UNIVERSIDAD PRIVADA FRANZ TAMAYO

CARRERA INGENIERIA DE SISTEMAS

SERVICIO WEB API REST SOBRE EL FRAMEWORK SPRING, HIBERNATE, JSON WEB TOKEN

Alumno: Adriana Contreras Arancibia

Materia: Programación III

Carrera: Ing. De Sistemas

Paralelo: Progra(1)

Docente: Lic. William R. Barra Paredes

Fecha: 27/04/2020

Cochabamba-Bolivia

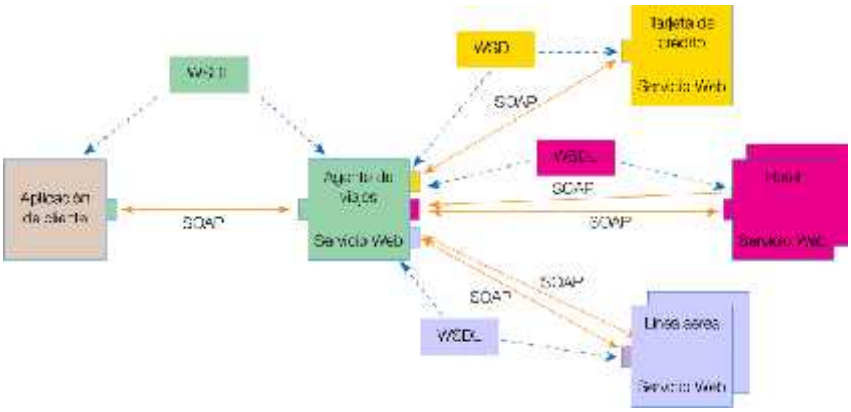
A. SERVICIOS WEB

Un web service es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, soportar la interacción máquina-a-máquina. Permiten la interoperabilidad entre plataformas de distintos fabricantes y desarrolladas en lenguajes de programación diferentes por medio de protocolos estándar y abiertos.

se podría decir que un web service es una función que diferentes servicios o equipos utilizan; es decir, solo se envían parámetros al servidor (lugar donde está alojado el web service) y éste responderá la petición.

La principal ventaja de utilizar un servicio web es que son bastante prácticos debido a que son independientes de las aplicaciones

- Ejemplo grafico



A.1 TIPOS DE SERVICIOS WEB

- Web Services Protocol Stack: Así se denomina al conjunto de servicios y protocolos de los servicios Web.
- XML (Extensible Markup Language): Es el formato estándar para los datos que se vayan a intercambiar.
- SOAP (Simple Object Access Protocol) o XML-RPC (XML Remote Procedure Call): Protocolos sobre los que se establece el intercambio.
- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como HTTP (Hypertext Transfer Protocol), * FTP (File Transfer Protocol), o SMTP (Simple Mail Transfer Protocol).
- WSDL (Web Services Description Language): Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web.
- UDDI (Universal Description, Discovery and Integration): Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles.
- WS-Security (Web Service Security): Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados.
- REST (Representational State Transfer): es un conjunto de principios de arquitectura para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato (XML, JSON, etc.) y sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes, como por ejemplo SOAP.

A.2 CARACTERISTICAS DE SOAP

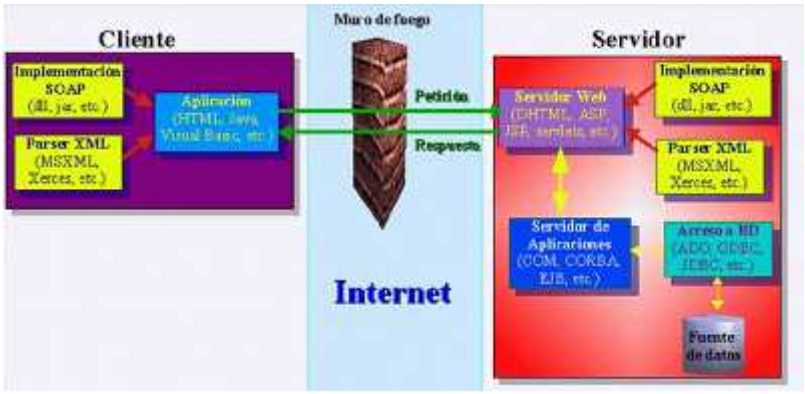
SOAP (Simple Object Access Protocol, Protocolo Simple de Acceso a Objetos) es un protocolo de mensajes entre computadores. SOAP especifica el formato de mensaje que accede e invoca a los objetos, más que un objeto en particular. Este protocolo está basado en XML y formado por tres partes:

- 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

Estructuración

- WSDL (Web Services Description Language) lenguaje de la interfaz pública para los servicios web, basada en XML
- UDDI (Universal Description, Discovery and Integration) protocolo que publica la información de los servicios web, comprueba qué servicios están disponibles.

- Ejemplo grafico



- Otro ejemplo

  Microsoft: Recientemente ha anunciado la disponibilidad de su primer WS, llamado MapPoint.Net. mediante este servicio, el usuario podrá conocer su localización exacta y otros datos adicionales relacionados con su posición actual, como información de tráfico, rutas posibles o puntos comerciales cercanos.

A.3 CARACTERISTICAS DE REST

REST (Representational State Transfer) es un estilo de arquitectura de software para sistemas hipermedias distribuidos tales como la Web, no es más que una colección de recursos definidos y diseccionados es utilizada para describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional como hace SOAP.

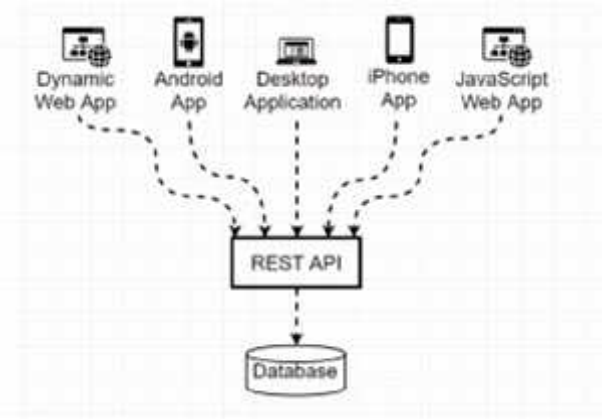
REST es un estilo de arquitectura basado en estándares como son HTTP, URL, una de las características principales de los servicios Web REST es el uso explícito de métodos HTTP (HyperText Transfer Protocol). Estos métodos son indicados en la cabecera HTTP por parte del cliente y son los siguientes:

- GET: recoge información de un recurso
- PUT: modifica o actualiza el estado de un recurso
- POST: crea un nuevo recurso en el servidor
- DELETE: elimina un recurso del servidor.

Algunas características son:

- Escalabilidad.
- Independencia.
- Identificación de recursos.
- Operaciones definidas
- Compatibilidad.
- Protocolo cliente/servidor sin estado.

Ejemplo grafico



A.4 COMPARATIVAS ENTRE REST Y SOAP

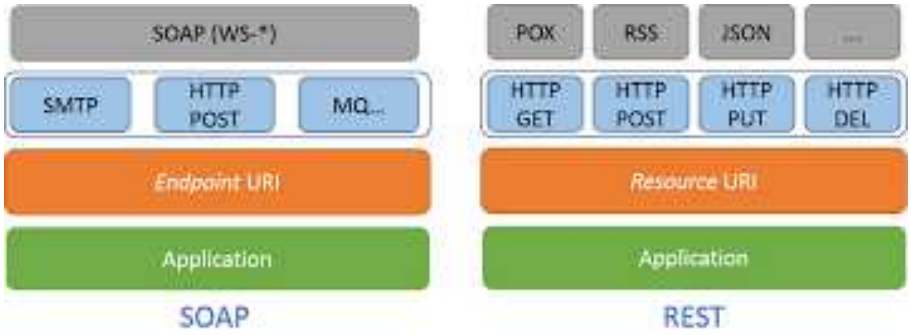
Se puede decir que SOAP es fuertemente acoplado, puede ser testado y depurado antes de poner en marcha la aplicación. Mientras que de REST tiene más escalabilidad en todo tipo de sistemas y escaso consumo de recursos debido al limitado número de operaciones y al esquema de direccionamiento unificado.

Ventajas

- SOAP
  - Pocas operaciones con muchos recursos
  - Se centra en la escalabilidad y rendimiento a gran escala para sistemas distribuidos hipermedia
  - XML auto descriptivo
  - Comunicación punto a punto y segura
  - HTTPS
  - Sincrono

- REST
  - Muchas operaciones con pocos recursos
  - Se centra en el diseño de aplicaciones distribuidas
  - Tipado fuerte, XML Schema
  - Sincrono y Asincrono
  - WS SECURITY
  - Comunicación origen a destino seguro

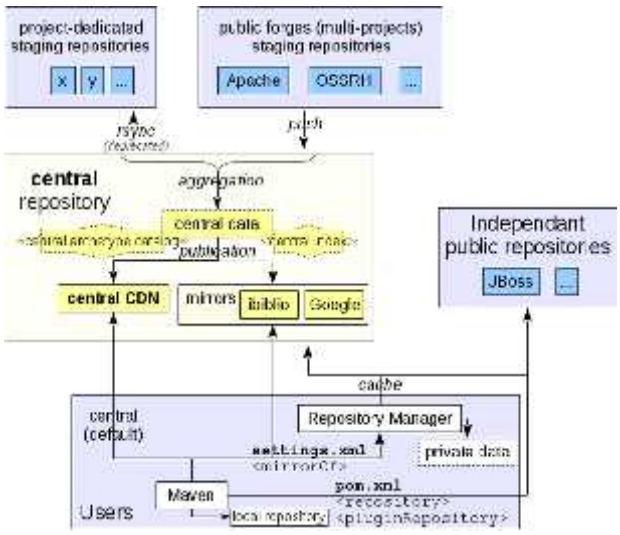
Ejemplo grafico



B. MAVEN

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

Ejemplo grafico

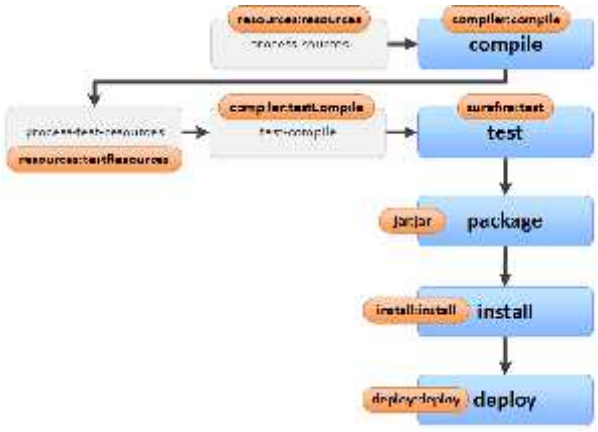


B.1 CICLOS DE VIDA

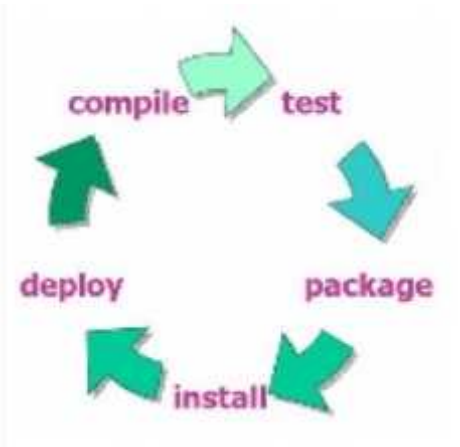
- **Validación** (validate): Validar que el proyecto es correcto.
- **Compilación** (compile).
- **Test** (test): Probar el código fuente usando un framework de pruebas unitarias.
- **Empaquetar** (package): Empaquetar el código compilado y transformarlo en algún formato tipo .jar o .war.
- **Pruebas 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).
- **Despliega**r el código a un entorno (deploy).

Ejemplo grafico



Ejemplo grafico 2

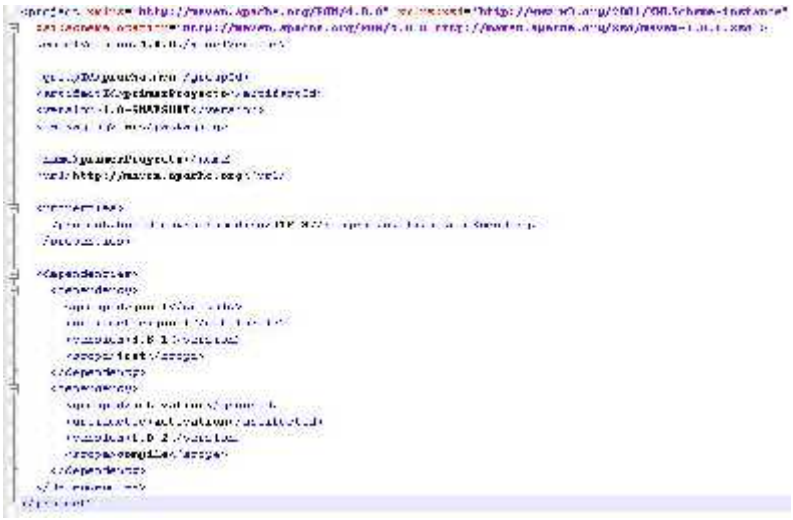


B.1 POM

POM (Project Object Model)es un modelo de objeto para un proyecto, o como describe la documentación de Maven, es un fichero xml está compuesto por un conjunto de etiquetas que dan forma al proyecto. La primera de ellas es la etiqueta, que engloba todo el POM y en la que se define la arquitectura del xml, Contiene información acerca del proyecto, fuentes, test, dependencias, plugins, versión.

Ejemplo de POM

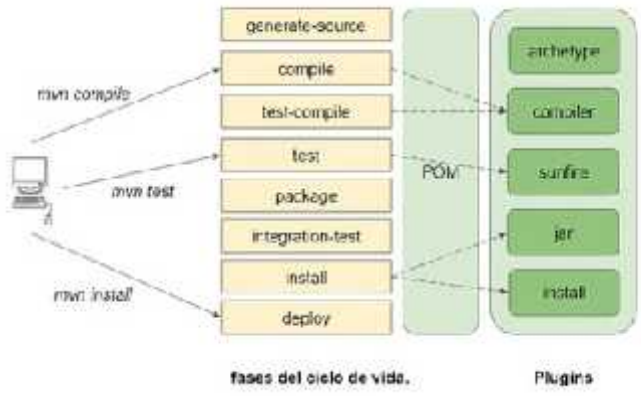
Ejemplo grafico 2



B.1 REPOSITORIO DE MAVEN

Permite agregar librerías propietarias a nuestros proyectos, estas librerías pueden no estar en los repositorios públicos de maven y tendremos que descargarlas y las guarda todas en un mismo repositorio, generalmente <USER_HOME>/.m2/repository, un ejemplo es el driver JDBC de SQLServer.

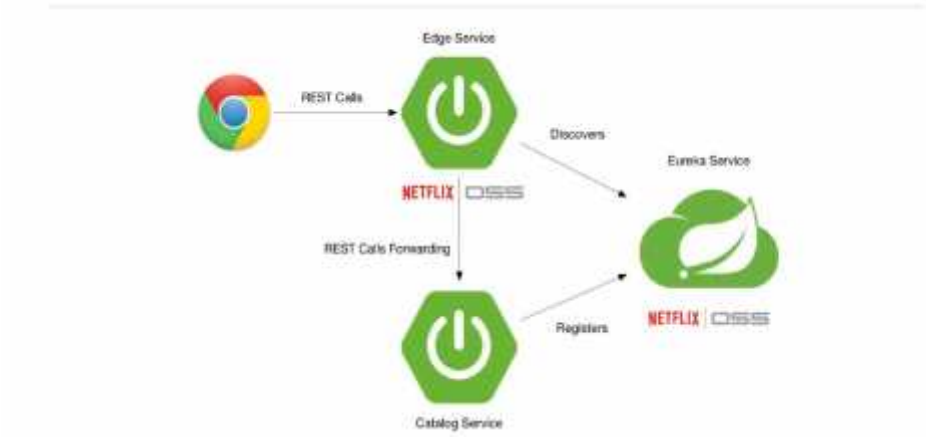
Ejemplo grafico



C. SPRING FRAMEWORK

Un Framework son un conjunto de clases y soluciones ya implementadas para su uso con el fin de estandarizar, agilizar y resolver los problemas. El Framework Spring nos permite desarrollar aplicaciones de manera más eficaz y rápida, ahorrando a su vez muchas líneas de código. En un modelo de aplicación web habitual, sin hacer uso de algún framework de gestión de las acciones de la web, 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.

Ejemplo grafico



C.1 MODULOS

Spring se apoya en diferentes módulos con sus respectivas funcionalidades

Algunos de los módulos más importantes de Spring:

- Core: La parte fundamental de este framework es el módulo Core, y los adyacentes Bean y Context. Proveen toda la funcionalidad para la inyección de dependencias, permitiéndole administrar la funcionalidad del contenedor de Beans. Además, también proveen de los servicios Enteprise como JDNI, EJB, …

- AOP: Se trata de un módulo que nos permitirá utilizar el paradigma de Programación Orientada a Aspectos (Aspect Oriented Programming). Este paradigma permite mejorar la modularización y separar las responsabilidades. De esta forma, podemos separar las funcionalidades comunes, que se utilizan transversalmente a lo largo de toda la aplicación, de aquellas que son propias de cada módulo.

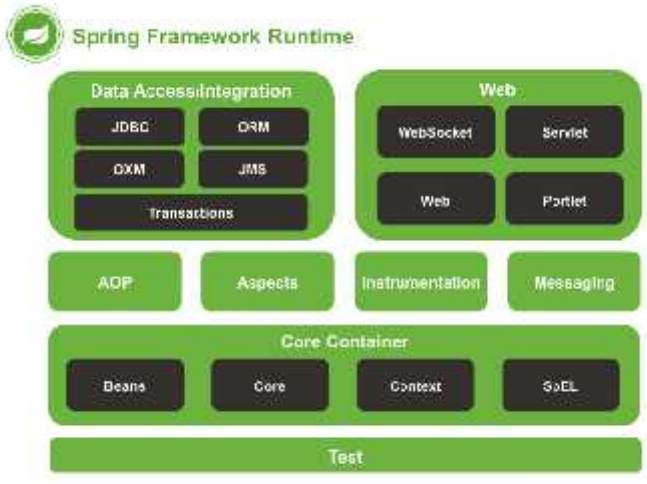
- Data: Se trata de un gran módulo, formado por múltiples submódulos, y que nos permite simplificar el acceso y persistencia de datos. Spring Data nos proporciona soporte para usar base de datos relacionales (JDBC), ORMs (como por ejemplo JPA, Hibernate, …) e incluso modelos de persistencia NoSQL (como por ejemplo, MongoDB).

- Web: Este módulo nos permitirá implementar el patrón Modelo-Vista-Controlador (MVC) de una manera sencilla y limpia, haciendo uso de forma transparente también de otros patrones de diseño, como FrontController. De esta forma, podemos separar limpiamente la lógica de negocio de la presentación de los datos y el acceso a los mismos. Además de aplicaciones que implican el uso de vistas y formularios, también podremos crear servicios web (por ejemplo, al estilo REST) de una forma sencilla y rápida.

El modularidad de Spring nos permitirá la posibilidad de usar solo algunas de las partes del framework, y poder combinar esta con otros frameworks diferentes:

- Un proyecto que implemente MVC mediante el uso de Struts podría incorporar el contenedor de IoC mediante el uso de Spring (sin necesidad de utilizar Spring Web MVC).
- Una aplicación web desarrollada con Spring Web MVC podría implementar su capa de datos mediante el uso de Hibernate (sin hacer uso de Spring Data).

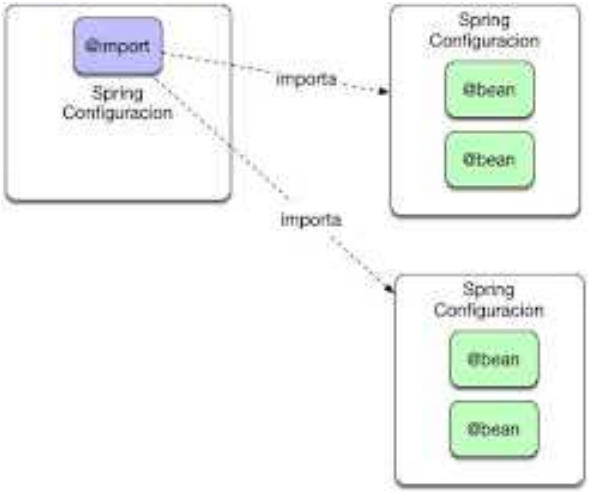
Ejemplo grafico



C.2 CREACIÓN DE CONTEXTO Y BEANS

Bean es un objeto Java que es administrado por Spring, es decir, 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), para poder usar este contenedor se tiene que configurar mediante un archivo XML de configuración de Spring.

Ejemplo grafico



Ejemplo2

```
        package com.arquitecturajava.spring;
1
2       import org.springframework.context.annotation.Bean;
3       import org.springframework.context.annotation.ComponentScan;
4       import org.springframework.context.annotation.Configuration;
5
6       @Configuration
7       public class SpringConfiguracionSeguridad {
8
9
10          @Bean
11          public static HelperSeguridad helper2() {
12
13              return new HelperSeguridad();
14          }
15
16      }
17
```

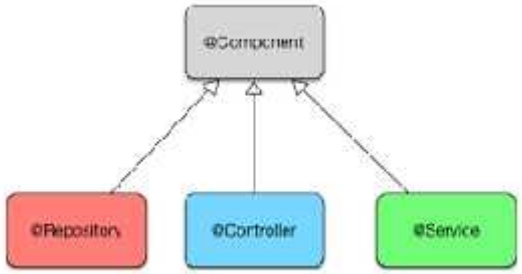
```
1       package com.arquitecturajava.spring;
2
3       import org.springframework.context.annotation.Bean;
4       import org.springframework.context.annotation.Configuration;
5
6       @Configuration
7
8       public class SpringConfiguracionPersistencia {
9
10          @Bean
11          public static HelperPersistencia helper1() {
12
13              return new HelperPersistencia();
14          }
15      }
```

C.3 SPRING ESTEREOTIPOS Y ANOTACIONES

Spring define un conjunto de anotaciones core que categorizan cada uno de los componentes asociándoles una responsabilidad concreta.

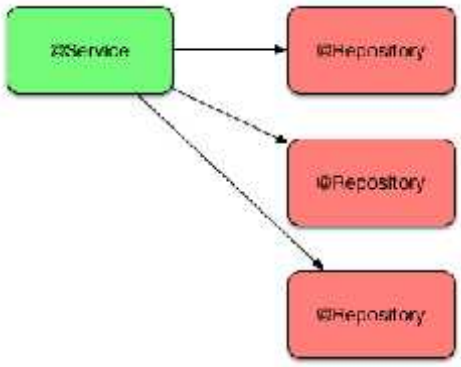
@Component: Es el estereotipo general y que permite anotar un bean para que spring lo considere uno de sus objetos.

Ejemplo



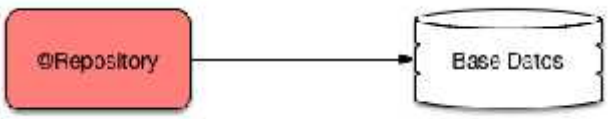
@Service: Este estereotipo se encarga de gestionar las operaciones de negocio más importantes a nivel de la aplicación y aglutina llamadas a varios repositorios de forma simultánea. su tarea fundamental es la de agregador.

Ejemplo



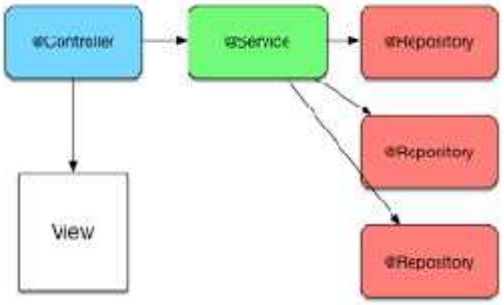
@Repository: Es el estereotipo que se encarga de dar de alta un bean para que implemente el patrón repositorio que es el encargado de almacenar datos en una base de datos o repositorio de información que se necesite. al marcar el bean con esta anotación Spring aporta servicios transversales como conversión de tipos de excepciones.

Ejemplo



@Controller: El último de los estereotipos que es el que realiza las tareas de controlador y gestión de la comunicación entre el usuario y el aplicativo.

Ejemplo



Otro ejemplo

Annotation	Use	Description
@Autowired	Constructor, Field, Method	Declares a constructor, field, setter method, or configuration method to be autowired by type. Items annotated with @Autowired do not have to be public.
@Configurable	Type	Used with <context:spring-configured> to declare types whose properties should be injected, even if they are not instantiated by Spring. Typically used to inject the properties of domain objects.
@Order	Type, Method, Field	Defines ordering, as an alternative to implementing the org.springframework.core.Ordered interface.
@Qualifier	Field, Parameter, Type, Annotation Type	Guides autowiring to be performed by means other than by type.
@Required	Method (setters)	Specifies that a particular property must be injected or else the configuration will fail.
@Scope	Type	Specifies the scope of a bean, either singleton, prototype, request, session, or some custom scope.

ANOTACIONES

- @Required

  Esta anotación se aplica a los métodos de “setters” de beans. Si se considera un escenario en él sé que necesita hacer cumplir una propiedad requerida. La anotación @Required indica que el bean afectado debe llenarse en el momento de la configuración con la propiedad requerida. De lo contrario, se genera una excepción de tipo BeanInitializationException

- @Autowired

  Esta anotación se aplica a campos, métodos de “setters” y constructores. La anotación @Autowired inyecta la dependencia del objeto implícitamente.Cuando se usa @Autowired en los campos y se pasen los valores de los campos con el nombre de la propiedad, Spring asignará automáticamente los campos con los valores que se pasan.

Ejemplo

```
public class Customer
{

    private Person person;
    @Autowired

    public void setPerson (Person person) {
       this.person=person;
    }
}
```

- @Qualifier

Esta anotación se usa junto con la anotación @Autowired. Cuando necesita más control del proceso de inyección de dependencia, se puede utilizar @Qualifier. @Qualifier se puede especificar en argumentos de constructor individuales o parámetros de método. Esta anotación se utiliza para evitar la confusión que ocurre cuando crea más de un bean del mismo tipo y se desea conectar solo uno de ellos con una propiedad.

```
@Component
public class BeanA {
    @Autowired
    @Qualifier("beanB2")
    private IBean dependency;
    ...
}
```

- @Configuration

Esta anotación se usa en clases que definen beans. @Configuration es un análogo para un archivo de configuración XML: se configura mediante clases Java. Una clase Java anotada con @Configuration es una configuración en sí misma y tendrá métodos para crear instancias y configurar las dependencias.

```
@Configuartion
public class DataConfig {
    @Bean
    public DataSource source() {
        DataSource source = new OracleDataSource();
        source.setURL();
        source.setUser();
        return source;
    }
    @Bean
    public PlatformTransactionManager manager() {
        PlatformTransactionManager manager = new
BasicDataSourceTransactionManager();
        manager.setDataSource(source());
        return manager;
    }
}
```

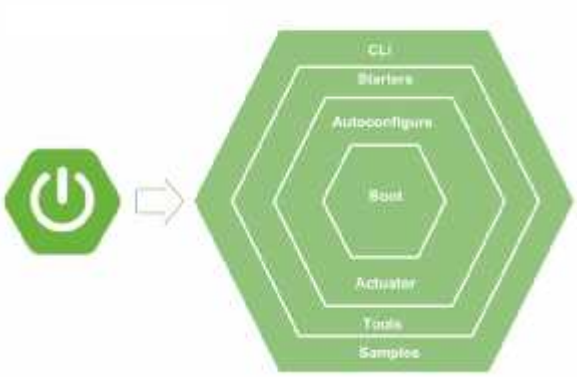
- @Bean

  Esta anotación se utiliza a nivel de método. La anotación @Bean funciona con @Configuration para crear beans Spring. Como se mencionó anteriormente, @Configuration tendrá métodos para crear instancias y configurar dependencias. Dichos métodos serán anotados con @Bean. El método anotado con esta anotación funciona como la ID del bean, y crea y devuelve el bean real.

```
@Configuration
public class AppConfig {
    @Bean
    public Person person() {
        return new Person(address());
    }
    @Bean
    public Address address() {
        return new Address();
    }
}
```

C.4 SPRING BOOT

Spring Boot es una infraestructura ligera que elimina la mayor parte del trabajo de configurar las aplicaciones basadas en Spring, tiene la intención de simplificar los pasos 1 y 3

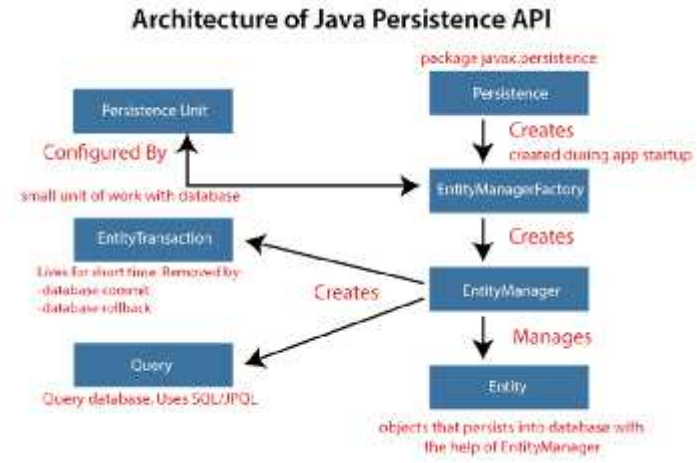


D. JPA (JAVA PERSISTENCE API)

JPA es la propuesta estándar que ofrece Java para implementar un Framework Object Relational Mapping (ORM), que permite interactuar con la base de datos por medio de objetos, de esta forma, JPA es el encargado de convertir los objetos Java en instrucciones para el Manejador de Base de Datos (MDB).

JPA es una interfaz común y generalmente ésta es implementada por frameworks de persistencia, en este proyecto se ha elegido Hibernate. De esta manera será Hibernate el que realice el trabajo, pero siempre funcionando bajo la API de JPA.

Ejemplo



D.1 ANOTACIONES PRINCIPALES

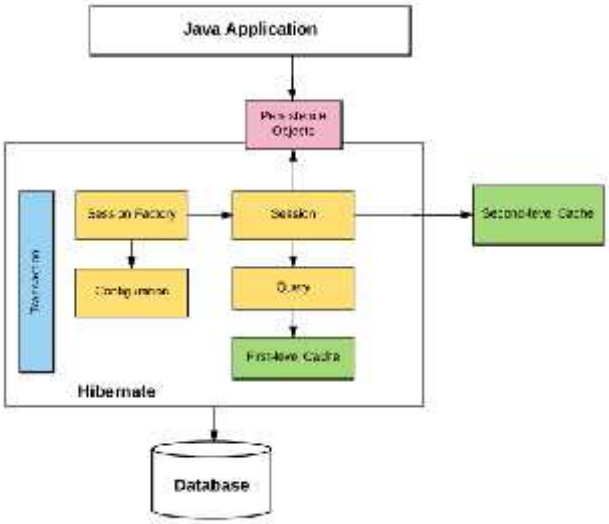
- @Entity: informa al proveedor de persistencia que cada instancia de esta clase es una entidad.
- @JoinColumn: permite configurar aquel atributo que contiene una clave foránea a otra tabla.
- @Colum: acompañará a cada uno de los atributos de la entidad y permitirá configurar el nombre de la columna 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.
- @Table: permite configurar el nombre de la tabla que se está mapeando con dicha entidad. Para ello se hace uso del atributo name.
- @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.
- @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.

E. HIBERNATE

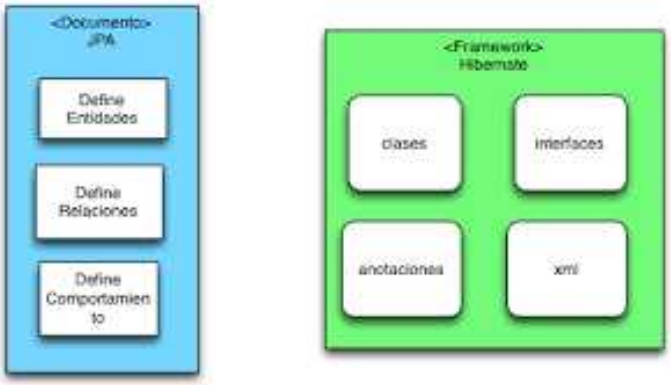
Hibernate es una herramienta de Mapeo objeto-relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos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

En otras palabras, Hibernate es un Framework que agiliza la relación entre la aplicación y la base de datos. Para poder aprender a utilizarlo es necesario contar con los conocimientos básicos de base de datos y SQL así como manejar el lenguaje Java.

Ejemplo



Cuadro comparativo de JPA y HIBERNADE

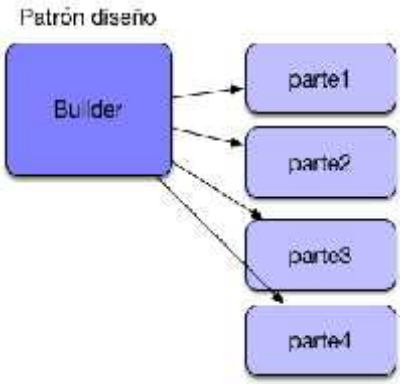


E.1 CRITERIA

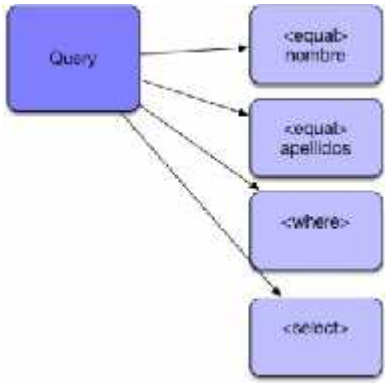
Criteria es una API creada por hibernate pensada específicamente para facilitar las consultas a base de datos. La principal ventaja que tiene es que la forma de construir las querys de base de datos es absolutamente orientada a objetos. Esto es de gran utilidad de cara a dar soporte a un formulario de búsqueda que presente multitud de campos, ya que nos evitaría líneas de comprobación para cada uno de los campos, simplificando mucho el código de las consultas.

Como funciona exactamente el API de Criteria

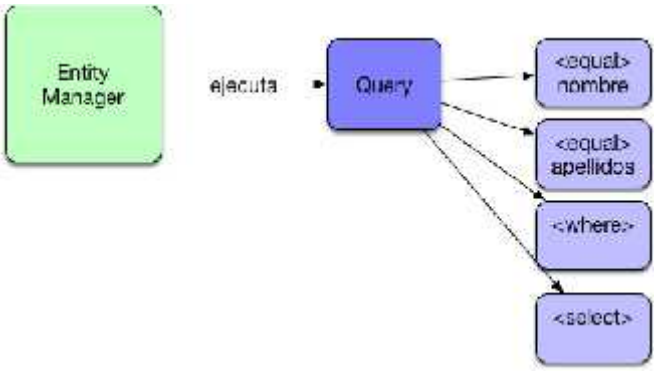
Este API se encarga de diseñar la consulta a través de un Builder, un patrón de diseño que construye un objeto paso a paso



A través de estos pasos se construye el objeto por complejo verificando que todo es correcto. Este es el enfoque JPA Criteria API



Para después ejecutar esa consulta construida utilizando el EntityManager:



E.2 JWT (JSON WEB TOKEN)

JWT (JSON Web Token) es un estándar qué está dentro del documento RFC 7519.

En el mismo se define un mecanismo para poder propagar entre dos partes, y de forma segura, la identidad de un determinado usuario, además con una serie de claims o privilegios.

Estos privilegios están codificados en objetos de tipo JSON, que se incrustan dentro de del payload o cuerpo de un mensaje que va firmado digitalmente.

Un token tiene tres partes:

- Header : encabezado dónde se indica, al menos, el algoritmo y el tipo de token, que en el caso del ejemplo anterior era el algoritmo HS256 y un token JWT.
- Payload : donde aparecen los datos de usuario y privilegios, así como toda la información que queramos añadir, todos los datos que creamos convenientes.
- Signature : una firma que nos permite verificar si el token es válido, y aquí es donde radica el quid de la cuestión, ya que si estamos tratando de hacer una comunicación segura entre partes y hemos visto que podemos coger cualquier token y ver su contenido con una herramienta sencilla, ¿dónde reside entonces la potencia de todo esto?

Ejemplo del ciclo de vida de un token JWT