# Spring 5.0 REST

## Acceso a Google Drive

# Google API

Application programming interfaces

https://developers.google.com/products/

OAuth 2.0 protocol (https://tools.ietf.org/html/rfc6749)

# Flujo de protocolo abstracto

# Autorización

- Código de autorización
- Implícito: utilizado con aplicaciones móviles o aplicaciones web
- Credenciales de contraseña del propietario del recurso: utilizado con aplicaciones confiables, como aquellas pertenecientes al servicio
- Credenciales del cliente: usadas con el acceso API de aplicaciones

# Google API

```
getFiles(auth:string): Observable<ListaArchivos> {
        let url:string =  "https://www.googleapis.com/drive/v3/files";

        return this.httpClient.get<ListaArchivos>(url, {
             headers: new HttpHeaders().set('Content-Type', 'application/json').set('Authorization',
`Bearer ${auth}`)})
             .do(response => console.log(JSON.stringify(response)))
             .catch(this.handleError);
    }
```

# Google Drive API (Rest)

https://developers.google.com/drive/api/v3/reference/

Servicios:

```
GET https://www.googleapis.com/drive/v3/files/fileId
```

```
GET https://www.googleapis.com/drive/v3/files
```

https://developers.google.com/drive/api/v3/reference/

Header: `Authorization: Bearer oauth2-token`

# Google Drive API (Rest)

```
export interface ListaArchivos {
   files:Archivo[];
   incompleteSearch:boolean;
   kind:string;
   nextPageToken:string;
}

export interface Archivo {
   name:string,
   id:string,
   kind:string,
   mimeType:string
};
```

# Google Drive Java Client

https://developers.google.com/api-client-library/java/

```
<dependency>
    <groupId>com.google.api-client</groupId>
    <artifactId>google-api-client</artifactId>
    <version>1.23.0</version>
</dependency>
```

# Google Drive API

| MIME Type | Description |
| --- | --- |
| application/vnd.google-apps.audio | |
| application/vnd.google-apps.document | Google Docs |
| application/vnd.google-apps.drawing | Google Drawing |
| application/vnd.google-apps.file | Google Drive file |
| application/vnd.google-apps.folder | Google Drive folder |
| application/vnd.google-apps.form | Google Forms |
| application/vnd.google-apps.fusiontable | Google Fusion Tables |
| application/vnd.google-apps.map | Google My Maps |
| application/vnd.google-apps.photo | |
| application/vnd.google-apps.presentation | Google Slides |
| application/vnd.google-apps.script | Google Apps Scripts |
| application/vnd.google-apps.site | Google Sites |
| application/vnd.google-apps.spreadsheet | Google Sheets |
| application/vnd.google-apps.unknown | |
| application/vnd.google-apps.video | |
| application/vnd.google-apps.drive-sdk | 3rd party shortcut |

# Spring

## *Conceptos Básicos*

- Framework de código abierto para el desarrollo de aplicaciones Java.
- Su aspecto modular lo hace flexible y configurable para cualquier tipo de aplicación.
- Soporte excelente para aplicaciones REST

# Spring

## *Conceptos Básicos*

- Motivación: Facilitar el desarrollo de aplicaciones J2EE, promoviendo buenas prácticas de diseño y programación.
- Framework de Frameworks: Spring integra diferentes frameworks gestionando los ciclos de vida de diversos objetos. Spring provee soporte a diversos frameworks: Struts, Hibernate, JSF, EJB, etc.

# Spring

**Versiones:**

- Primera versión: publicada en 2002 por Rod Johnson.
- En 2003 se publica una versión bajo la licencia Apache 2.0 (open source).
- En 2004 se lanza la versión 1.0 la que fue más popularmente conocida.
- 2006, Spring 2.0.
- 2009 Spring 3.0.
- 2013, Spring 4.0.
- Ultima version: 5.1.0.

# Spring

## *Instalación y requisitos*

- Java: Spring 4 soporta desde Java SE 6 a Java 8
- Gestión y construcción: Maven o Gradle
- IDE: Spring Tool Suite (basado en Eclipse) es el IDE oficial aunque se pueden desarrollar aplicaciones en otros IDEs como IntelliJ IDEA con ayuda de plugins.
- Servidor: Versiones de Tomcat de 5 a 8
- BBDD: MySQL

# Spring

Los módulos de Spring que se desean agregar en el proyecto se añadirán al pom.xml para que Maven los integre. Ejemplo:

```
<dependency>
	<groupId>org.springframework</groupId>
	<artifactId>spring-context</artifactId>
	<version>4.3.9.RELEASE</version>
</dependency>
```

# Spring

### *Ventajas*

- Muy flexible: implementación mediante interfaces. Bajo acoplamiento a partir de la inyección de dependencias.
- Spring framework dispone de templates para JDBC, Hibernate, JPA etc.
- Ofrece un framework para todas las capas de la aplicación.
- Fácil para testear.

# Spring - Inyección de Dependencias

### *Inversión de control*

- Una de los conceptos fundamentales de Spring, la DI resulta ser una aplicación del concepto de Inversión de Control.
- En una aplicación tradicional, los objetos hacen llamadas a librerías reusables para hacer ciertos procesos. En un paradigma de IoC se consigue que este código reusable sea el que influya en el código que va a realizar el proceso.

# Spring - Inyección de Dependencias

***Bean***

Un bean es un objeto gestionado por el contenedor IoC, es decir, es el propio contenedor de IoC quien lo instancia y controla su ciclo de vida. Los beans se pueden configurar mediante ficheros XML donde se especifica la metadata de configuración;

1.1) Cómo se crea.

1.2) Detalle del ciclo de vida.

1.3) Dependencias.

# Spring - Inyección de Dependencias

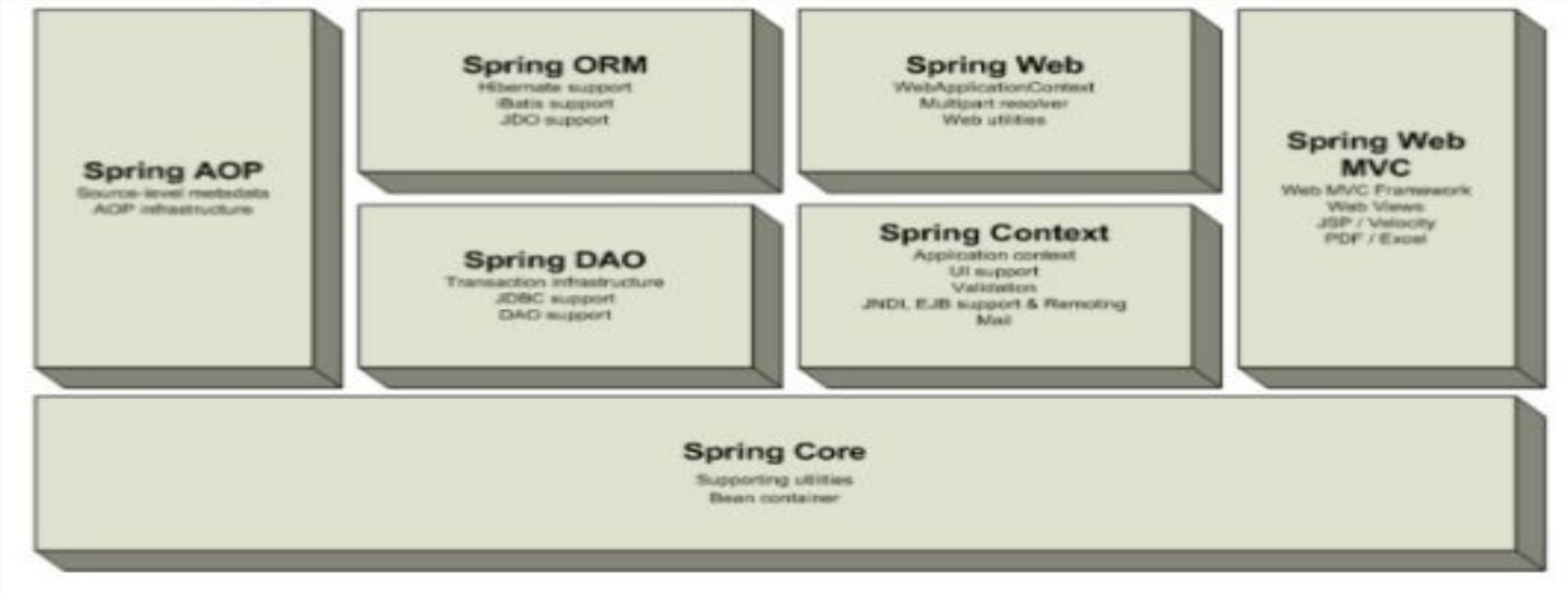
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

  <bean id="..." class="...">
      <!-- configuración del bean -->
  </bean>
  <!-- Bean con método de inicialización -->
  <bean id="..." class="..." init-method="...">
      <!-- configuración del bean -->
  </bean>
  <!-- Bean con destructor -->
  <bean id="..." class="..." destroy-method="...">
      <!-- configuración del bean -->
  </bean>

</beans>
```

# Spring

Principales módulos

- Spring Core (Inversión del control (IoC) / Inyección de dependencias (DI))
- Spring AOP (Programación orientada a Aspectos)
- Spring JDBC (Acceso a Datos)
- Spring MVC (Desarrollo Web según el patrón MVC)
- Spring Remoting (Distribución)
- Spring Transaction Abstraction (Transacciones)

# Spring


Spring AOP
Source-level metadata
AOP infrastructure
Spring ORM
Hibernate support
iBatis support
JDO support
Spring DAO
Transaction infrastructure
JDBC support
DAO support
Spring Web
WebApplicationContext
Multipart resolver
Web utilities
Spring Context
Application context
UI support
Validation
JNDI, EJB support & Remoting
Mail
Spring Web MVC
Web MVC Framework
Web Views
JSP / Velocity
PDF / Excel
Spring Core
Supporting utilities
Bean container

# Servicios Web REST

- REST “Representational State Transfer”
- Es un tipo de arquitectura de software para desarrollar servicios web basada en HTTP.
- Permite crear servicios y aplicaciones que pueden ser usadas por cualquier dispositivo o cliente que entienda HTTP.

# Servicios Web REST (2)

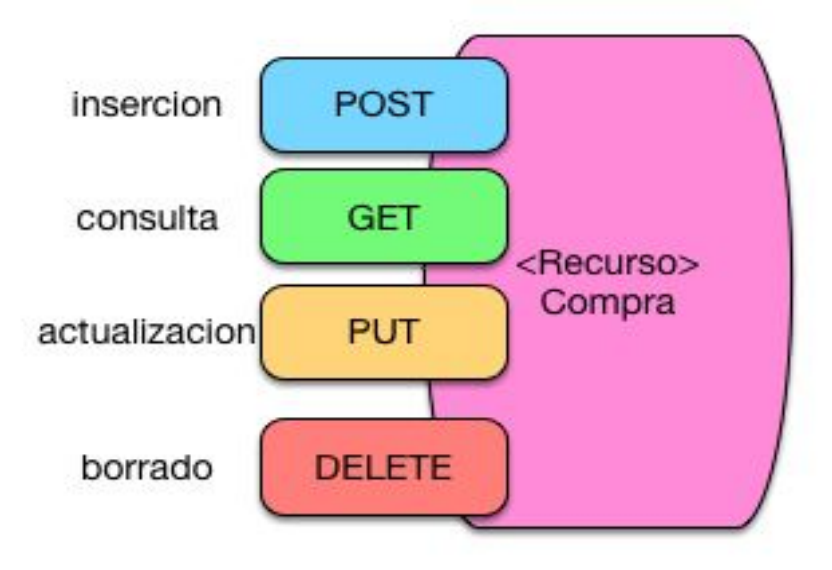
- En la arquitectura REST, los servicios no publican operaciones, publican recursos.
- Se exponen mediante URI'S (Uniform Resource Identifier) las cuales identifican de forma única al recurso.
- Una de las claves de REST es que no tiene estado (es *stateless*).
- El cliente debe pasar el estado en cada llamada (ejemplo: token).

# Servicios Web REST (3)

Métodos:

- GET: Para consultar y leer recursos.
- POST: Para crear recursos.
- PUT: Para editar recursos.
- DELETE: Para eliminar recursos.

# Servicios Web REST (4)



# Servicios Web REST (5)

Utiliza los códigos de respuesta nativos de HTTP:

- 200 OK
- 401 UNAUTHORIZED
- 404 NOT FOUND
- 500 INTERNAL SERVER ERROR

# RESTTemplate

- RESTTemplate es la clase que ofrece Spring para el acceso desde la parte cliente a Servicios REST.

- El cliente Rest se encarga de realizar la conexión HTTP, sólo hay que indicar la url del servicio a consumir.

# RESTTemplate

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-web</artifactId>
        <version>4.1.6.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.5.3</version>
    </dependency>
    ...
</dependencies>
```

# RESTTemplate

```
public File getFile(String authToken, String fileId) {
        final String uri = "https://www.googleapis.com/drive/v3/files/" + fileId;
    HttpEntity<String> entity = new HttpEntity<String>("parameters", this.getHeaders(authToken));
    RestTemplate restTemplate = new RestTemplate();
    ResponseEntity<File> result = restTemplate.exchange(uri, HttpMethod.GET, entity, File.class);
    return result.getBody();
}
```