligeramente, dependiendo de las elecciones que haya realizado antes de hacer clic en Generar. En cualquier caso, el archivo zip contendrá un proyecto básico para comenzar a desarrollar una aplicación con Spring Boot.

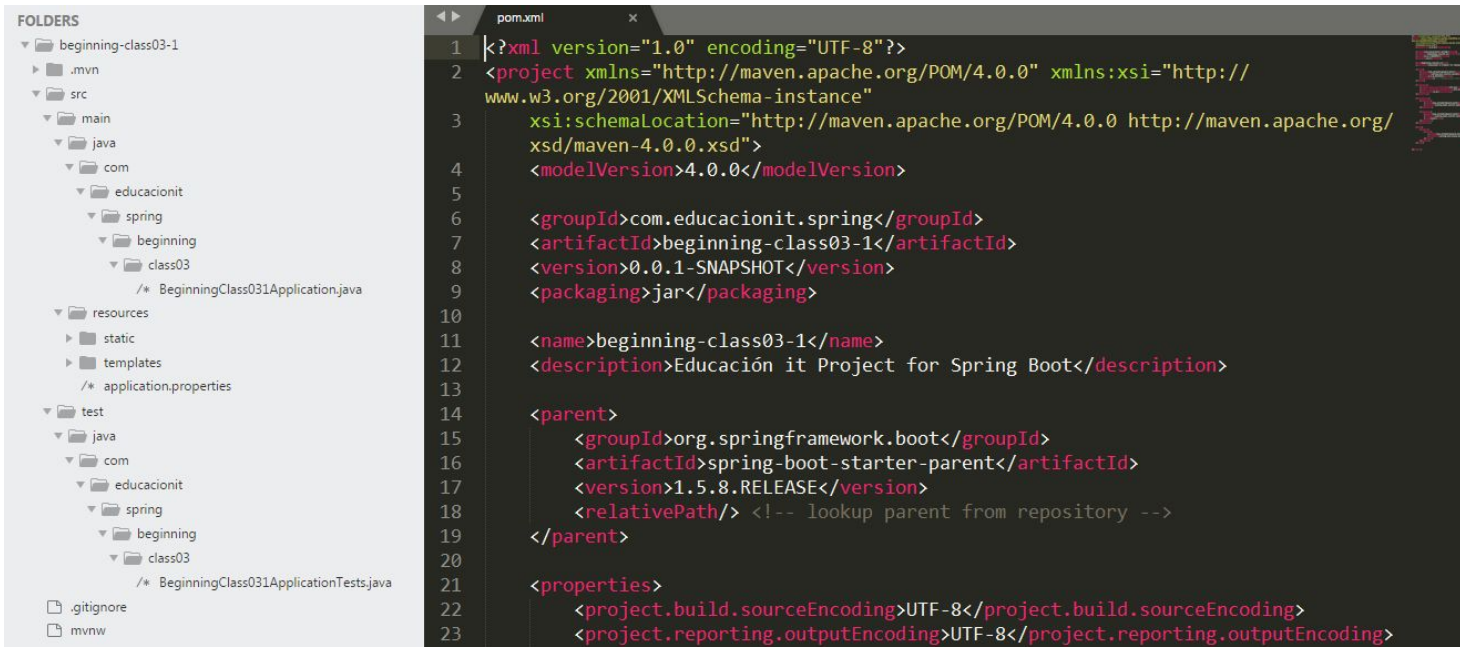


Desarrollando la primera Aplicación Spring Boot

Crearemos a través de Spring Initializr un proyecto con los siguientes valores:



Y una vez generado, al ser descargado deberíamos de tener la siguiente estructura:



Como se está trabajando con Maven o Gradle, somos libre de utilizar el IDE que más nos guste, para este caso nosotros estamos usando un subline text y una ventana de comandos.

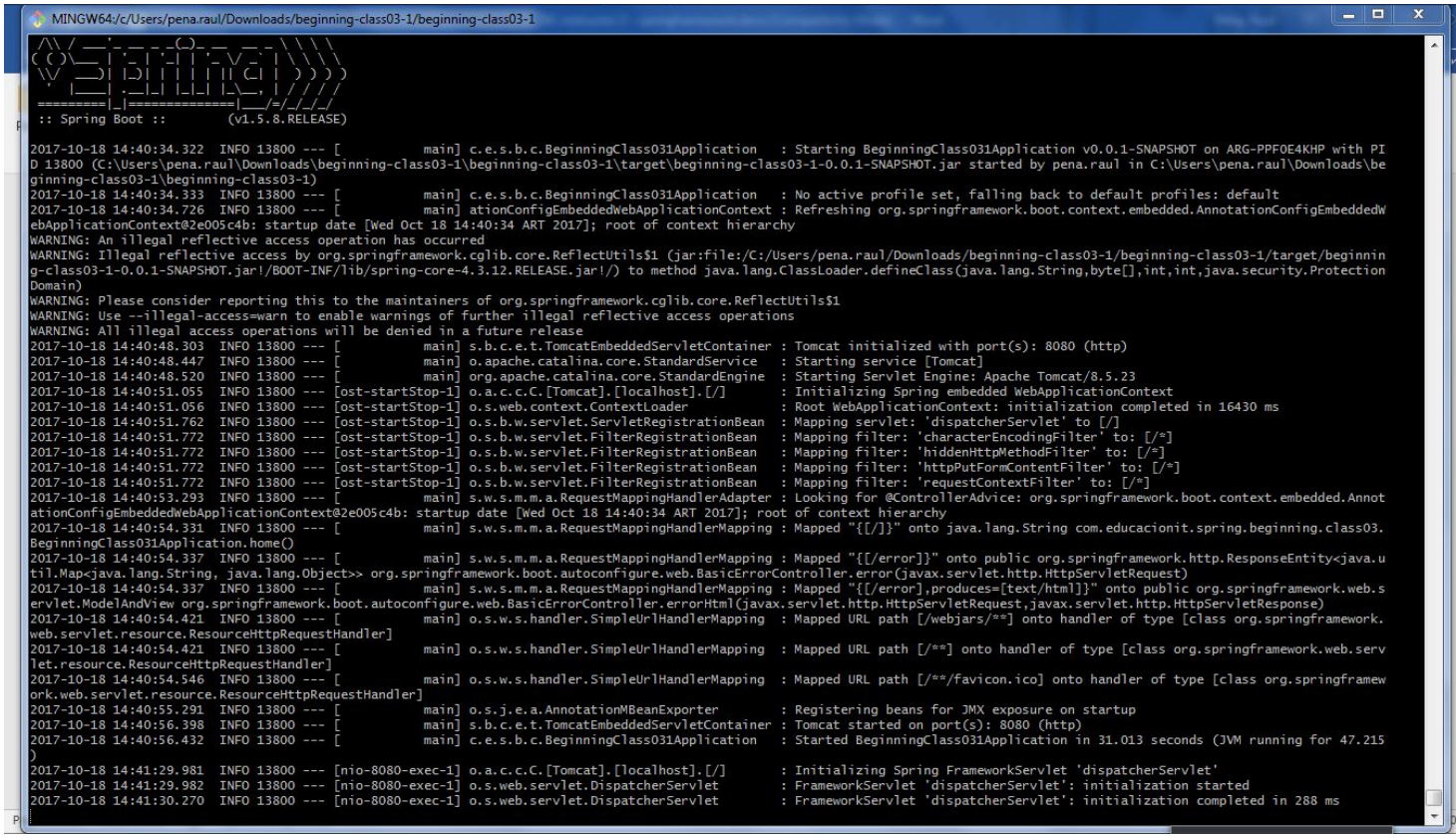
1. Abrir la clase BeginningClass031Application y sustituir el contenido generado por el siguiente:

```
package com.educacionit.spring.beginning.class03;

import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.stereotype.*;
import org.springframework.web.bind.annotation.*;

@Controller
@EnableAutoConfiguration
public class BeginningClass031Application {

        @RequestMapping("/")
    @ResponseBody
    String home() {
        return "Hello World!";
    }

        public static void main(String[] args) {
                SpringApplication.run (BeginningClass031Application.class, args);
        }
}
```

2. Abrir una ventada de comandos y en la raíz del proyecto ejecutar el siguiente comando para compilar la aplicación: mvn clean package -Dmaven.test.skip=true, esto no es necesario hacerlo si se está utilizando un IDE como Eclipse o Netbeans.

3. Una vez compilada la aplicación ejecutar el siguiente comando: java -jar target/beginning-class03-1-0.0.1-SNAPSHOT.jar esto iniciara la aplicación. En el caso de usar un IDE, es simplemente ejecutar la clase BeginningClass031Application desde el IDE.

Si los pasos anteriores tuvieron éxito, deberían de ver este mensaje en la ventana de comandos o en la consola del IDE utilizado:



Luego nos dirigiremos al navegador de nuestra preferencia y colocaremos la siguiente URL: http://localhost:8080/

Y si todo está bien, podremos ver el siguiente mensaje en el navegador:

Hibernate

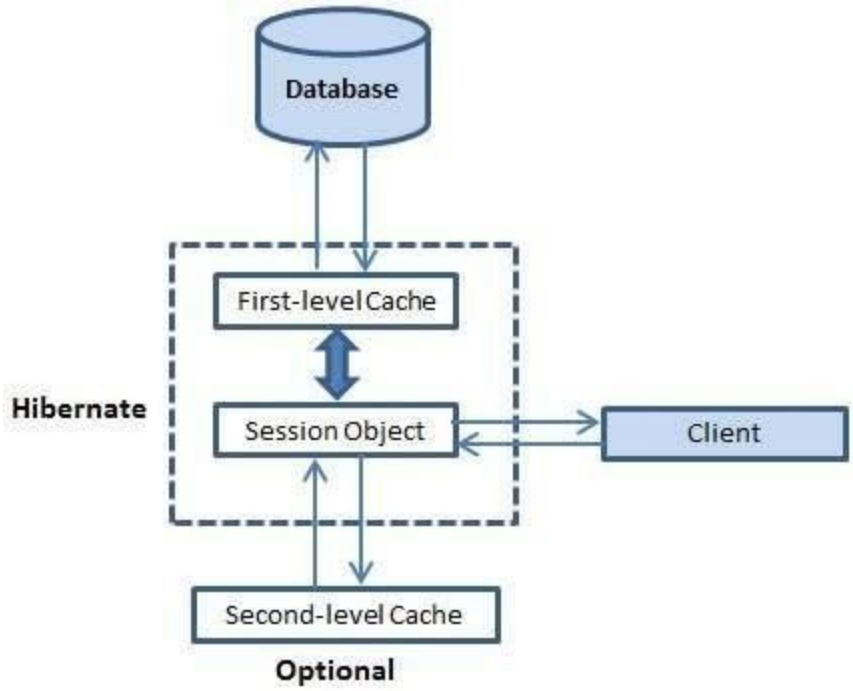
Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.

¿Por qué usar un framework ORM?

Cuando uno desarrolla una aplicación en la gran mayoría de los casos todo termina siendo un conjunto de ABMs (alta, baja, modificación de datos) para luego poder consultarlos. Para ello se utiliza una base de datos, donde existirán muchas tareas repetidas: por cada objeto que quiero persistir debo crear una clase que me permita insertalo, eliminarlo, modificarlo y consultarlo. Salvo aquellas consultas fuera de lo común, el resto es siempre lo mismo. Aquí es donde entra a jugar un rol importante un ORM: con solo configurarlo ya tiene todas esas tareas repetitivas realizadas y el desarrollador solo tendrá que preocuparse por aquellas consultas fuera de lo normal.

¿Cómo funciona Hibernate?

Básicamente el desarrollador deberá configurar en un archivo XML o mediante annotations donde corresponde un atributo de una clase, con una columna de una tabla. Es una tarea relativamente sencilla donde existen herramientas que lo hacen por nosotros.



¿Cómo instalar Hibernate ?

Muy simple… agregando las librerías al classpath, la mejor forma es utilizar Maven o Gradle.

Mapeo de una entidad

Las clases Java deberán tener las siguientes características:

1. Deben tener un constructor público sin ningún tipo de argumentos
2. Para cada propiedad que queramos persistir debe haber un método get/set asociado.
3. Implementar la interfaz Serializable.

Esta clase puede encontrarse completa en el alumni.

Archivo de mapeo ".hbm.xml"

Para cada clase que queremos persistir se creará un archivo ".xml" con la información que permitirá mapear la clase a una base de datos relacional. Este archivo estará en el mismo paquete que la clase a persistir. Para este ejemplo, si queremos persistir la clase Profesor deberemos crear el archivo Profesor.hbm.xml en el mismo paquete que la clase Java.

Nada impide que el archivo .hbm.xml esté en otro paquete distinto al de la clase Java. En este sentido suele haber 2 posibilidades:

1. Almacenar el archivo ".hbm.xml" en el mismo paquete que la clase Java a la que hace referencia.
2. Crear un árbol alternativo de paquetes donde almacenar los archivos ".hbm.xml". Por ejemplo, si tenemos el paquete raíz com.educacionit.spring.beginning donde se guardan todas las clases Java a persistir, crear otro paquete llamado com.educacionit.spring.beginning.dao donde almacenar los archivos .hbm.xml.

La ventaja de la segunda opción es que en caso de que no queramos usar Hibernate, simplemente hay que borrar toda la carpeta com.educacionit.spring.beginning.dao y ya está, mientras que la ventaja de la primera opción es que la clase Java y su correspondiente archivo de mapeo están más juntos facilitando en caso de algún cambio en la clase Java el cambio en el archivo de mapeo.

Esta clase puede encontrarse completa en el alumni.

Anotaciones

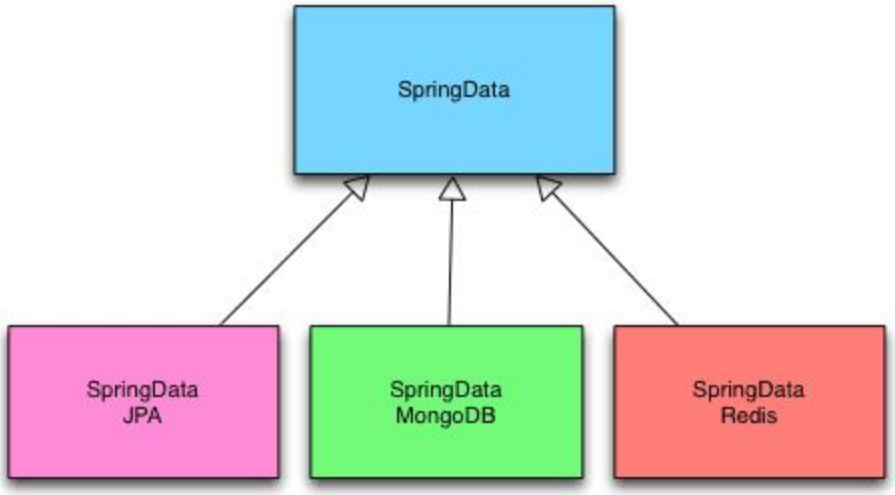
Anteriormente hemos visto cómo mediante un Archivo ".hbm.xml" podemos especificar cómo mapear la clases Java en tablas de base de datos.

Desde Java 5.0 se creó el concepto llamado anotaciones Java en el propio código. Estas anotaciones permiten especificar de una forma más compacta y sencilla la información de mapeo de las clases Java. Inicialmente Hibernate creó sus propia anotaciones en el paquete org.hibernate.annotations pero a partir de la versión 4 de Hibernate la mayoría de dichas anotaciones han sido java.lang.Deprecated y ya no deben usarse. Las anotaciones que deben usarse actualmente son las del estándar de JPA que se encuentran en el paquete javax.persistence. Sin embargo hay características específicas de Hibernate que no posee JPA lo que hace que aun sea necesario usar alguna anotación del paquete org.hibernate.annotations pero en ese caso Hibernate 4 no las ha marcado como java.lang.Deprecated

Para ver la clase Profesor anotada pueden conseguirla en el alumni.

Spring Data

Spring Data es uno de los frameworks que se encuentra dentro de la plataforma de Spring. Su objetivo es simplificar al desarrollador la persistencia de datos contra distintos repositorios de información.



El acceso a bases de datos es una de las tareas más comunes en el desarrollo de software, al principio esta tarea se realizaba simplemente haciendo uso de JDBC y poco a poco fue evolucionando utilizando patrones de diseño como el DAO y a través de frameworks y API's tales como Hibernate y JPA, estos ORM nos han ayudado a reducir muchas de las tareas que se hacían antes con JDBC.

Spring Data es un módulo de Spring que vamos a utilizar sobre JPA para hacer las tareas de acceso a base de datos aún más sencillas, las ventajas se notarán a simple vista ya que nosotros simplemente crearemos métodos en una interfaz y Spring Data se encargará de hacer las implementaciones por nosotros de tal modo que si nombramos un método como "findByName" Spring Data creará la implementación necesaria para buscar en la base de datos a través del nombre sin que nosotros creemos ni una sola conexión ni procesemos ningún resultado.