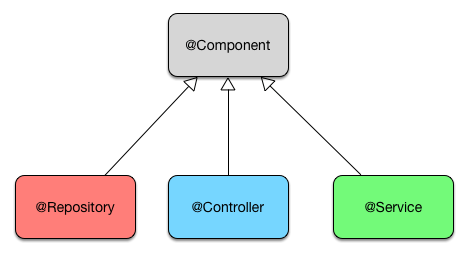
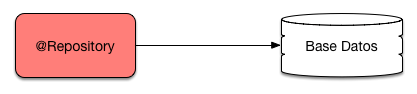
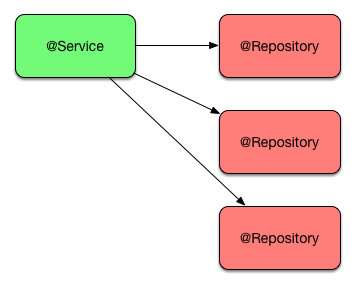
@Component: Es el estereotipo general y permite anotar un bean para que Spring lo considere uno de sus objetos.



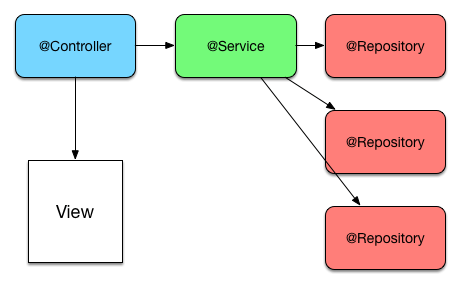
@Repository: es un anotador que da de alta un bean para cualquier clase que sea de acceso a DATOS o persistencia, Al marcar el bean con esta anotación Spring aporta servicios transversales como conversión de tipos de excepciones.



@Service: Esta anotación se encarga de gestionar las reglas de negocio más importantes a nivel de la aplicación y hace llamadas a varios repositorios (@Repository) de forma simultánea. Su tarea fundamental es la de agregador.



@Controller : La última de las anotaciones que realiza las tareas de controlador y gestión de la comunicación entre el usuario y el aplicativo. Para ello se apoya habitualmente en algún motor de plantillas o librería de etiquetas que facilitan la creación de páginas (vistas).



REST es un mecanismo para el intercambio y manipulación de recursos a través de Internet. Es semejante a los Web Services, pero no usa un protocolo concreto para el intercambio de la información.

Inicialmente REST se describe en el contexto de HTTP (aunque no se limita sólo a este protocolo), por lo que se aprovecha de todas sus características: URIs, tipos de media, sesiones, seguridad, cache, etc.

Las operaciones típicas que podemos hacer con REST son:

* GET, para recuperar un recurso. Es idempotente, es decir si la ejecutamos más de una vez siempre devuelve el mismo recurso.
* POST, para añadir recursos. No es idempotente, es decir si la ejecutamos dos veces estaremos añadiendo dos recursos.
* PUT, para modificar un recurso. Es idempotente, si la ejecutamos más de una vez la modificación es siempre la misma (por ejemplo cambiar el nombre de una persona en una agenda).
* DELETE, para borrar un recurso. Es idempotente, si lo ejecutamos más de una vez el resultado es siempre el mismo: el recurso deja de estar en el sistema (la primera vez se borra realmente, las siguientes veces simplemente se ignora la petición, pero no da error).

Otros verbos que tenemos disponibles son:

* HEAD, para pedir un recurso sin recuperarlo, nos sirve para saber si existe o para traer meta información sobre el recurso.
* OPTIONS, para preguntarle a un servidor sobre que otros verbos son aplicables a un recurso determinado.
* PATH, es de reciente adopción (en el 2010), es un intento de expresar de forma estándar actualizaciones parciales de un recurso.

Goals

* Clean: borra el directorio de salida (target)
* Compile: compila el proyecto y deja el resultado en target/clases
* Package: empaqueta el proyecto y lo dejará en target/rSNAPSHOT.jar
* Install: guarda el proyecto en el repositorio

Scopes

* Compile: es el valor por defecto. Se utiliza en todos los casos (compilar, ejecutar, …).
* Provided: también se utiliza en todos los casos, pero se espera que el jar sea suministrado por la JDK o el contenedor. Es decir, no se incluirá al empaquetar el proyecto, ni en el repositorio.
* Runtime: no se utiliza para compilar, pero si es necesario para ejecutar.
* Test: sólo se utiliza para compilar o ejecutar los test.
* System: es similar a provided, pero eres tú el que tiene que suministrar el jar. No se incluirá al empaquetar el proyecto, ni en el repositorio.
* Import:

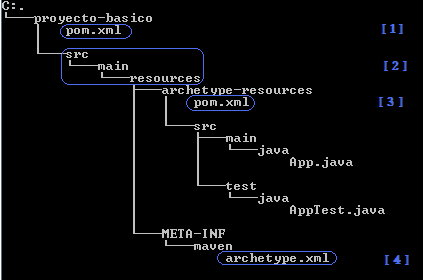
Archetypes: son plantillas, parametrizadas o configuradas para utilizar determinadas tecnologías, que los programadores utilizan como base para escribir y organizar el código de la aplicación.

Estructura básica de un arquetipo de maven

Todo arquetipo se compone de:

* Un pom.xml a nivel raíz del arquetipo, necesario para la construcción del mismo
* Los ficheros que compondrán el cuerpo del arquetipo. Se sitúan bajo src/main/resources/archetype-resources/
* Los pom.xml de los módulos que componen el prototipo del arquetipo
* El descriptor de arquetipos archetype.xml, que se sitúa en el directorio src/main/resources/META-INF y que indica al mecanismo de generación de arquetipos todo el contenido del que estamos definiendo.

Podemos ilustrarlo con la siguiente estructura de un arquetipo de maven básico:



Artifact: es el archivo JAR (por defecto) que se produce con el maven build. Cada artifact tiene un groupID, artifactID y una versión.

