SERVICIOS WEB

NOMBRE : Nicolas Gonzalo Aguilar Arimoza
CURSO : 4 SEMESTRE
FECHA : 27/04/2020
MATERTIA : PROGRAMACION 3

1. SERVICIOS WEB
    - TIPOS DE SERVICIOS WEB
    - CARACTERISTICAS DE SOAP
    - CARACTERISTICAS DE REST
    - COMPARATIVA ENTRE RES Y SOAP
2. MAVEN
    - CICLO DE VIDA
    - POM
    - EL REPOSITORIO MAVEN
3. SPRING FRAMEWORK
    - MODULOS
    - CREACION DE CONTEXTO Y BEANS
    - SPRING ESTEREOTIPOS Y ANOTACIONES
4. JPA (JAVA PERSISTENCE API)
    - ANOTACIONES PRINCIPALES
5. HIBERNATE
    - CRITERIA
6. JWT (JSON WEB TOKEN)

1. SERVICIOS WEB

El consorcio W3C (World Wide Web Consortium) define un servicio web como un sistema software diseñado para soportar la interacción máquina-a-máquina, a través de una red, de forma interoperable.

• TIPOS DE SERVICIOS WEB

Existen diferentes tipos de servicios web como :

- REMOTE PROCEDURE CALLS

Están basados en RPC y presentan una interfaz llamada procedimientos remotos y funciones distribuidas también es una comunicación de nodo a nodo entre cliente y servidor donde el cliente pide la ejecución de cierto procedimiento o función y los servidores envían la respuesta correcta.

- SERVICE ORIENTED ARCHITECTURE

Es una arquitectura de una aplicación en la cual todas las funciones están definidas como servicios independientes con interfaces invocables que puede ser llamados en secuencias bien definidas para tomar los procesos de negocio.

- REST

Son un conjunto de principios de arquitectura para describir cualquier interfaz entre sistemas que utiliza directamente HTTP ara la obtención de datos o indicar la ejecución de operaciones sobre los datos.

• CARACTERISTICAS DE SOAP

SOAP es un protocolo que se encarga del intercambio de información con el servidor web de las cuales un servicio web se puede construir, este protocolo esta basado en XML y formado por tres partes:

- Envelope: el cual define qué hay en el mensaje y cómo procesarlo.
- Conjunto de reglas de codificación para expresar instancias de tipos de datos.
- La convención para representar llamadas a procedimientos y respuestas.

En la arquitectura de un servicio web que implementa este protocolo también se pueden diferenciar tres partes:

- Proveedor del servicio.
- Solicitante.
- Publicador.

- # CARACTERISTICAS DE REST

REST se ha centrado en explotar el éxito de la Web, que no es más que el uso de formatos de mensaje extensibles, estándares y un esquema de direccionamiento global. En particular, el concepto central de la Web es un espacio de URIs unificado. Las URIs identifican recursos, los cuales son objetos conceptuales. La representación de tales objetos se distribuye por medio de mensajes a través de la Web. Este sistema es extremadamente desacoplado.

- **Escalabilidad**. La variedad de sistemas y de clientes crece continuamente, pero cualquiera de ellos puede acceder a través de la Web. Gracias al protocolo HTTP, pueden interactuar con cualquier servidor HTTP sin ninguna configuración especial.
- **Independencia**. Los clientes y servidores pueden tener puestas en funcionamiento complejas. Diseñar un protocolo que permita este tipo de características resulta muy complicado. HTTP permite la extensibilidad mediante el uso de las cabeceras, a través de las URIs.
- **Compatibilidad**. En ocasiones existen componentes intermedios que dificultan la comunicación entre sistemas, como pueden ser los firewalls. Las organizaciones protegen sus redes mediante firewalls y cierran casi todos los puertos TCP salvo el 80, el que usan los navegadores web. REST al utilizar HTTP sobre Transmission Control Protocol (TCP) en el puerto de red 80 no resulta bloqueado. Es importante señalar que los servicios web se pueden utilizar sobre cualquier protocolo, sin embargo, TCP es el más común.
- **Identificación de recursos**. REST utiliza una sintaxis universal como es el uso de URIs. HTTP es un protocolo centrado en URIs, donde los recursos son los objetos lógicos a los que se le envían mensajes.
- **Protocolo cliente/servidor sin estado**. Cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes. Sin embargo, en la práctica, muchas aplicaciones basadas en HTTP utilizan cookies y otros mecanismos para mantener el estado de la sesión
- **Operaciones bien definidas**. HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE.

- COMPARATIVA ENTRE RES Y SOAP

REST	SOAP
• Las operaciones se definen en los mensajes. • Una dirección única para cada instancia del proceso. • Cada objeto soporta las operaciones estándares definidas. • Componentes débilmente acoplados. • Bajo consumo de recursos. • Las instancias del proceso son creadas explícitamente. • El cliente no necesita información de enrutamiento a partir de la URI inicial. • Los clientes pueden tener una interfaz "listener" (escuchadora) genérica para las notificaciones. • Generalmente fácil de construir y adoptar.	• Las operaciones son definidas como puertos WSDL. • Dirección única para todas las operaciones. • Múltiples instancias del proceso comparten la misma operación. • Componentes fuertemente acoplados. • Fácil (generalmente) de utilizar. • La depuración es posible. • Las operaciones complejas pueden ser escondidas detrás de una fachada. • Envolver APIs existentes es sencillo • Incrementa la privacidad. • Herramientas de desarrollo.

2. MAVEN

Maven es una herramienta para la gestión y construcción de proyectos java, que se basa en el concepto POM (Proyect Object Model). Con Maven se pueden generar arquetipos, gestionar librerías, compilar, empaquetar.

Se basa en patrones y estándares y trabaja con arquetipos, los cuales son configurables a través de su fichero protagonista.

- CICLO DE VIDA

Validación (validate): Validar que el proyecto es correcto. – Compilación (compile).

– Test (test): Probar el código fuente usando un framework de pruebas unitarias.

– Empaquetar (package): Empaquetar el código compilado y transformarlo en algún formato tipo .jar

– Pruebas de integración (integration-test): Procesar y desplegar el código en algún entorno donde se puedan ejecutar las pruebas de integración.

– Verificar que el código empaquetado es válido y cumple los criterios de calidad (verify).

– Instalar el código empaquetado en el repositorio local de Maven, para usarlo como dependencia de otros proyectos (install).

– Desplegar el código a un entorno (deploy).

*Ciclos de limpieza

clean: elimina todos los ficheros generados por construcciones anteriores.

*Ciclo de documentación

site: genera la página web de documentación del proyecto.

site-deploy: despliega la página de documentación en el servidor indicado.

- POM

POM es un modelo de objeto para un proyecto, o como describe la documentación de Maven:

"El fichero "pom.xml" es el núcleo de configuración de un proyecto Maven. Simplemente es un fichero de configuración, que contiene la mayoría de la información necesaria para construir (build) un proyecto al gusto del desarrollador"

Las etiquetas mas usadas por pom son:

GROUPLD: Suele ser el nombre o la web de la organización. Aunque no es necesario que tenga.

ARTIFACTLD: el nombre del artefacto.

VERSION: La versión de del proyecto. Por defecto se suele usar "1.0.0". Aunque es posible usar la metodología de versiones que más nos guste.

PACKAGING: Con ella se indica cómo se desea que sea empaquetado el proyecto cuando Maven lo construya. Si por ejemplo se usa como valor "jar", se nos creará una biblioteca de Java. Otro ejemplo sería definirlo como "war", que sería un empaquetado web para desplegar en un servidor.

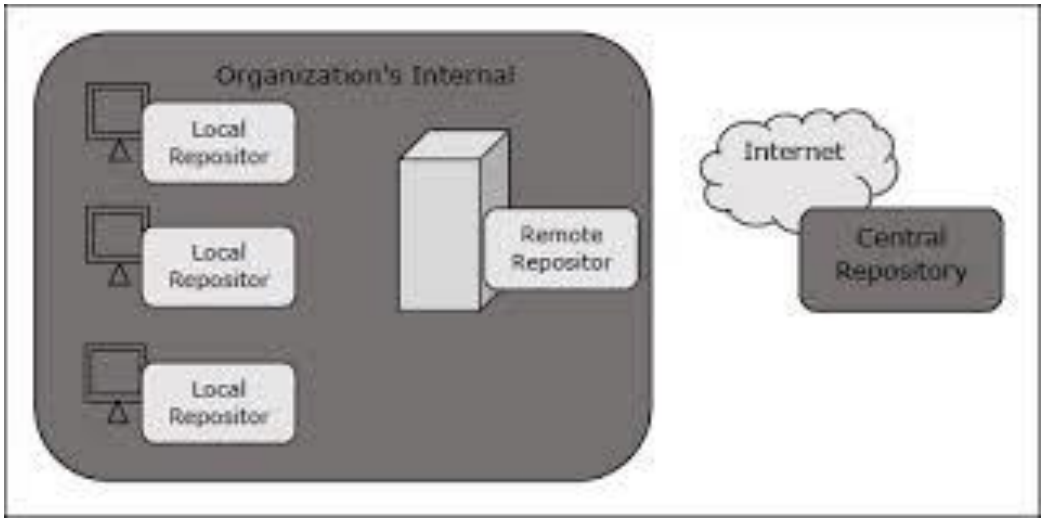
REPOSITORIES: se indica cuáles serán los repositorios en los que Maven debe buscar las librerías para nuestro proyecto.

DEPENDENCIES: engloba a todas sus etiquetas hijas **DEPENDENCY** en las que se definen los artefactos que quiere que Maven descargue e incorpore a nuestro proyecto.

GROPULD,ARTIFACLD,VERSION: es posible identificar la biblioteca y la versión. Además, en la segunda de ellas se ve como se hace uso de una de las variables globales que se han definido anteriormente en la etiqueta PROPERTIERS .

- EL REPOSITORIO MAVEN

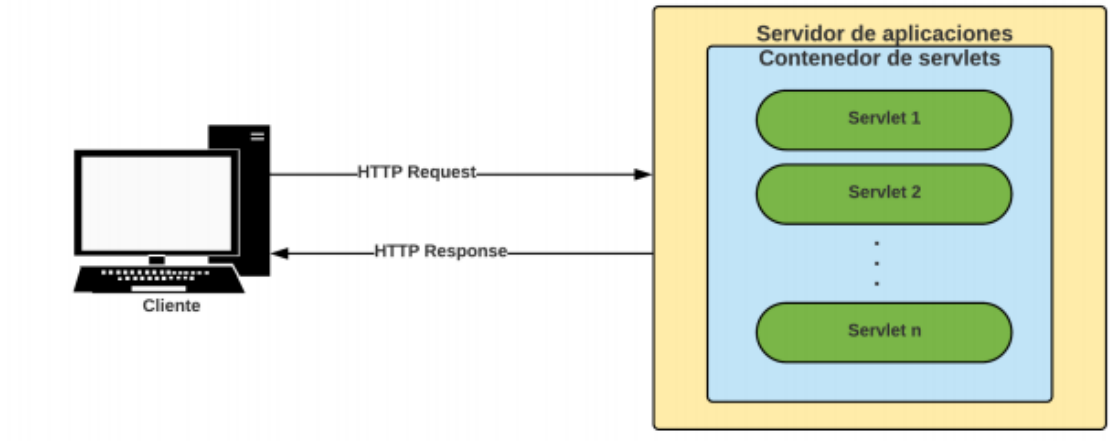
Una vez definidas todas las dependencias, Maven las descarga y las guarda todas en un mismo repositorio, generalmente /.m2/repository. Aunque se puede cambiar si se desea.



3. SPRING FRAMEWORK

Framework es un conjunto de clases y soluciones para su uso con el fin de estandarizar, agilizar y resolver los problemas también nos permite desarrollar aplicaciones de manera más eficaz y rápida, ahorrando a su vez muchas líneas de código.

En varios casos el cliente envía peticiones HTTP al servidor y éste tendrá configurados distintos servlets para dar una respuesta al cliente. De esta manera, la aplicación estará formada por un número determinado de servlets que desarrollarán las distintas funcionalidades definidas para la misma.



llevará a cabo la creación y destrucción de las instancias de los objetos en relación a cómo se definan en el contexto. De esta manera se encuentran los siguientes ámbitos de creación de bean:

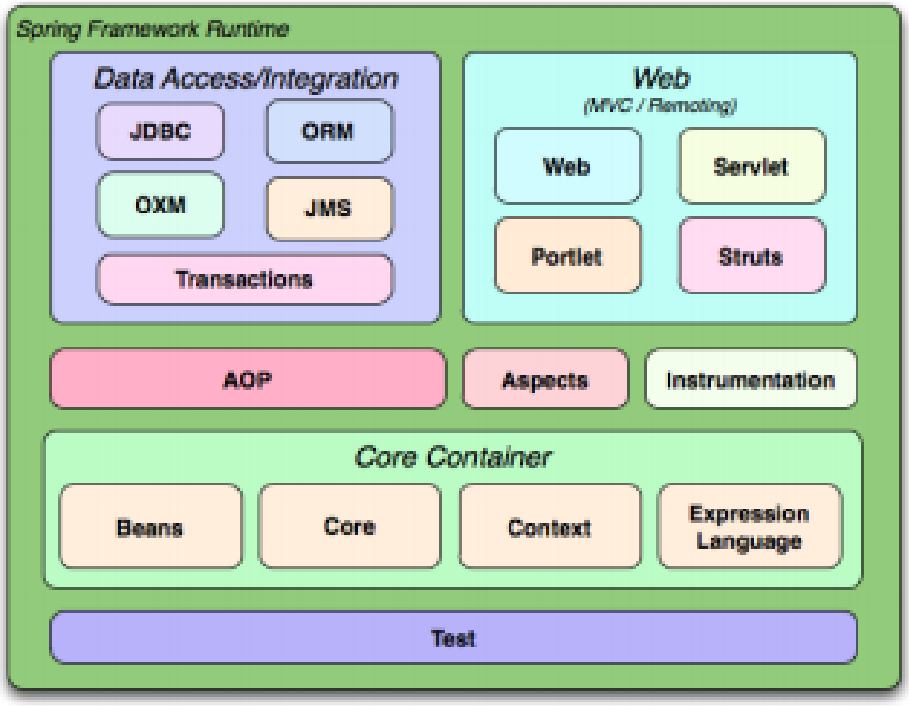
***Singleton**: Es el ámbito por defecto de Spring, es decir, si no se especifica el tipo en la creación del bean, Spring lo creará con este ámbito. El contenedor de Spring creará una única instancia compartida de la clase designada por este bean, por lo que siempre que se solicite este bean se estará inyectando el mismo objeto.

***Prototype**: El contenedor de Spring creará una nueva instancia del objeto descrito por el bean cada vez que se le solicite el mismo. En algunos casos puede ser necesario, pero hay que tener en cuenta que no se debe abusar de este tipo puesto que puede causar una pérdida de rendimiento en la aplicación.

***Request**: El contenedor de Spring creará una nueva instancia del objeto definido por el bean cada vez que reciba un HTTP request.

***Session**: El contenedor de Spring creará una nueva instancia del objeto definido por el bean para cada una de las sesiones HTTP y entregará esa misma instancia cada vez que reciba una petición dentro de la misma sesión.

- MODULOS



El módulo de core, que permite crear e inyectar beans de cara al diseño de inversión del control (IoC) que proporciona Spring. Este diseño consiste en especificar respuestas y acciones deseadas ante sucesos para que algún tipo de entidad o arquitectura externa lo ejecute.

- CREACION DE CONTEXTO Y BEANS

inversión de control (IoC), este contenedor es el encargado de administrar los beans, un bean es un objeto Java que es administrado por Spring.

la tarea de instanciar, inicializar y destruir objetos será delegada a este contenedor, el mismo también realiza otras tareas como la inyección de dependencias (DI)

Es posible formar su cuerpo, añadiéndole beans a través de etiquetas y pudiendo éstos ser usados en cualquier parte de la aplicación. El XML se mostraría de la siguiente manera:

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

<beans:bean id="servicioA" class="es.prueba.ServicioA"></beans:bean>

<beans:bean id="servicioB" class="es.prueba.ServicioB"></beans:bean>

</beans:beans>
```

al finalizar se debe cargar todas las anotaciones creadas y se puede utilizar las etiquetas tales como:

CONTEXT:ANNOTATION-CONFIG: con la que se le informa al framework que se van a usar anotaciones en el código.

CONTEXT:COMPONENT-SACN: se le indica al framework la ruta de paquetes que debe escanear en busca de clases para crear sus beans

- SPRING ESTEREOTIPOS Y ANOTACIONES

Spring ha definido multitud de etiquetas y anotaciones con el fin de categorizar componentes dentro del código y asignarles un cometido.

Estereotipos

@Component: es el estereotipo principal, indica que la clase anotada es un componente o bean de Spring.

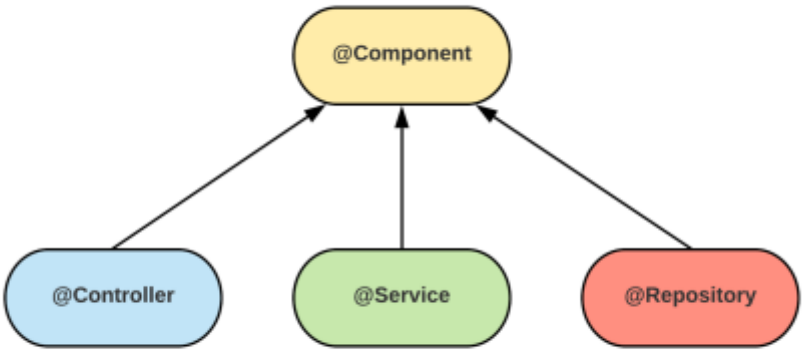


Ilustración 11 – Estereotipos Spring Framework

@Repository: Es el estereotipo que tiene como función dar de alta un bean para que implemente el patrón repositorio, que es el encargado de almacenar datos en una base de datos o repositorio de información que se necesite.

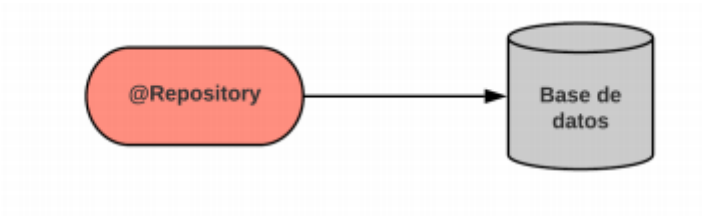


Ilustración 12 – Estereotipo @Repository en Spring Framework

@Service: Este estereotipo se encarga de gestionar las operaciones de negocio más importantes a nivel de la aplicación y aglutina llamadas a varios repositorios de forma simultánea. Su tarea fundamental es la de agregador.

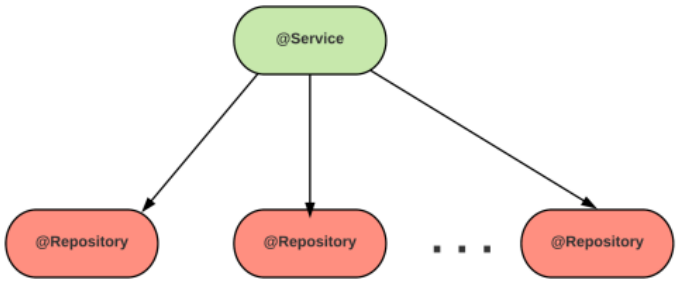


Ilustración 13 - Estereotipo @Service en Spring Framework

@Controller: El último de los estereotipos, es el 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.

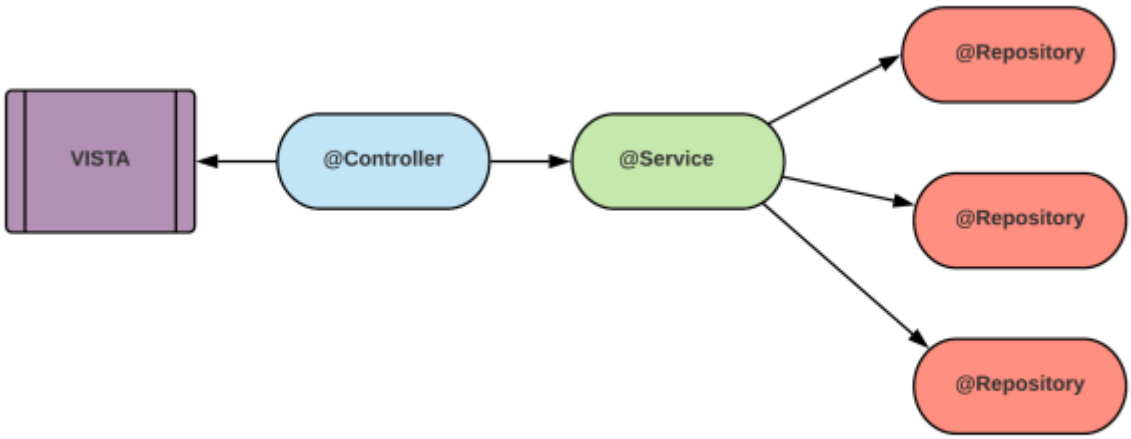


Ilustración 14 - Estereotipo @Controller en Spring Framework

4. JPA (JAVA PERSISTENCE API)

JAVA es orientado a objetos, sin embargo, las bases de datos relacionales almacenan la información en forma de tabla con sus respectivas filas y columnas. Para realizar una correlación entre ambos se necesita de una interfaz que permita guardar la información que se maneja dentro de la aplicación JAVA en el modelo relacional de base de datos.

JPA es una interfaz común y generalmente ésta es implementada por frameworks de persistencia

- ANOTACIONES PRINCIPALES

Cuando se habla de una entidad, se hace referencia en cualquier caso a un objeto POJO (Plain Old Java Object), es decir un objeto sencillo que no extiende de otro ni implementa funcionalidad fuera del mismo.

Una entidad se caracteriza por ser una clase de primer nivel, no ser final, proporcionar un constructor e implementar la interfaz java.io.Serializable.

Cualquier entidad JPA presenta numerosas etiquetas que la caracterizan, las necesarias para formar una entidad sencilla serían:

@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 puede definir como un tipo de lectura temprana, es decir, en la consulta del objeto se obtienen todos los valores de las entidades que están relacionadas con la entidad.

*** LAZY**: se define como lectura demorada, permite obtener un objeto de base de datos sin los valores de la relación a la que hace referencia el atributo.

5. HIBERNATE

Hibernate es un framework de persistencia de software libre que implementa la gestión de la API de persistencia de Java. En lo referente a la flexibilidad y escalabilidad, hibernate está diseñado para adaptarse a cualquier esquema y modelo de datos, además ofrece la relación inversa

El First Level Caché es el que presenta automáticamente Hibernate cuando dentro de una transacción se interactúa con la base de datos .Se puede considerar como una caché de corta duración ya que es válido solamente entre el begin y el commit de una transacción, de forma aislada a las demás.

El Second Level Cache permite ser configurado de cara a la mejora del rendimiento. La diferencia fundamental es que éste tipo de caché es válida para todas las transacciones y puede persistir en memoria durante todo el tiempo en que el aplicativo esté online, se puede considerar como una caché global.

Hibernate genera las consultas SQL y libera al desarrollador del manejo más primario de las mimas, además posee un lenguaje de consulta propio denominado HQL (Hibernate Query Language) y de una API para construir consultas conocida como Criteria.

- ### CRITERIA

Criteria es una API creada por hibernate pensada específicamente para facilitar las consultas a base de datos. La principal ventaja que tiene es que la forma de construir las querys de base de datos es absolutamente orientada a objetos.

se crea una instancia de Criteria para un objeto en concreto y haciendo uso de métodos de esta instancia se da forma a las restricciones de búsqueda en torno a ese objeto

. Criteria crit = sess.createCriteria(Cat.class);

*Para limitar el número de resultados:

crit.setMaxResults(50);

*Añadir criterios individuales:

crit.add(Restrictions.like("name", "Fritz%"))

crit.add(Restrictions.between("weight", minWeight, maxWeight))

*Para disyunciones:

crit.add(Restrictions.or(

Restrictions.eq("age", new Integer(0)),

Restrictions.isNull("age")))

*Ordenar resultados:

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

6. JWT (JSON WEB TOKEN)

La autenticación basada en token es una referencia en el desarrollo de aplicaciones web, ya que presenta algunas ventajas respecto a la autenticación más común, en la que se guardan en sesión los datos del usuario

En la autenticación con token, el usuario se identifica bien con un usuario/contraseña o mediante una única clave y la aplicación web le devuelve una especie de firma cifrada que el usuario usará en las cabeceras de cada una de las peticiones HTTP.

Header

Es la primera parte del token, está formada por el tipo de token y el algoritmo de codificación utilizado:

```
{
"typ": "JWT",
 "alg": "HS512"
}
```

- 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:


```
{
  "sub": "",
  "exp": 1535967018,
  "iat": 1535880618
}
```