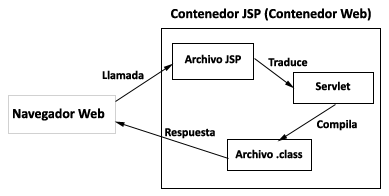
**3.3 Spring Framework:**

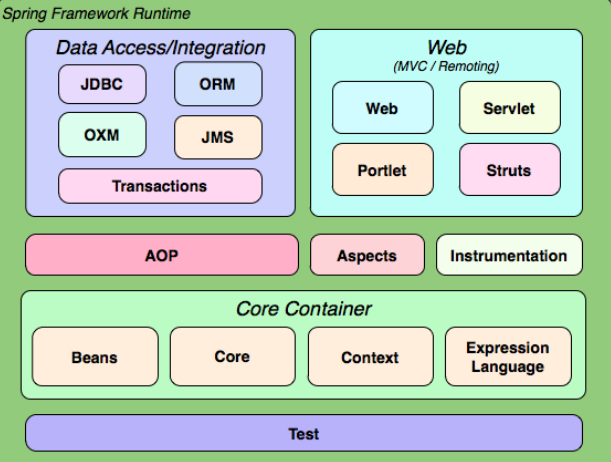
Es un modelo de aplicación web habitual sin hacer uso de algún framework en la web, el cliente hace peticiones mediante HTTP al servidor, el servidor tendrá servlets para la respuesta del cliente, estarán guardadas por un numero de servlets solicitados.

Es un conjunto de clases y soluciones para el fin de estandarizar, agilizar y resolver problemas.  
El framework spring permite desarrollar aplicaciones de manera mas eficaz ahorrando líneas de código.

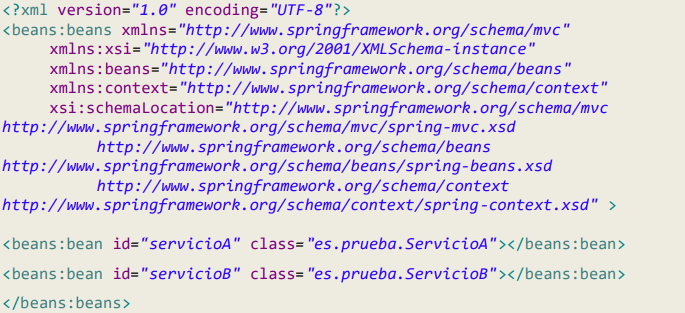


Llevará a cabo la creación y destrucción de las instancias de los objetos de relación a como se definan en el contexto. Se encontrarán los siguientes ámbitos de creación de Bean:

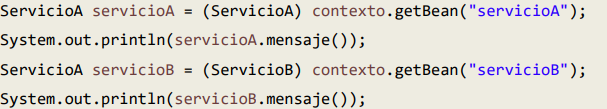
* **Singleton:** Es el ámbito por defecto de spring es decir que no se especifica la creación de Bean, solo spring se encargara de eso, por lo que siempre se solicita que el vean este inyectando el mismo objeto.
* **Prototye:** El contenedor de spring crea una instancia de objeto descrito por el vean cada vez que sea solicitado, en algunos casos es necesario, pero se toma en cuenta que no se debe abusar por que puede haber una perdida de rendimiento en la aplicación.
* **Request:** El contenedor de spring creara una nueva instancia del objeto definido por el vean cada ves que reciba u HTTP request
* **Session:** El contenedor de spring crea una nueva instancia del objeto definido por el vean para las sesiones HTTP y entrega la misma instancia cada vez que reciba una petición dentro de la misma sesión.

**3.3.1 Módulos:**Te permite crear e inyectar beans de cara al diseño de inversión del control (IoC) que proporciona spring. Consiste en que especifica respuestas y acciones deseadas ante cualquier tipo de entidad o arquitectura externa para su ejecución, sin necesidad de preocuparse la manera en la que se realiza.  
Otro modulo importante es el web, spring integra en este modulo el modelo vista controlador (MVC) y la definición interna de servlets.

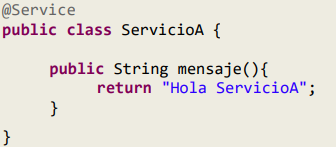
**3.3.2Creacion de contextos y bean:**El principal componente del frameork spring es el contenedos de inversión de control (ioc), es el encargado para administrar los beans, un bean es un objeto Java que es administrado por spring la cual instancia, inicializa y destruye objetos, también realiza otras tareas como la inyección de dependencias (DI). Para poder usar el contenedor se configura la forma tradicional a travez de un archivo XML de configuración de spring, es posible utilizar anotaciones y código java.  
Una vez definido el contexto forma un cuerpo añadiéndole beans a través de etiquetas y pueden ser usados en otra parte de la aplicación, el XML se mostrará de la siguiente manera:



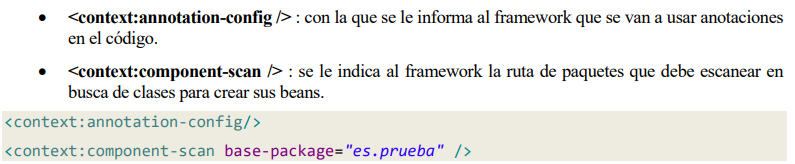
Se aporta una ID para acceder a ella:



Si no se declara un gran numero de variables, el fichero XML se hace pesado en el manejo y lectura. Se soluciona por el spring que da la posibilidad de usar anotaciones en clases para obtener el mismo resultado, pero más cómodo y se usa la etiqueta @Service.



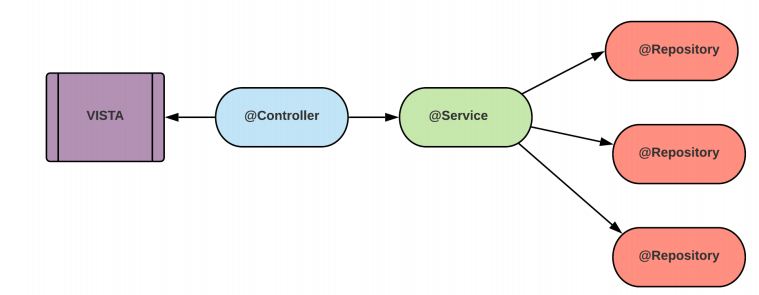
Se debe indicar al XML que cargue las anotaciones creadas, para eso se usan 2 etiquetas:



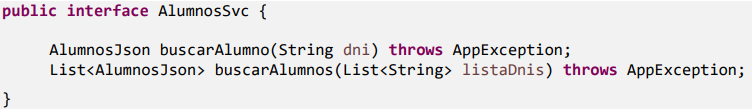
**3.3.3 Spring estereotipos y anotaciones**Tiene muchas etiquetas y anotaciones para categorizar componentes dentro de un código.

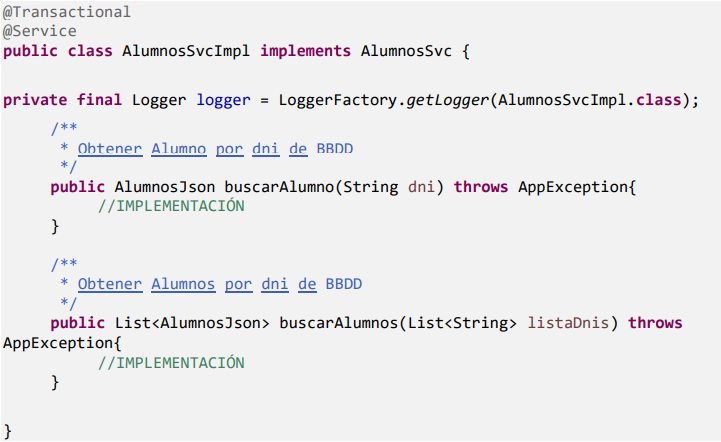
**Estereotipos:**

**@component:** Indica que la clase anotada es un componente o bean de spring  
@controller @service @repository  
**@respository:** Tiene como función dar de alta un bean para que implemente el patrón repositorio, este se encarga de almacenar datos. Al marcar el bean con esta anotación, Spring porta servicios transversales como conversión de tipos de excepciones  
**@service:** Se encarga de gestionar las operaciones de negocio mas importantes a nivel de la aplicación y aglutina llamadas a varios repositorios de forma simultánea. Es el agregador.  
**@controller:** Es el que realiza tareas de controlador y gestión de la comunicación entre el usuario y el aplicativo. Se apoya en algún motor de plantilla o librería de etiquetas para facilitar la creación de páginas.

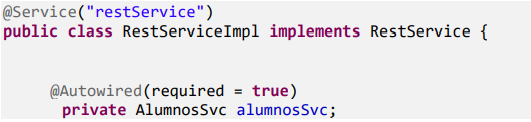


**Anotaciones:**   
**@Autowired:** Se puede inyectar un componente como ser un servicio o un bean.

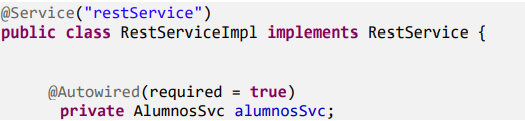
Se implementan métodos de interfaz:



Usando @autowired



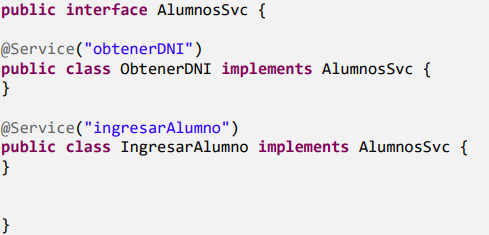
Usando la interfaz en otra clase



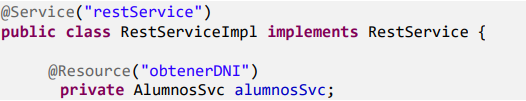
Hacer uso de propiedad y métodos:



Si tiene métodos declarados con @service, se puede especificar la inyeccio con @qualifier



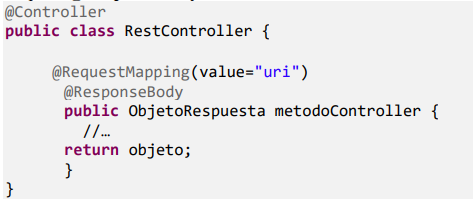
Hay otro recurso que serviría usando **@Resoursce**



Con la etiqueta @Restcontroller facilita el escenario de un API REST. Donde se debe configurar un servlet de escucha en el fichero web.xml



Otra manera es usando @ResponseBody



**3.4.1 Anotaciones principales:**Una entidad se caracteriza por ser una clase de primer nivel, proporciona un constructor e implementa la interfax java.io. Entidad hace referencia a un objeto POJO (Plain Old Java Object). Cualquier entidad JPA presenta numerosas etiquetas que la caracterizan, para esto tenemos las siguientes entidades:

 @Entity: informa al proveedor de persistencia que cada instancia de esta clase es una entidad.   
 @Table: permite configurar el nombre de la tabla que se está mapeando con dicha entidad. Para ello se hace uso del atributo name.   
 @Colum: acompañará a cada uno de los atributos de la entidad y permitirá configurar el nombre de la columna a la que dicho atributo hace referencia dentro del sistema relacional. Esta etiqueta posee numerosos atributos, donde se puede indicar al framework propiedades que debe tener en cuenta para la columna.   
 @Id: acompañará aquel atributo que permita diferenciar las distintas entidades de manera que cada una sea única, normalmente acompaña al que se asocie a la clave primaria de la tabla que se esté mapeando.  
  @JoinColumn: permite configurar aquel atributo que contiene una clave foránea a otra tabla.   
 @OneToMany: esta etiqueta irá acompañando a la anterior en el caso de que el atributo de la entidad puede referenciar a más de un atributo de la tabla a la que hace referencia. A la hora de obtener estos valores en una consulta, se pueden diferenciar dos tipos:  
Eager: Se define como lectura temprana, en la consulta del objeto se obtiene todos los valores de las entidades que están en relación con la entidad.  
Lazy: Se define como lectura demorada, permite que un objeto de base de datos sin los valores de la relación hacer referencia a un tributo.

**3.5 Hibernate**Es un framework de persistencia de software que implementa la gestión de la API de Java. Aporta al usuario escalabilidad, eficiencia y facilidad para hacer consulta a la base de datos.   
Esta diseñado para adaptarse a cualquier esquema y modelo de datos, así obtiene un modelo de tablas relacional a partir de la definición de entidades en JAVA.  
Presenta un sistema de doble cache que es totalmente configurable. Se definen en nivel 1 y nivel 2.  
El primer nivel: Presenta automáticamente hibernate cuando en una transacción se interactúa la base de datos. Se considera un cache de corta duración ya que es válido entre el begin y el commit de una transacción.

El segundo nivel: Puede ser configurado de cara a la mejora del rendimiento. La diferencia es que este tipo de cache es valida para todas las transacciones y persiste en la memoria durante todo el tiempo en el que el aplicativo este online y es global.

Hibernate Genera consultas SQL y libera el desarrollador mas primario, posee un lenguaje denominado HQL (hibernate query lenguaje) y de una API para que construya consultas. Denominado “criteria”

**3.5.1 Criteria**

Es una API creada por hibernate para facilitar consultas en una base de datos. Una de las ventajas principales es que la forma de construir querys es orientado a objetos. Es muy útil ya que da soporte a un formulario de búsqueda y nos evitara líneas de comprobación para cada campo haciendo el código mucho mas corto.

Se crea una instancia de criteria para un objeto haciendo uso de métodos de esa instancia y se da forma a las restricciones de búsqueda en torno a ese objeto.

Crit = sess.createCriteria(Cat.class);

Se limita el número de resultados:

Crit.setMaxResults(50);

Se añade criterios individuels:

Crit.add(Restrictions.like(“name”,”Fritz%”))  
Crit.add(Restrictions.btween(“weight”, minWeight, maxWeight))

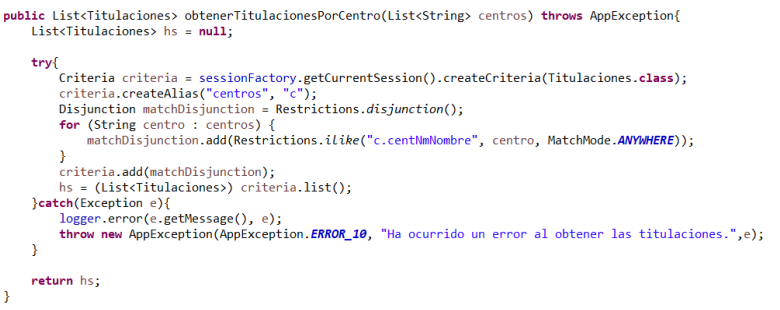
Para disyunciones:

crit.add( Restrictions.or( Restrictions.eq( "age", new Integer(0) ), Restrictions.isNull("age") ) )

ordenar:

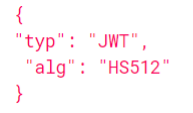
crit.addOrder( Order.asc("name") )

Con todo esto se consigue cualquier consulta básica, es extenso y en esto se puede introducir código sql, crear proyecciones o formar subconsultas:

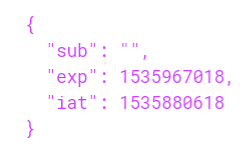


**3.6 JWT (Json web token)**   
Esta basada en token en una referencia en desarrollo de aplicaciones web presentan una ventaja que es la de guardar sesión para los datos de un usuario.   
En la autenticación con token, el usuario identifica bien con un usuario/contraseña o una ID y la web le devuelve una especie de cifra firmada para que el usuario use en las cabeceras de las peticiones HTTP.  
El token se envía como respuesta a la aplicación, tiene un tiempo de vida al cual debe configurarse acorde a lo que interese.  
Esta formado por 3 cadenas:

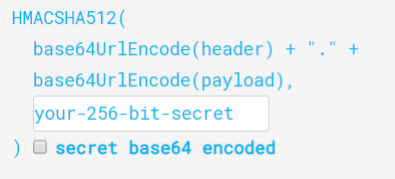
* **Header:** Es la primera parte del token, esta dormado por el tipo de token y el algoritmo de codificación utilizado:



* **Payload:** Está compuesto por atributos llamados Claims, existen:   
  - iss: especifica la tarea para la que se va a usar el token.   
  - sub: presenta información del usuario.  
   - aud: indica para que se emite el token. Es útil en caso de que la aplicación tenga varios servicios que se quieren distinguir.   
  - iat: indica la fecha en la que el token fue creado.   
  - exp: indica el tiempo de expiración del token, se calcula a partir del iat.  
   - nbf: indica el tiempo en el que el token no será válido hasta que no transcurra.   
  - jti: identificador único del token. Es utilizado en aplicaciones con diferentes proveedores. Los más comunes son sub, iat y exp. Para el proyecto en cuestión, en el que no se hace login por usuario:



* **Signature:** La firma es la ultima de las 3 partes esta formada por el header y el payload codificada en base 64, mas una clave secreta.



Las 3:

