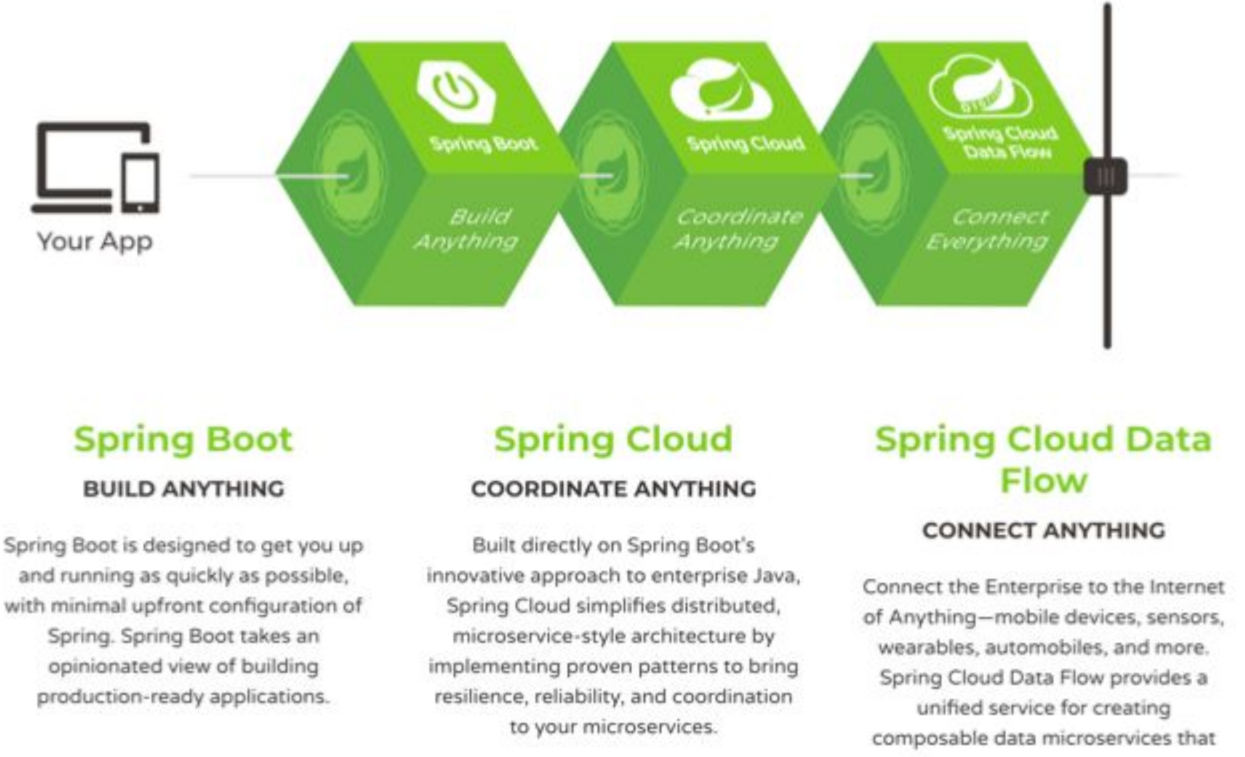
Spring Framework

Introduction to Spring Framework

Spring



Spring Boot

BUILD ANYTHING

Spring Boot is designed to get you up and running as quickly as possible, with minimal upfront configuration of Spring. Spring Boot takes an opinionated view of building production-ready applications.

Spring Cloud

COORDINATE ANYTHING

Built directly on Spring Boot's innovative approach to enterprise Java, Spring Cloud simplifies distributed, microservice-style architecture by implementing proven patterns to bring resilience, reliability, and coordination to your microservices.

Spring Cloud Data Flow

CONNECT ANYTHING

Connect the Enterprise to the Internet of Anything—mobile devices, sensors, wearables, automobiles, and more. Spring Cloud Data Flow provides a unified service for creating composable data microservices that

Spring Projects

Modular by design

Spring Framework	Spring Boot	Spring Data Flow
Spring Cloud	Spring Data	Spring Integration
Spring Batch	Spring Security	Spring AMQP
Spring LDAP	Spring WebFlow	Spring REST Doc

https://spring.io/projects

Spring Framework

What ?

Application development framework for Java

Create high performing, testable and reusable code without any lock-in

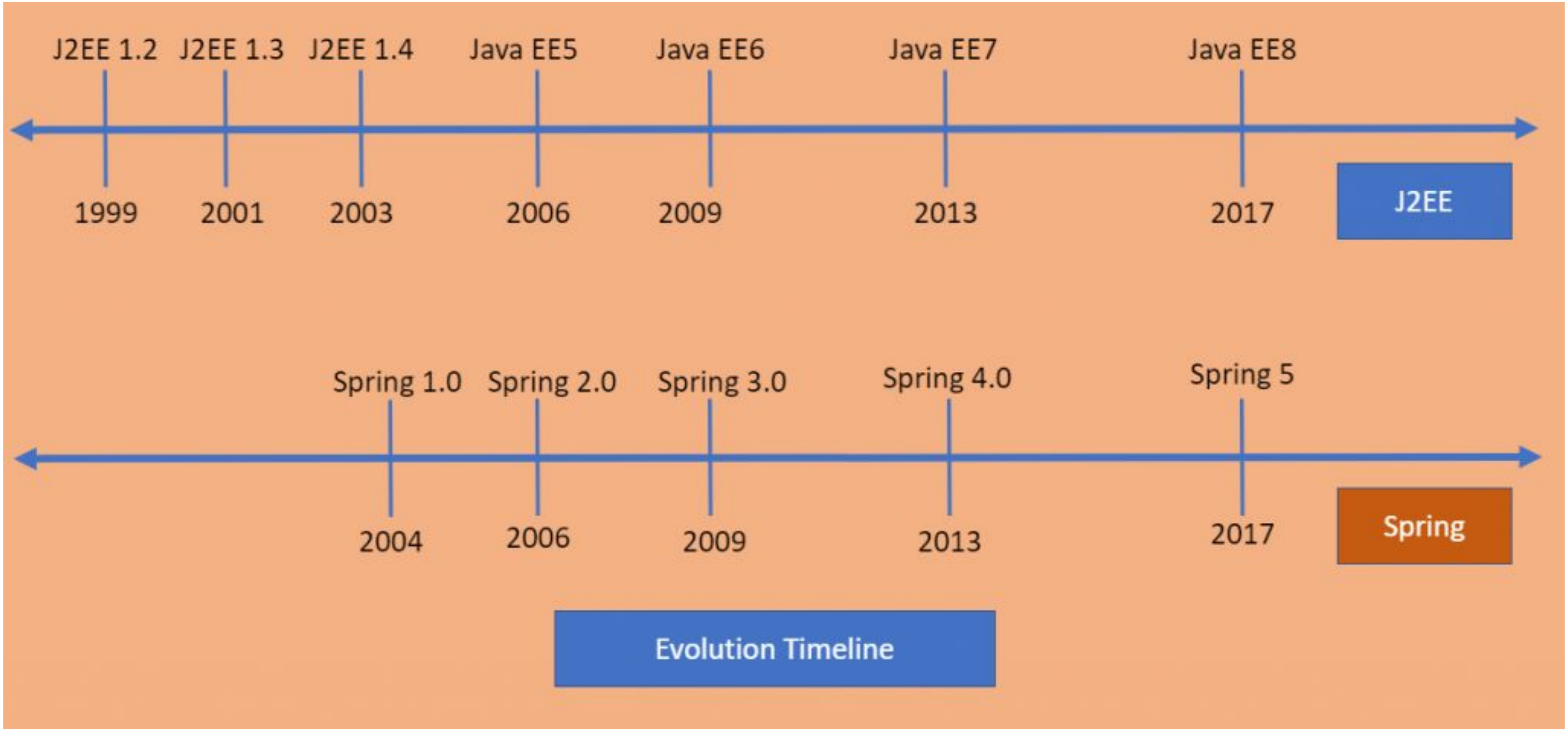
Open source since 2003

Reduce code and speed up development

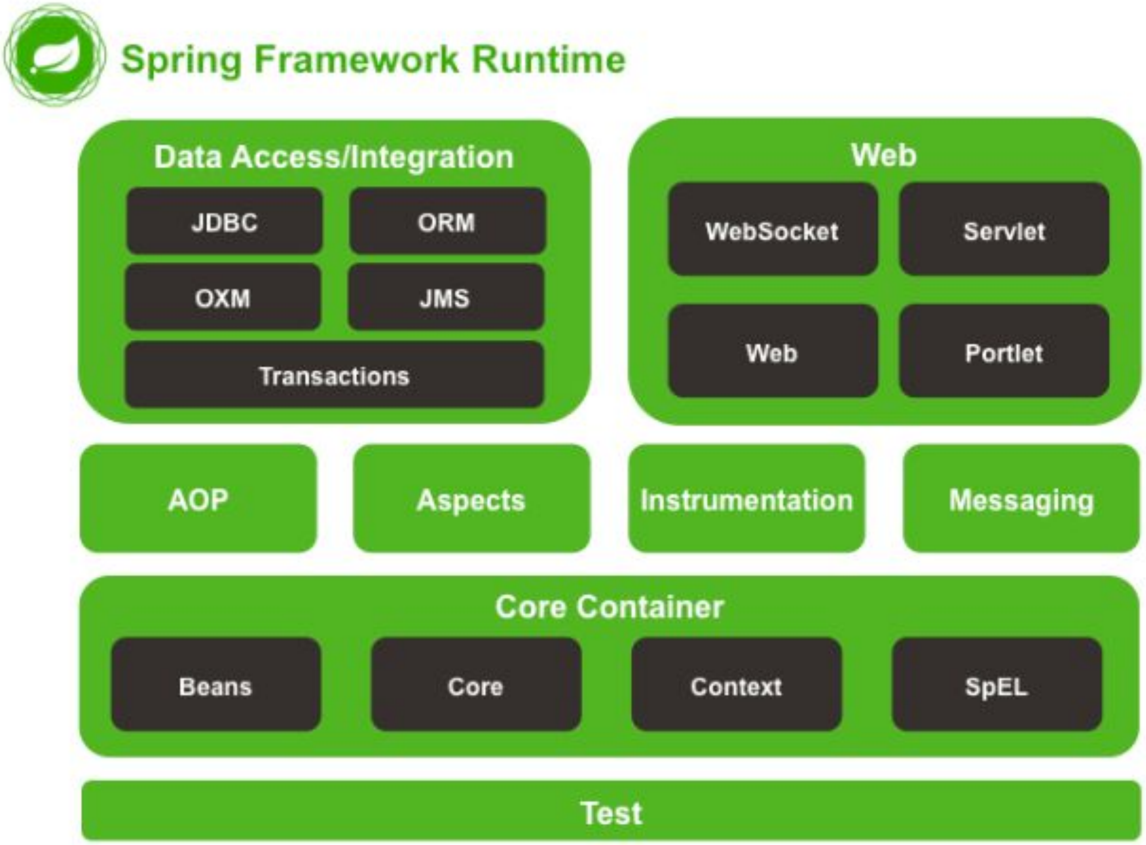
Current version **5.x**

Required J**DK 8+**

History of Spring framework



Overview of Spring Framework



Core Container

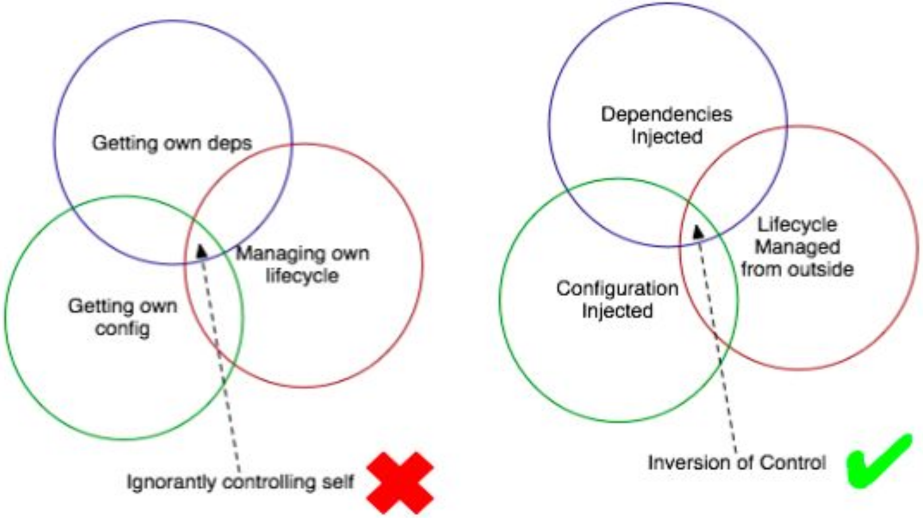
Core and Beans

Context

Expression language

Core and Beans

Provide the fundamental parts of framework
Including **IoC** and **Dependency Injection**



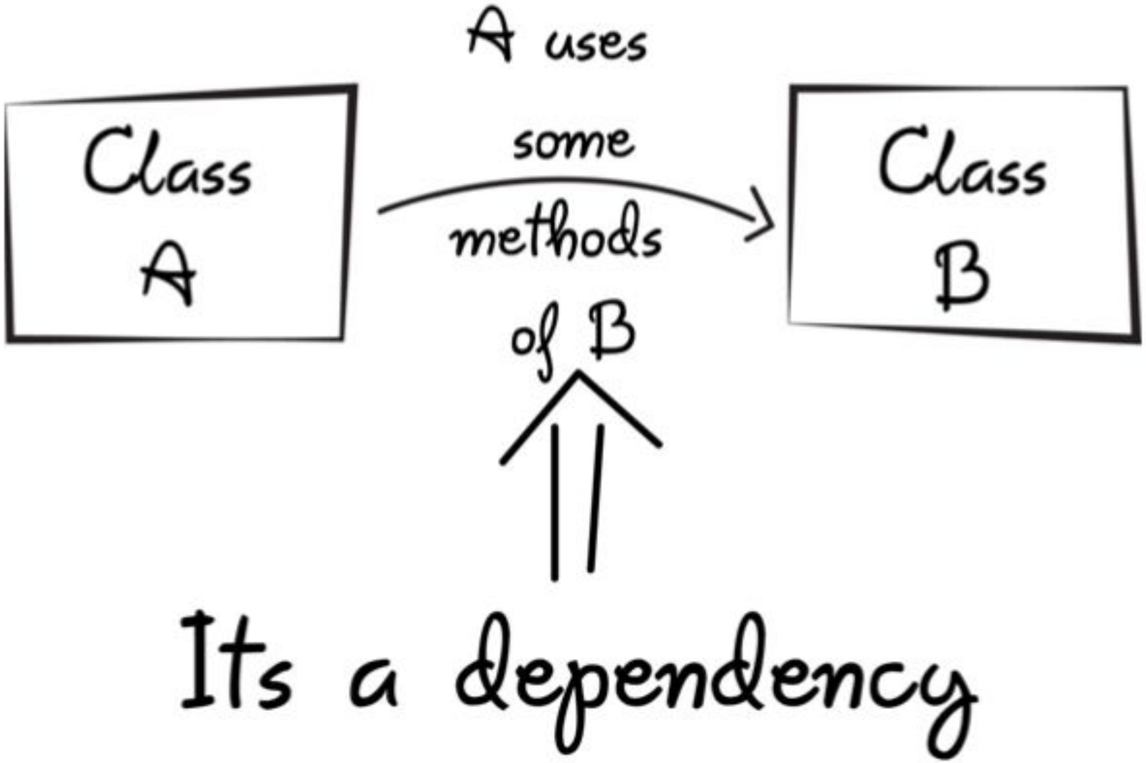
https://www.martinfowler.com/articles/injection.html

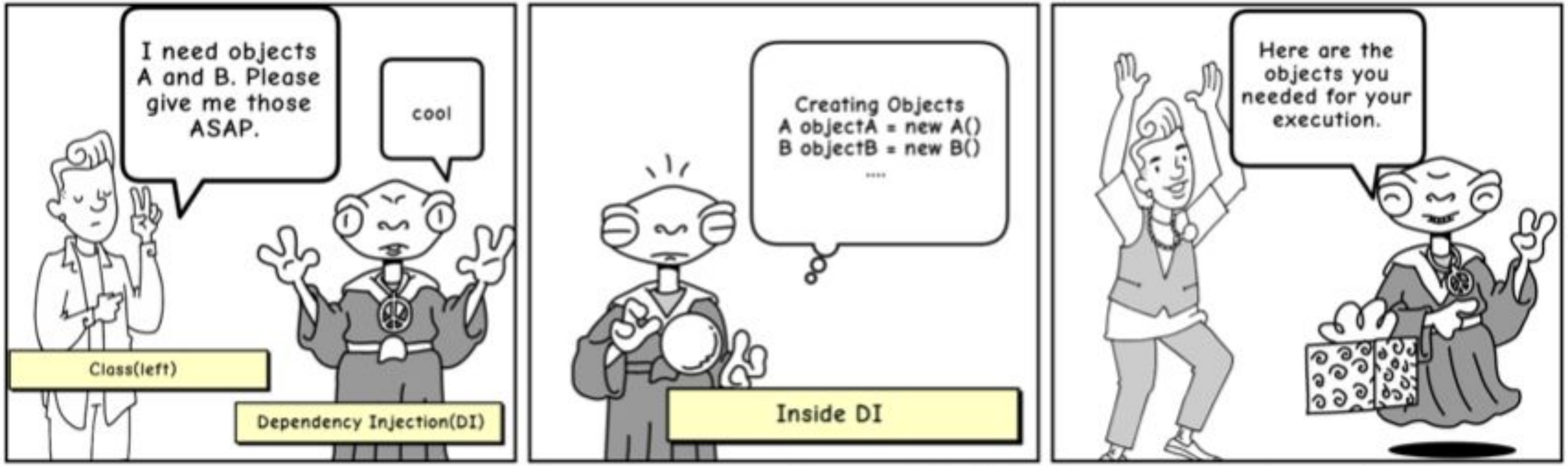
Inversion of Control (IoC)

Concept in application development

Don’t call me, I will call you

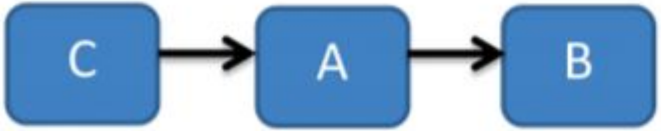
Dependency Injection



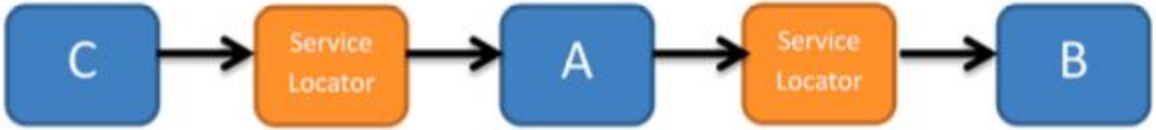


This comic was created at www.MakeBeliefsComix.com. Go there and make one now!

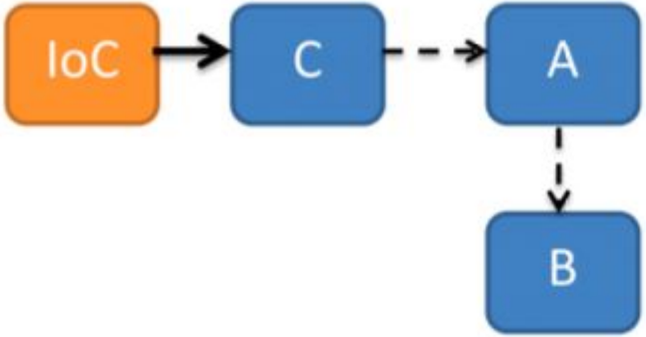
Class Dependencies



Service Location / Active Calling



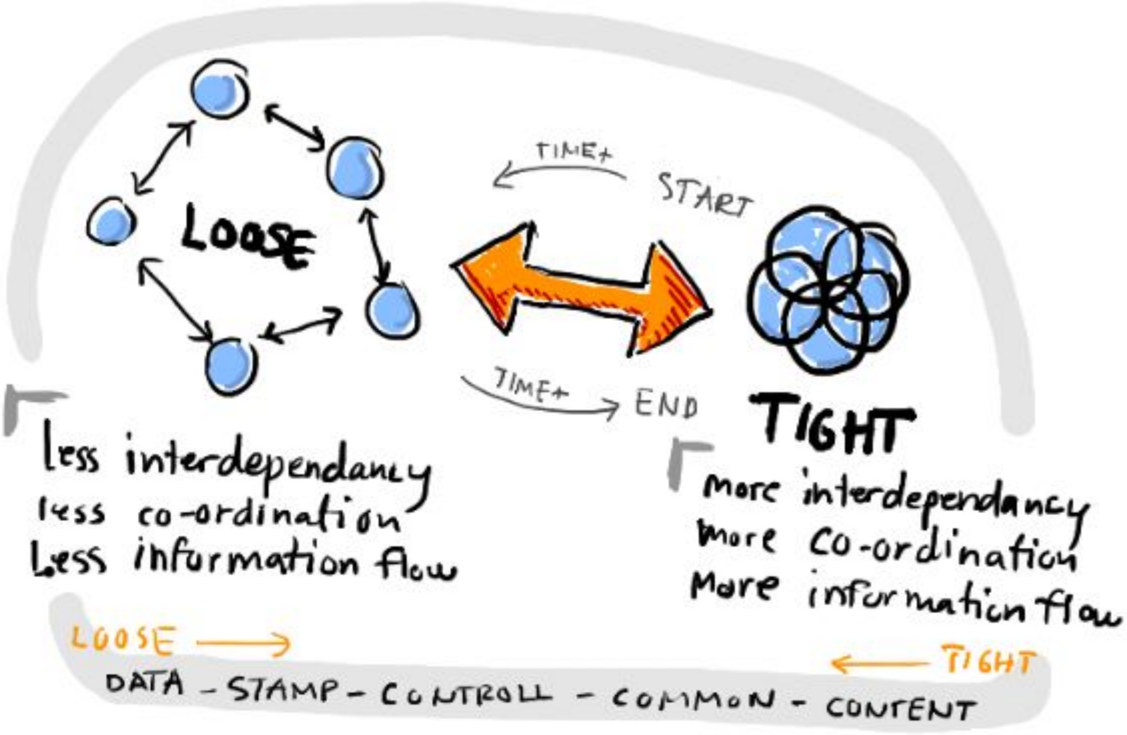
IoC / DI / Auto-Wiring / Passive Calling



Types of Dependency Injection

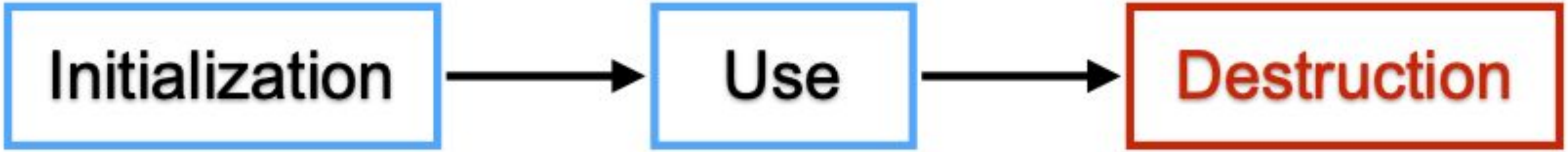
Constructor injection
Property/Setter injection
Method injection
Interface injection

Tight coupling
Loose coupling



Back to Spring Framework

Application Lifecycle



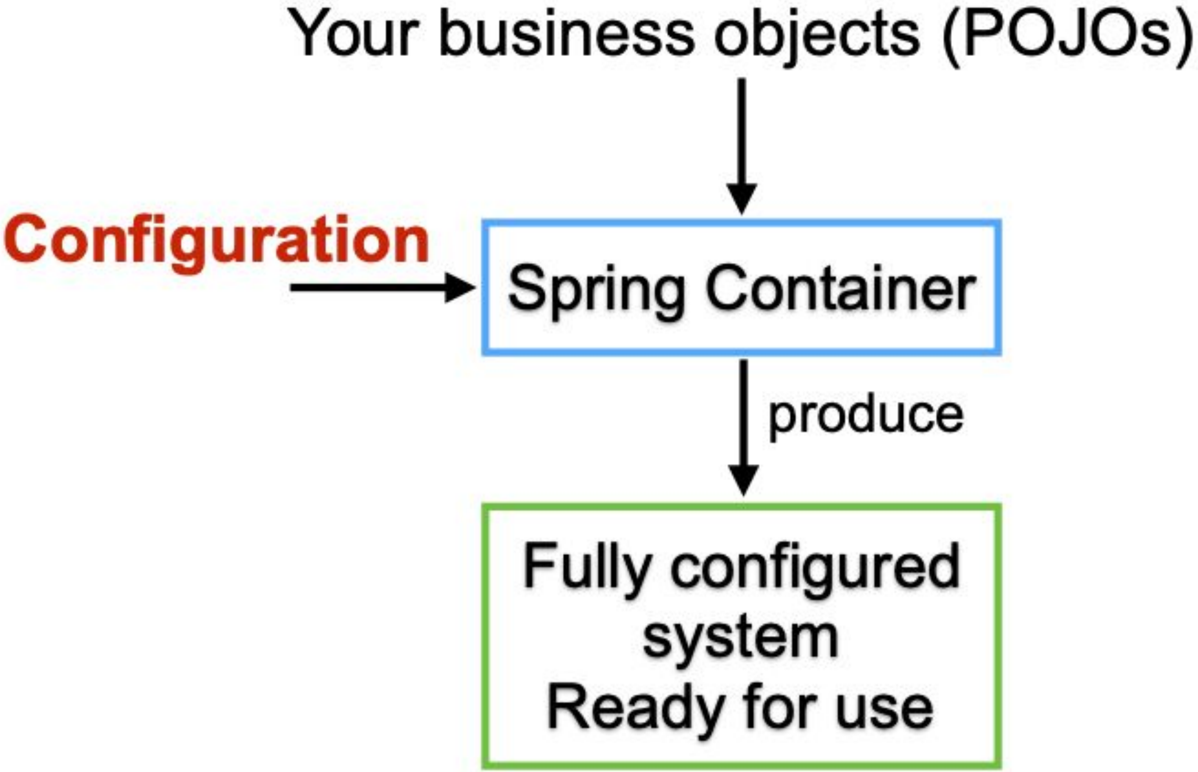
Initialization → Use → Destruction

Initialization
Create services
Allocate resources

Use
Process by requests

Destruction
Shutdown
Release resources

Spring IoC container



Your business objects (POJOs)

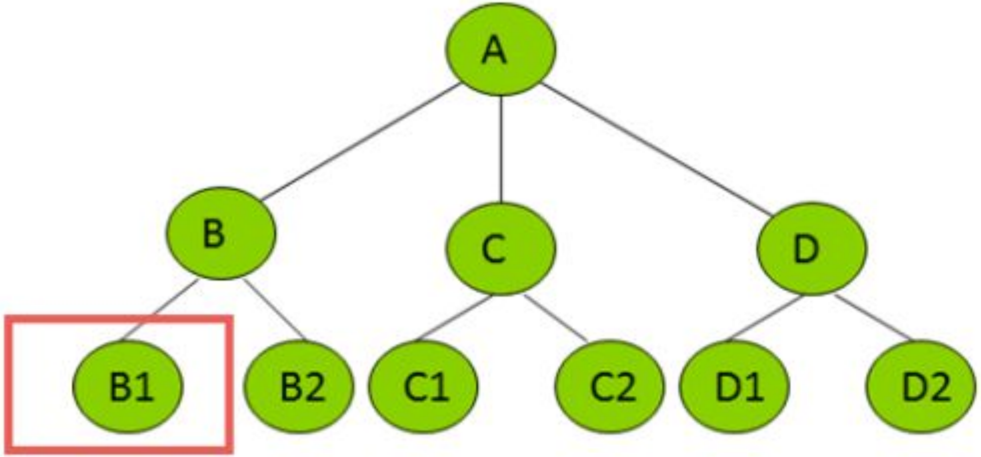
Configuration

Spring Container

produce

Fully configured system
Ready for use

Spring IoC container



Bean B1

Configuration

From XML file

Annotation-based configuration (2.5)

Java-based configuration (3.0)

Beans

Bean Definitions

Package-qualified class name

Bean behavioral (scope, lifecycle, callback)

Reference to other beans

Other configuration setting to create new object

Spring IoC container manages one of more beans

Beans are created with configuration

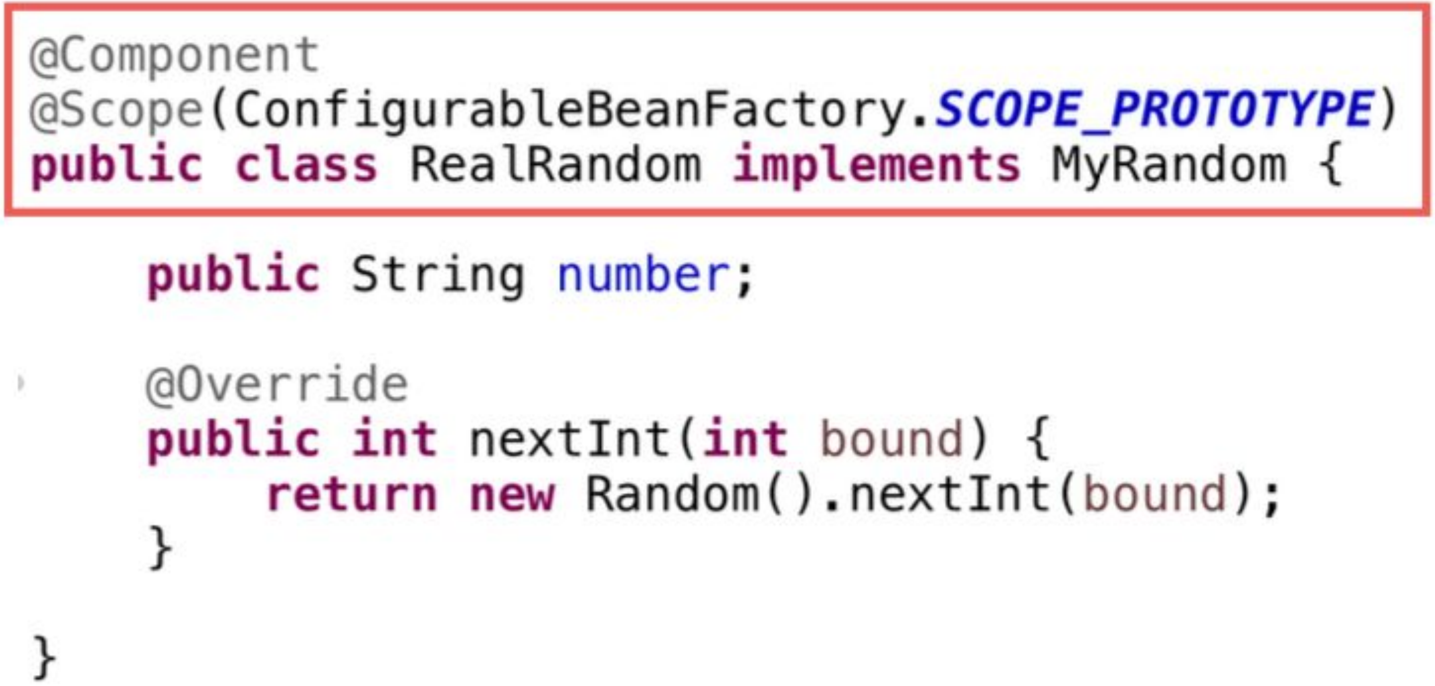
Bean Definitions

Property	Section
Class	Instantiating beans
Name	Naming beans
Scope	Beans scopes
Constructor arguments	Dependency Injection
Properties	Dependency Injection
Autowiring mode	Autowiring collaborators
Lazy initialization mode	Lazy-initialized beans

Bean Scopes

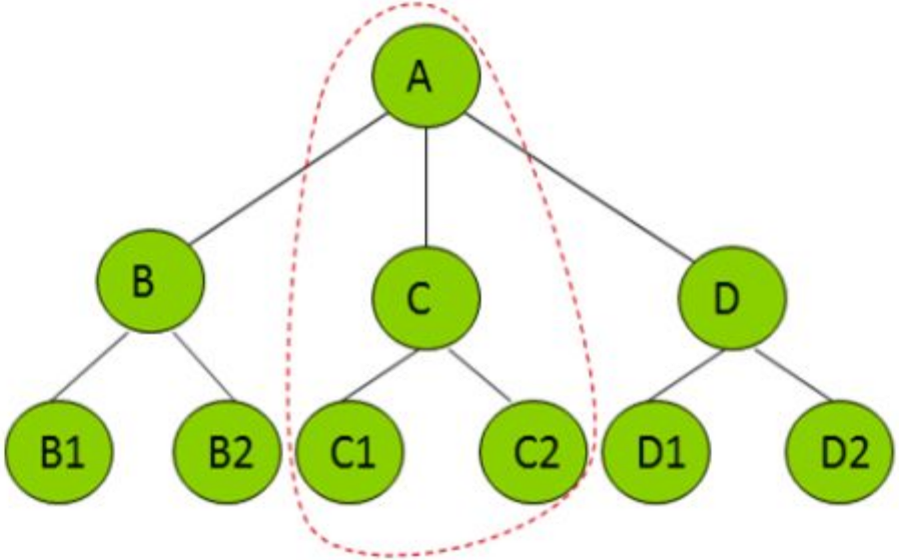
Scope	Description
singleton	Single instance for each container
prototype	Single bean definition to any number of object instances.
request	HTTP request
session	HTTP session
application	ServletContext

Change scope of bean



```
@Component
@Scope(ConfigurableBeanFactory.SCOPE_PROTOTYPE)
public class RealRandom implements MyRandom {

    public String number;

    @Override
    public int nextInt(int bound) {
        return new Random().nextInt(bound);
    }

}
```

Lazy-load dependencies



BeanFactory ?

Interface defines basic functionality for Spring container

Factory design pattern

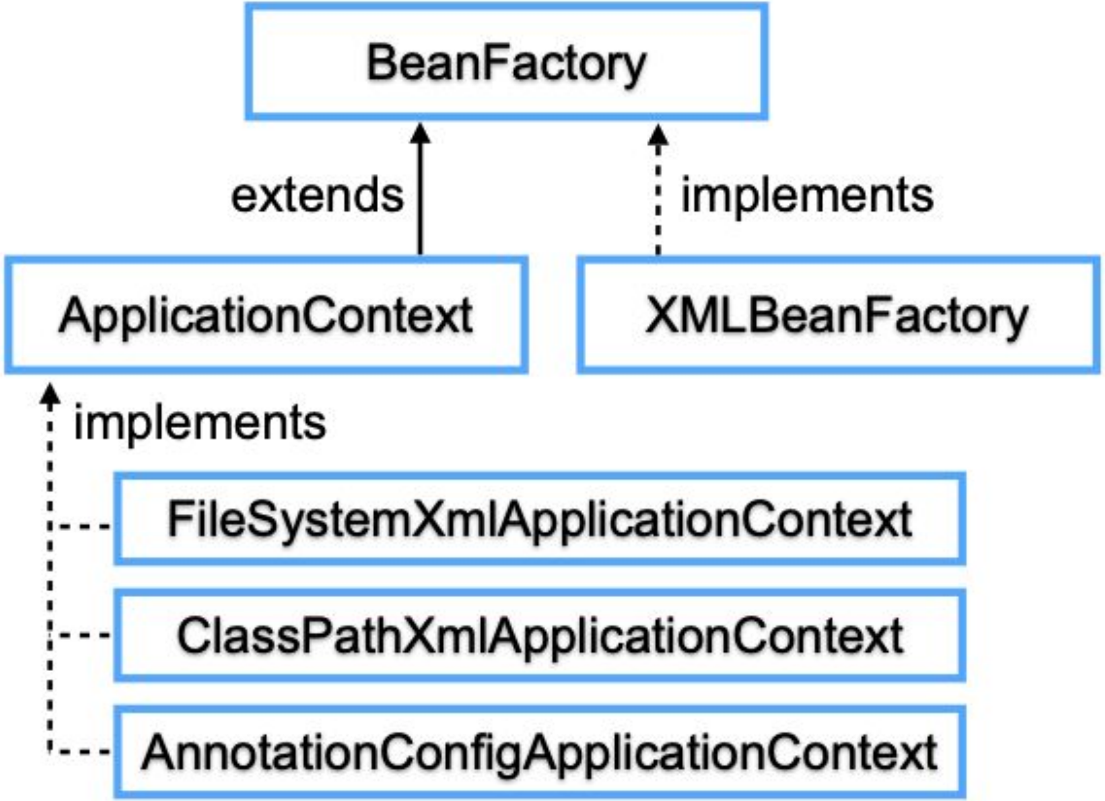
Load beans from configuration source

Instantiated the bean when requested

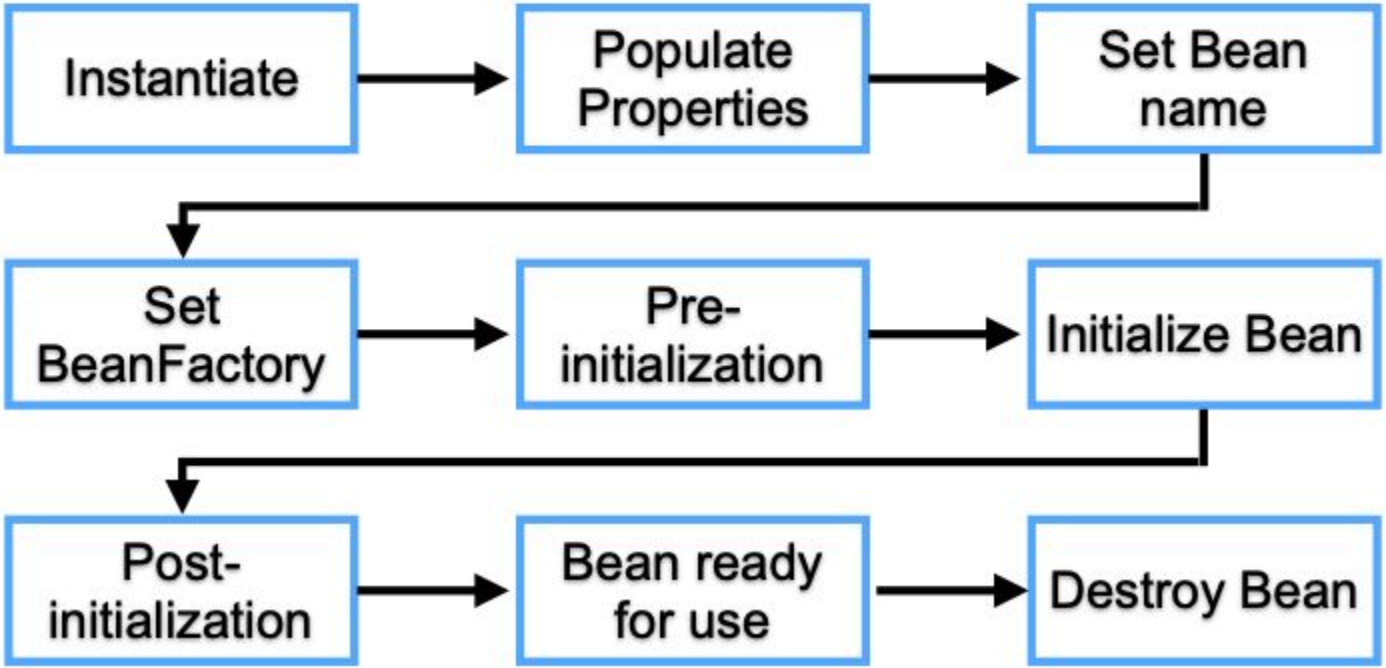
Wire dependencies and properties for beans

Manage the bean lifecycle

BeanFactory ?


BeanFactory
extends
implements
ApplicationContext
XMLBeanFactory
implements
FileSystemXmlApplicationContext
ClassPathXmlApplicationContext
AnnotationConfigApplicationContext

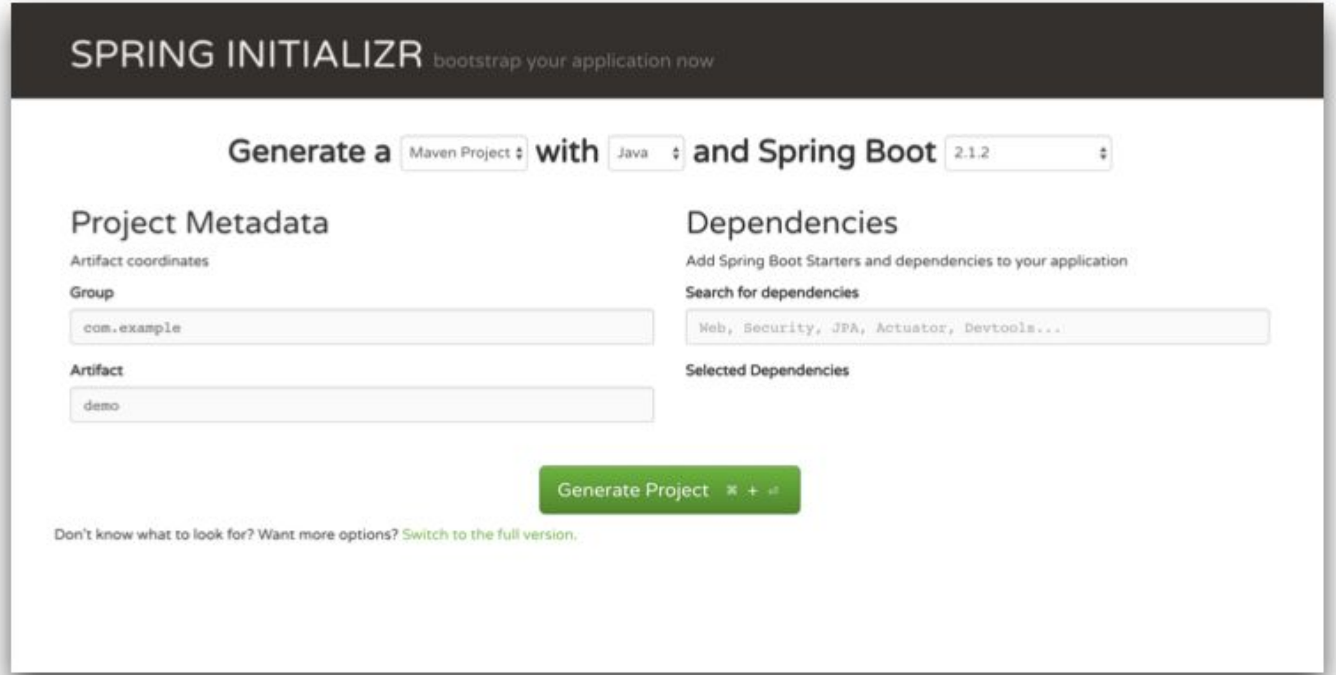
Lifecycle of BeanFactory



Let’s start

Create new Project

Use **spring initializr**



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 2.1.2

Project Metadata

Artifact coordinates

Group: com.example

Artifact: demo

Dependencies

Add Spring Boot Starters and dependencies to your application

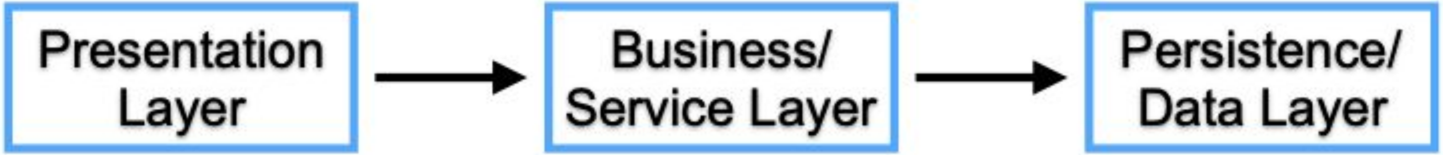
Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Don't know what to look for? Want more options? Switch to the full version.

https://start.spring.io/

Layer of application



Using Spring to manage dependencies

@Component

@Autowired

Constructor and Setter injection

@Primary

@Qualifier

Scope of beans

Singleton
Prototype

What are difference of …

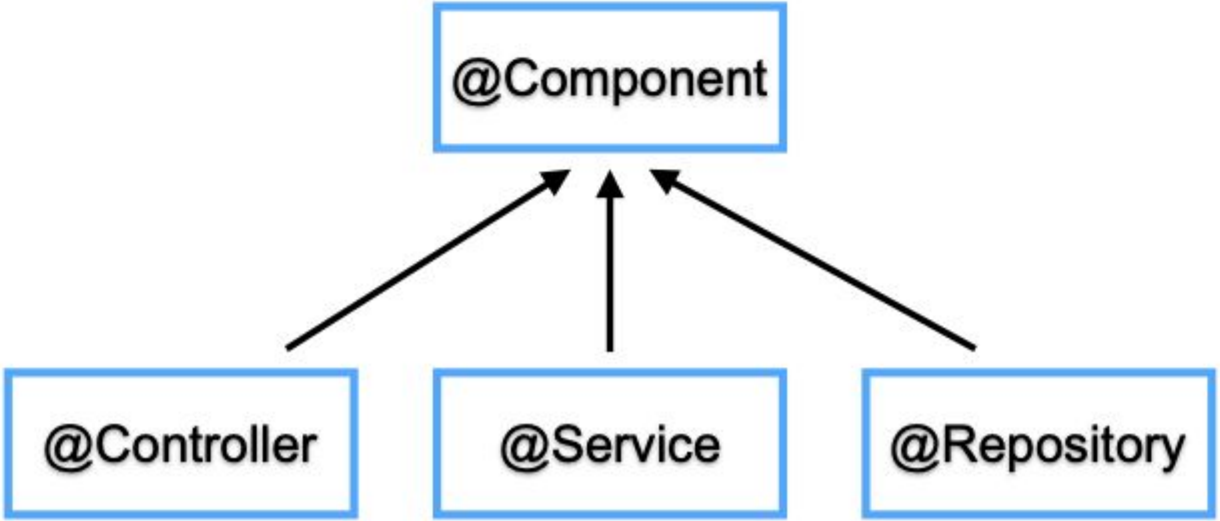
@Component

@Controller

@Service

@Repository

What are difference of …



What are difference of …

@Component

Generic stereotype for any component or bean

@Controller

Stereotype for the presentation layer (Spring MVC)

@Service

Stereotype for the service layer

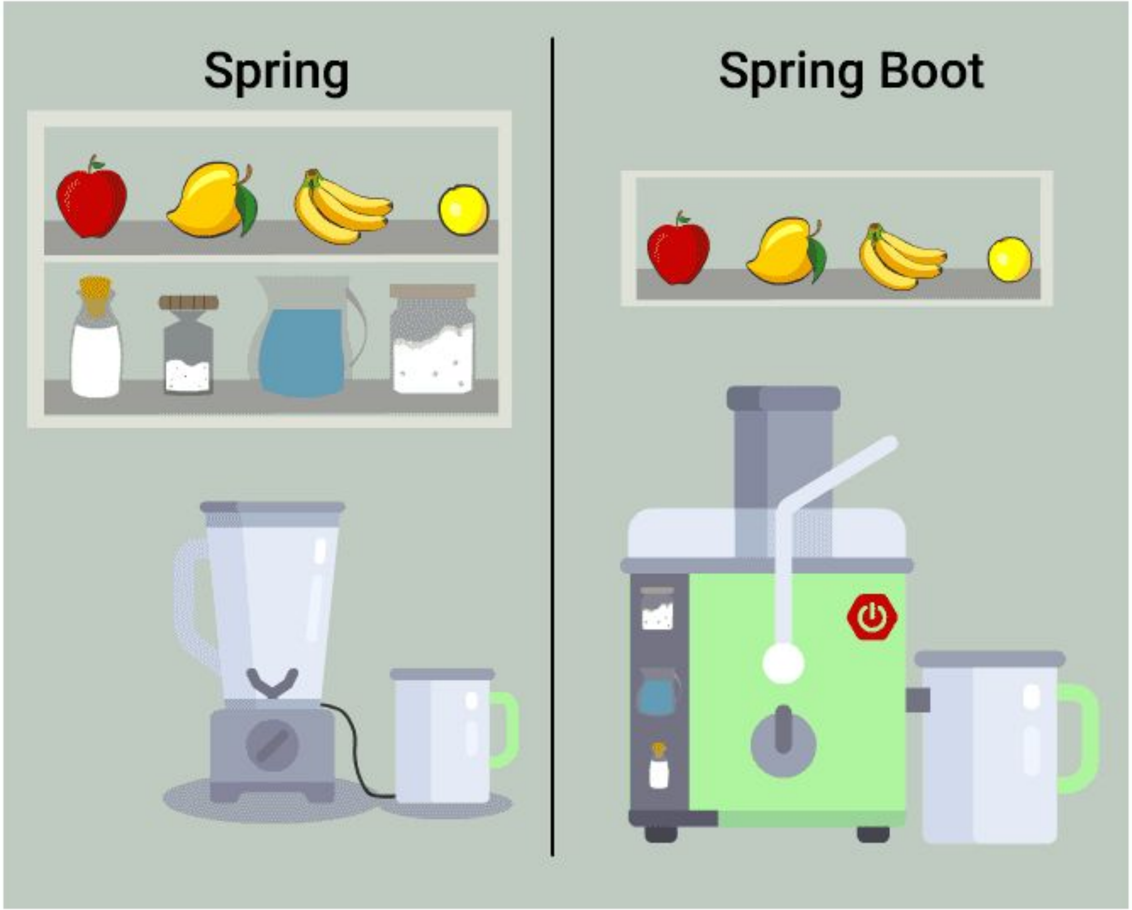
@Repository

Stereotype for the persistence layer

Why Spring is popular ?

Enable testable code
No plumbing code
Flexible architecture
Staying current

Spring Boot



Spring

Spring Boot

REST

REST

REpresentation **S**tate **T**ransfer

The style of software architecture behind RESTful services

Defined in 2000 by Roy Fielding

Goals

Scalability

Generality of interfaces

Independent deployment of components

RESTful service

REST Request Messages

RESTful request is typically in form of **U**niform **R**esource **I**dentifiers (URI)

REST Request Messages

RESTful request is typically in form of **U**niform **R**esource **I**dentifiers (URI)

Structure of URI depend on specific service

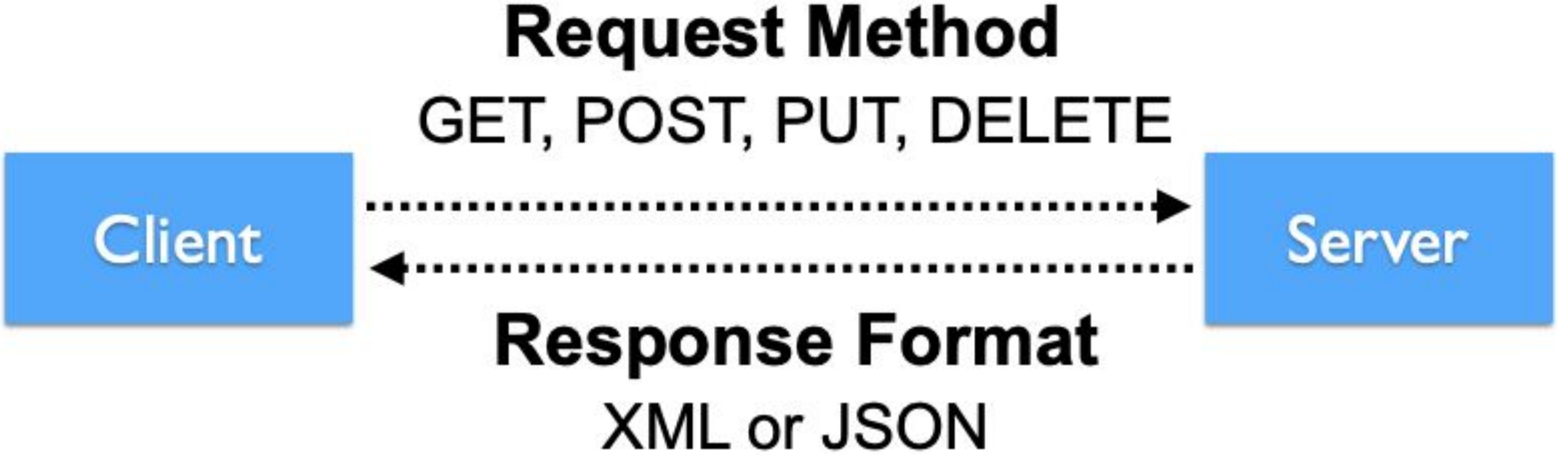
REST Request Messages

RESTful request is typically in form of **U**niform **R**esource **I**dentifiers (URI)

Structure of URI depend on specific service

Request can include parameter and data in body of request as XML, JSON etc.

REST Request & Response



Request Method

GET, POST, PUT, DELETE

Client

Server

Response Format

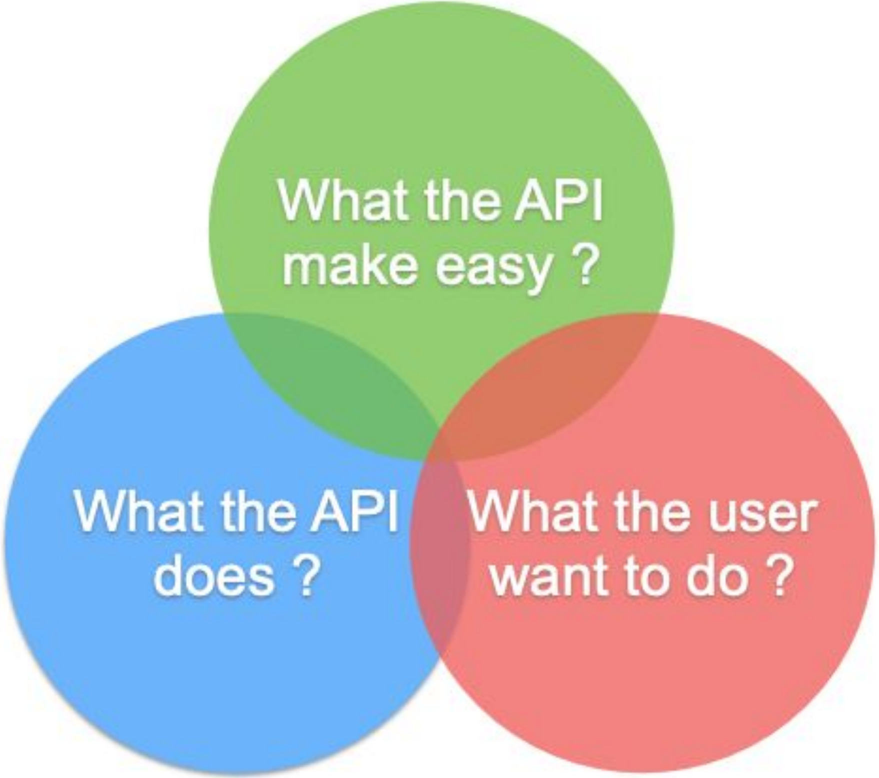
XML or JSON

HTTP Methods meaning

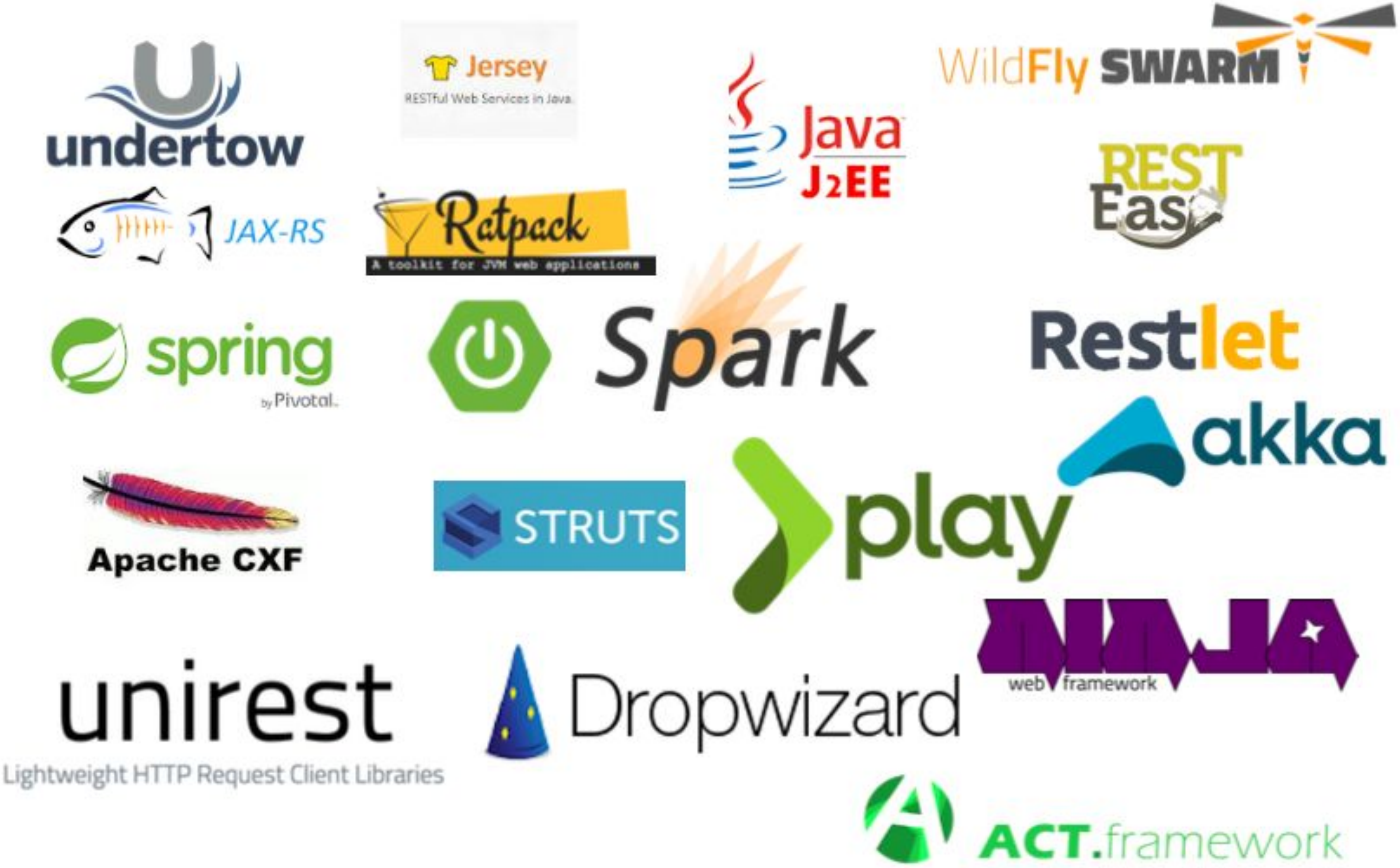
Method	Meaning
GET	Read data
POST	Create/Insert new data
PUT/PATCH	Update data or insert if a new id
DELETE	Delete data

Response format ?

Good APIs ?


What the API make easy ?
What the API does ?
What the user want to do ?

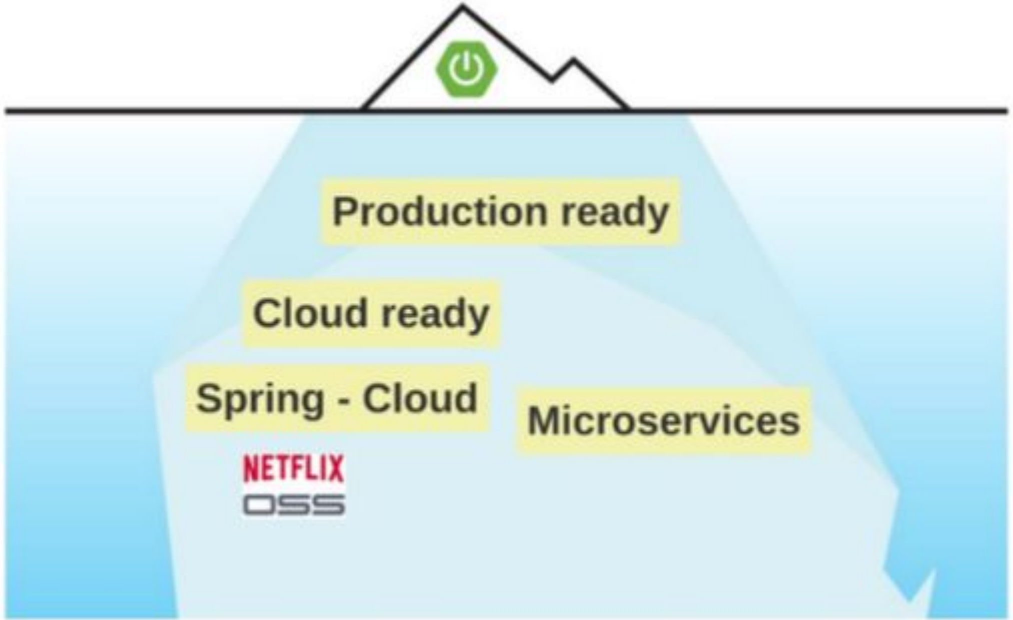
Java Framework for RESTful



Hello Spring Boot 2.x

Why ?

Application skeleton generator
Reduce effort to add new technologies


Production ready
Cloud ready
Spring - Cloud
Microservices
NETFLIX
OSS

What ?

Embedded application server

Integration with tools/technologies (starter)

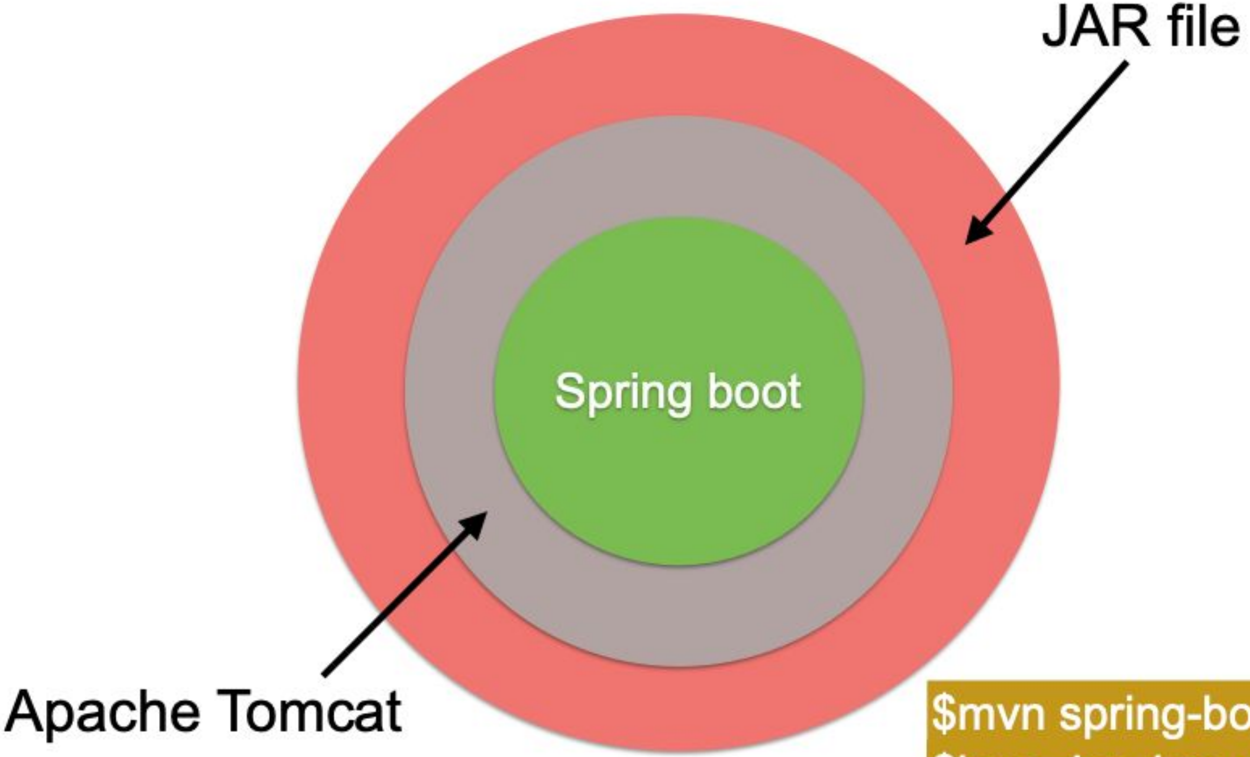
Production tools (monitoring, health check)

Configuration management

Dev tools

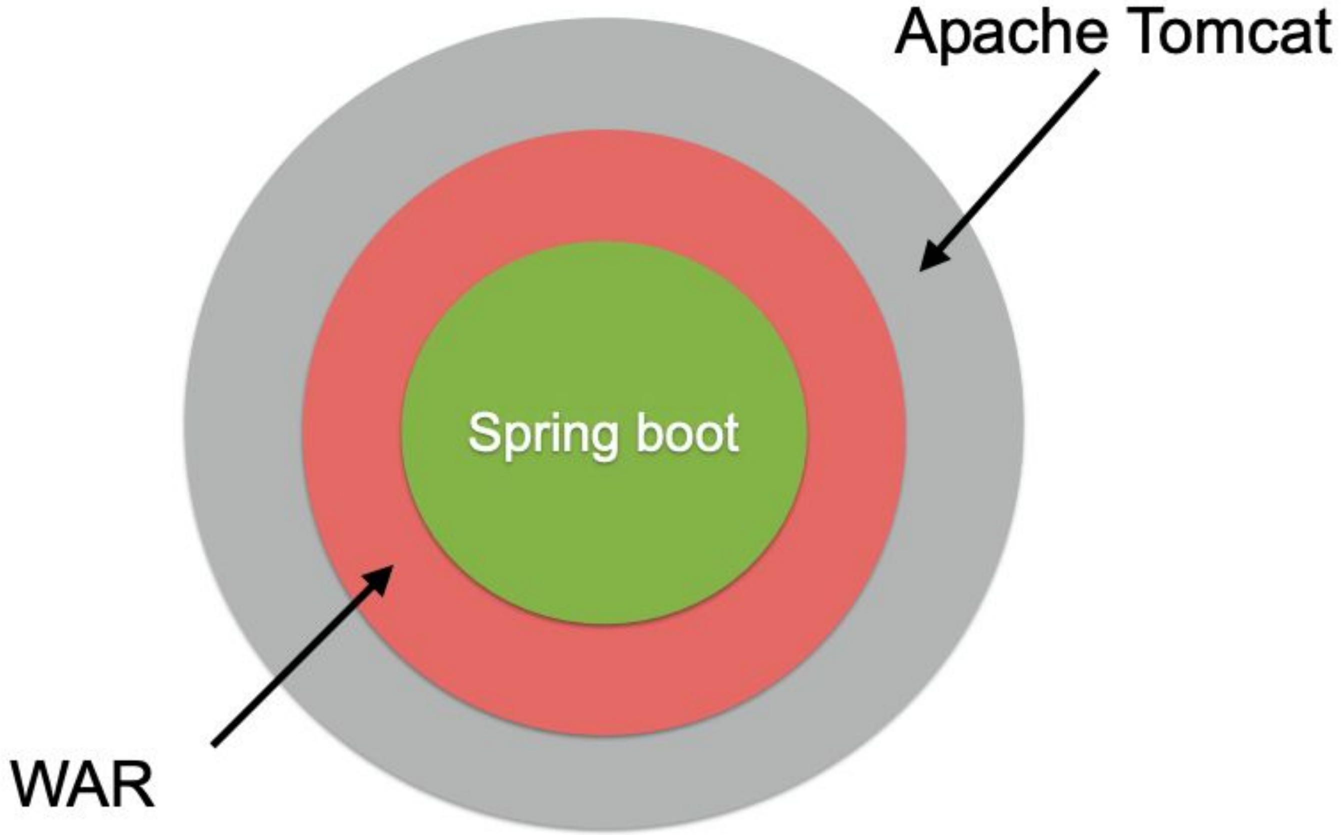
No source code generation, no XML

How ?



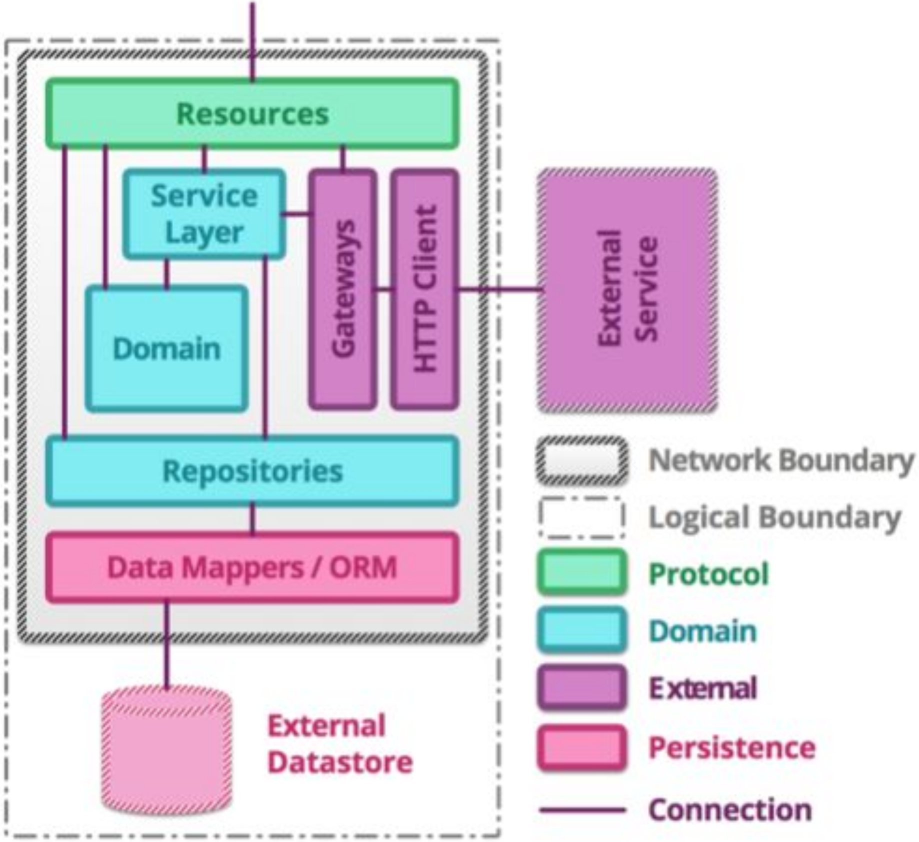
```
$mvn spring-boot:run
$java -jar demo.jar
```

How ?

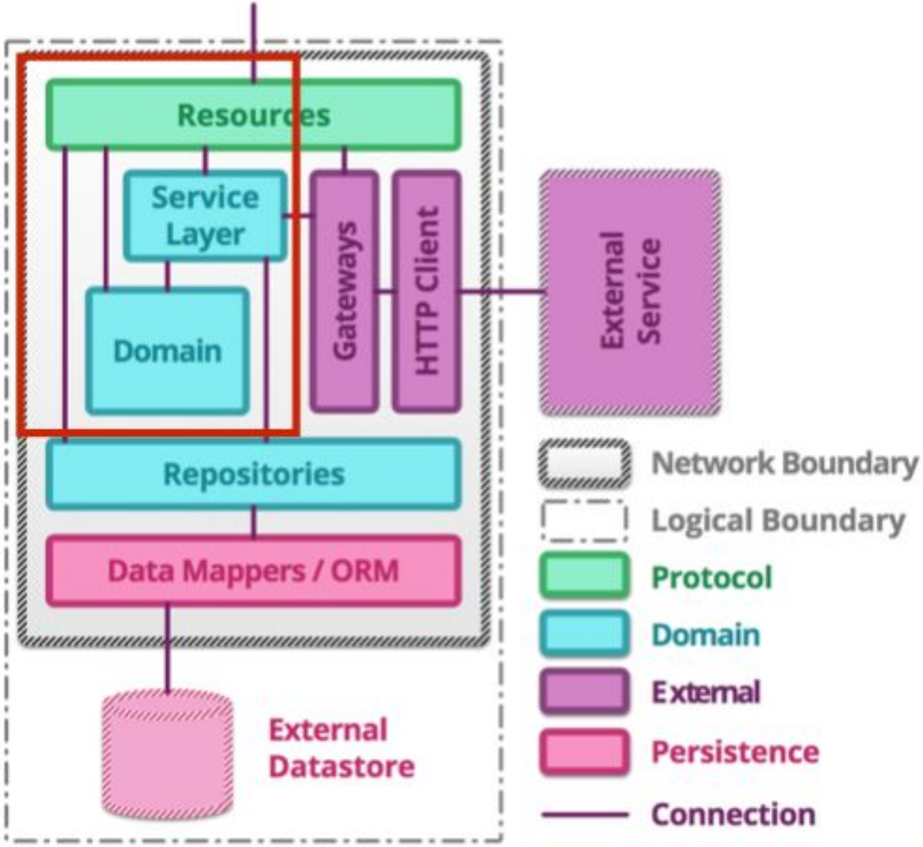


Building RESTful API with Spring Boot

Service Structure



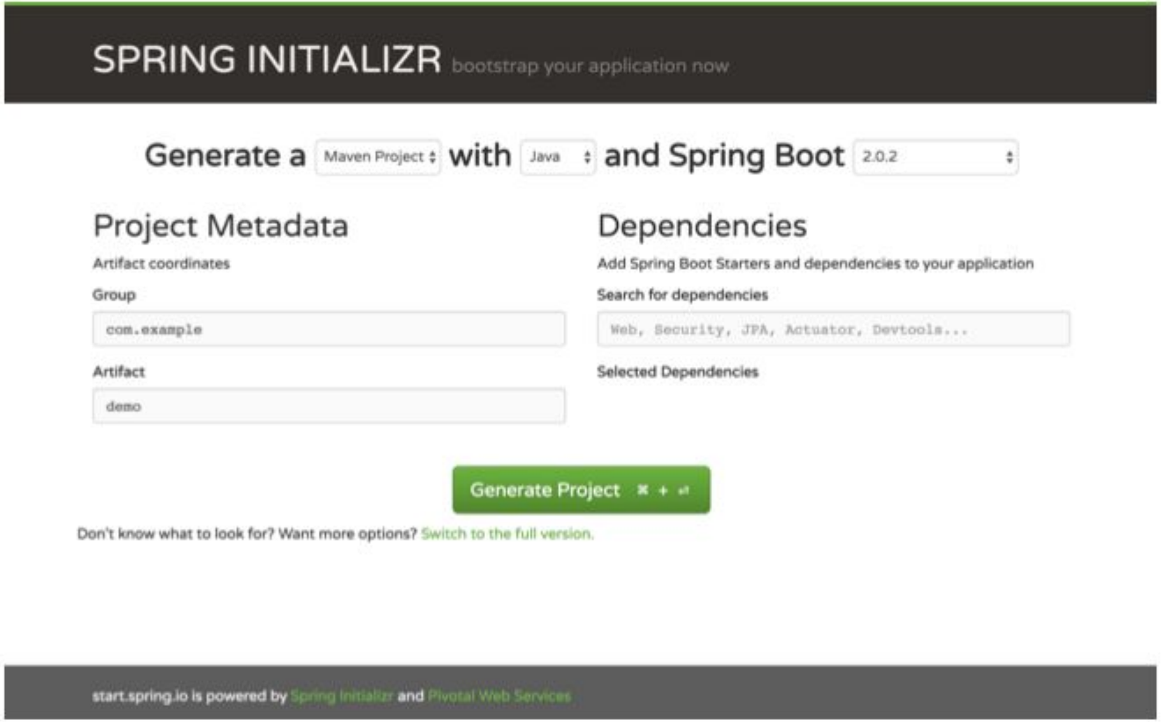
Service Structure



Create project with Spring Initializr

Spring Initializr

https://start.spring.io/



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 2.0.2

Project Metadata

Artifact coordinates

Group: com.example

Artifact: demo

Dependencies

Add Spring Boot Starters and dependencies to your application

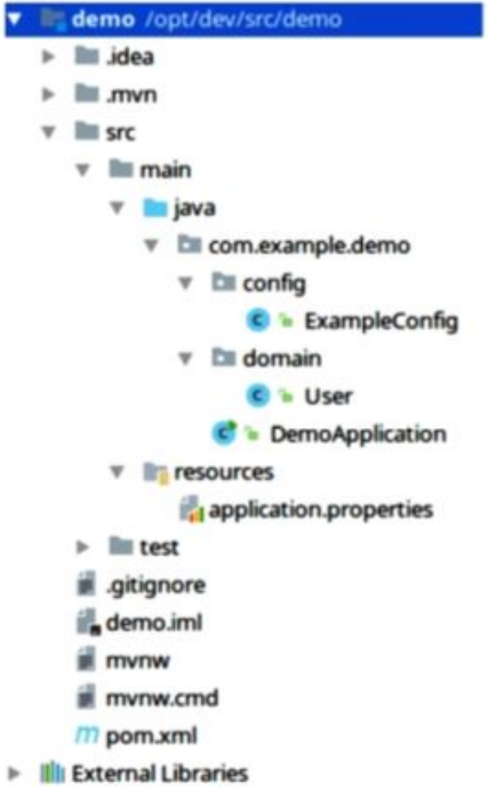
Search for dependencies: Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Don't know what to look for? Want more options? Switch to the full version.

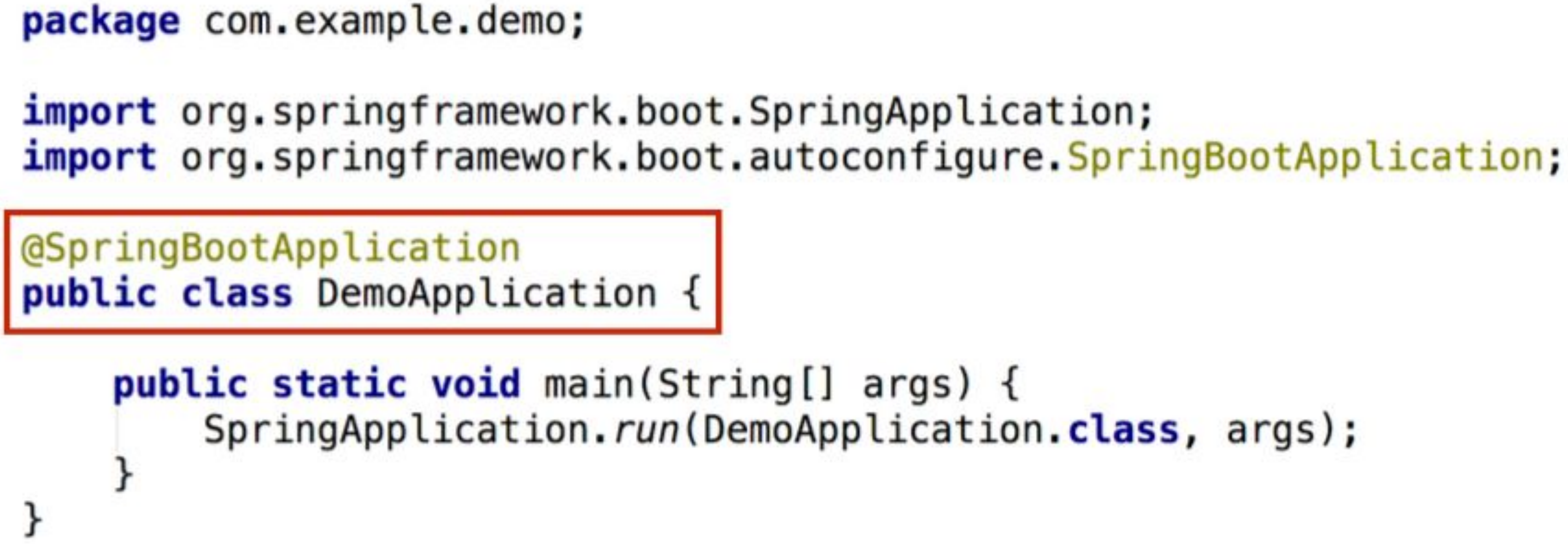
start.spring.io is powered by Spring Initializr and Pivotal Web Services

Structure of Spring Boot



```
▼ demo /opt/dev/src/demo
    ▶ .idea
    ▶ .mvn
    ▼ src
        ▼ main
            ▼ java
                ▼ com.example.demo
                    ▼ config
                          ExampleConfig
                    ▼ domain
                          User
                      DemoApplication
            ▼ resources
                  application.properties
        ▶ test
      .gitignore
      demo.iml
      mvnw
      mvnw.cmd
      pom.xml
▶ External Libraries
```

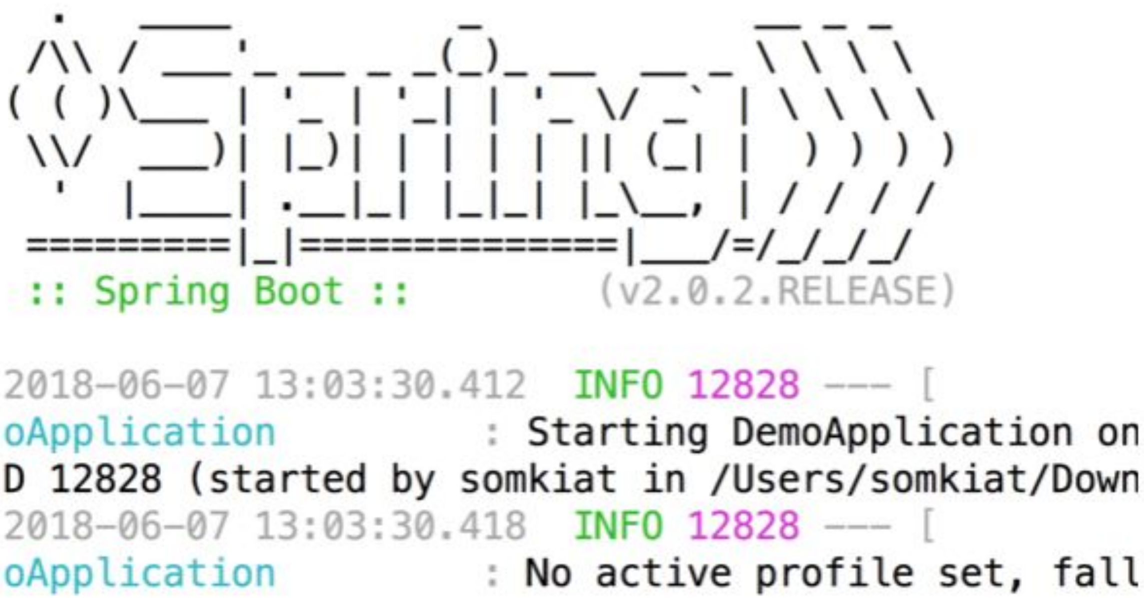
Spring Boot main class



```java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

Run project (Dev mode)

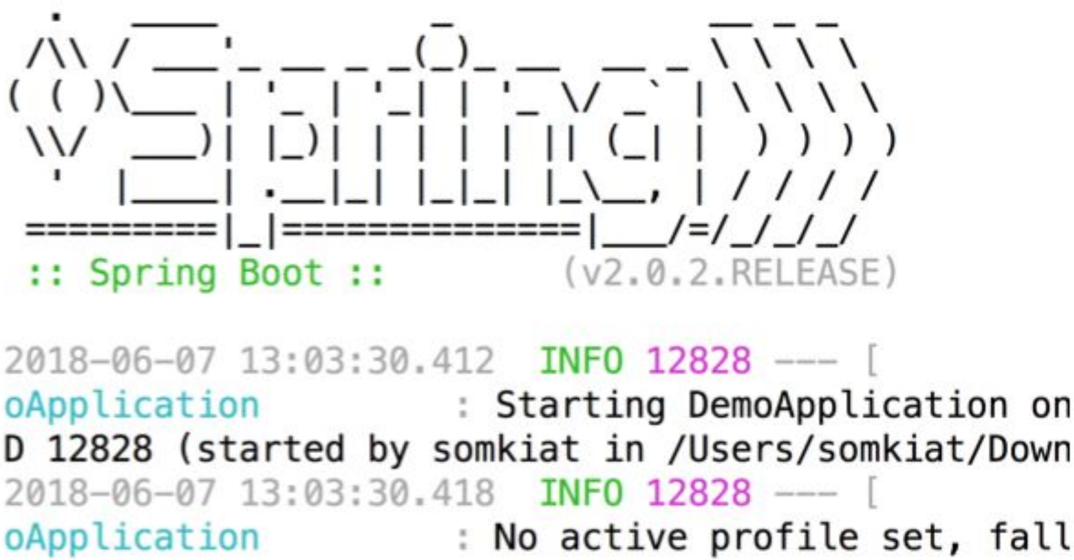
$./mvnw spring-boot:run



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.2.RELEASE)

2018-06-07 13:03:30.412  INFO 12828 --- [
oApplication                : Starting DemoApplication on
D 12828 (started by somkiat in /Users/somkiat/Down
2018-06-07 13:03:30.418  INFO 12828 --- [
oApplication                : No active profile set, fall
```

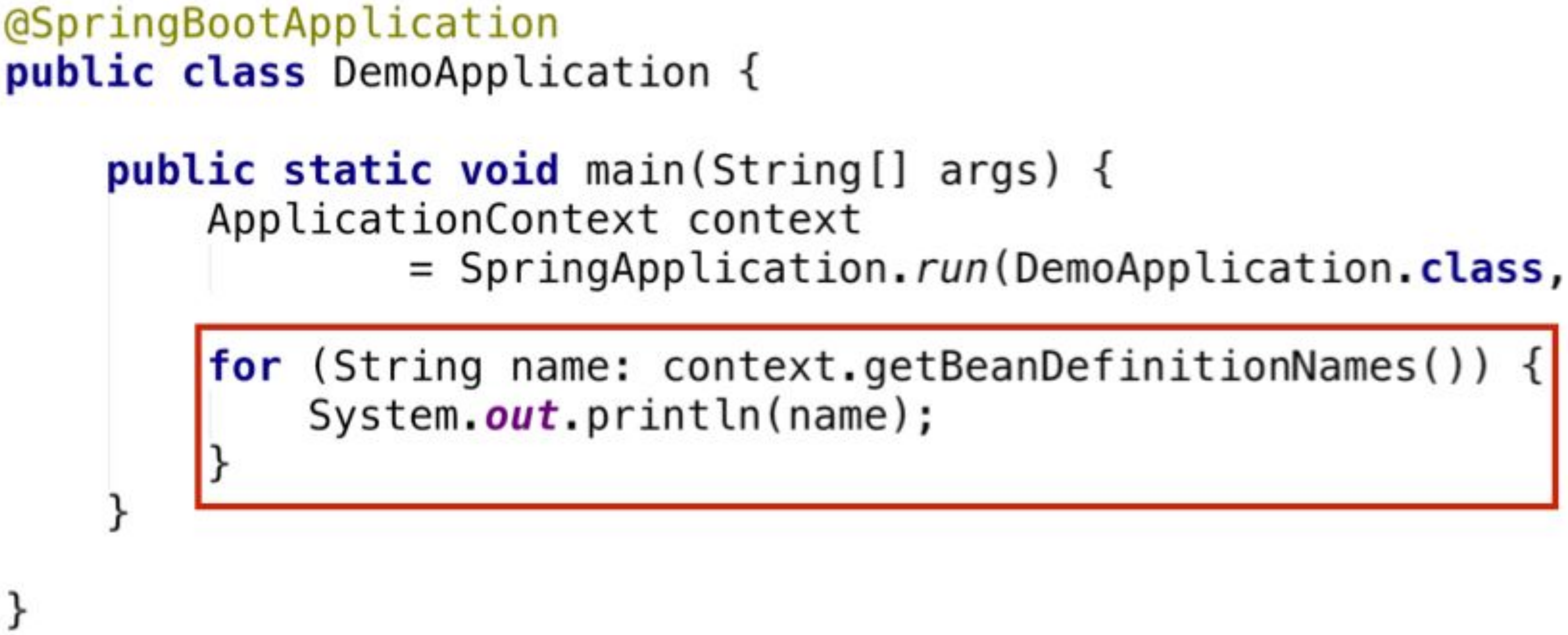
Run project (production mode)

```
$./mvnw package
$java -jar target/<file name>.jar
```



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.2.RELEASE)

2018-06-07 13:03:30.412  INFO 12828 --- [
oApplication             : Starting DemoApplication on
D 12828 (started by somkiat in /Users/somkiat/Down
2018-06-07 13:03:30.418  INFO 12828 --- [
oApplication             : No active profile set, fall
```

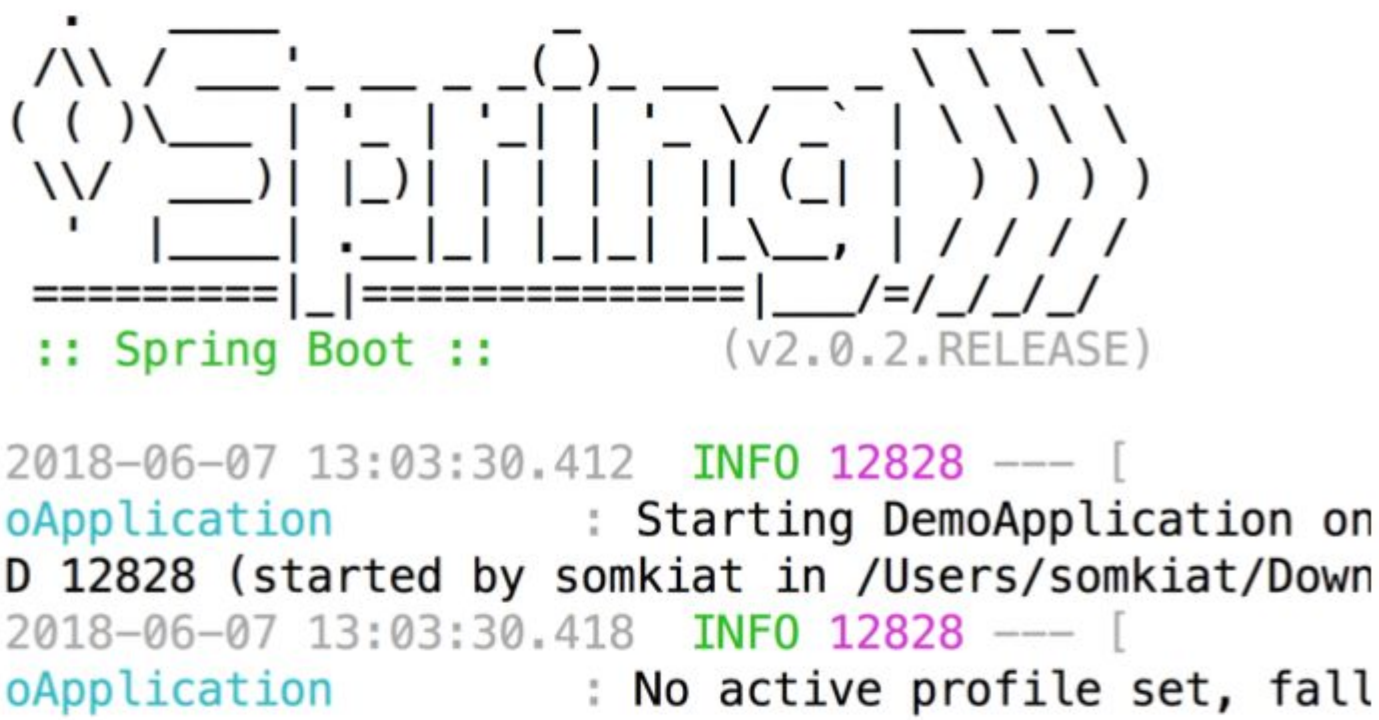
Display all beans !!



```java
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        ApplicationContext context
                = SpringApplication.run(DemoApplication.class,

        for (String name: context.getBeanDefinitionNames()) {
            System.out.println(name);
        }
    }

}
```

Custom Banner

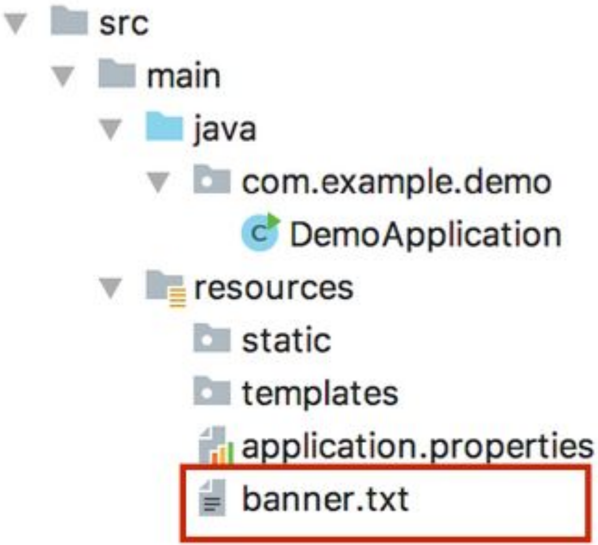
Custom banner



```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.2.RELEASE)

2018-06-07 13:03:30.412  INFO 12828 --- [
oApplication                 : Starting DemoApplication on
D 12828 (started by somkiat in /Users/somkiat/Down
2018-06-07 13:03:30.418  INFO 12828 --- [
oApplication                 : No active profile set, fall
```

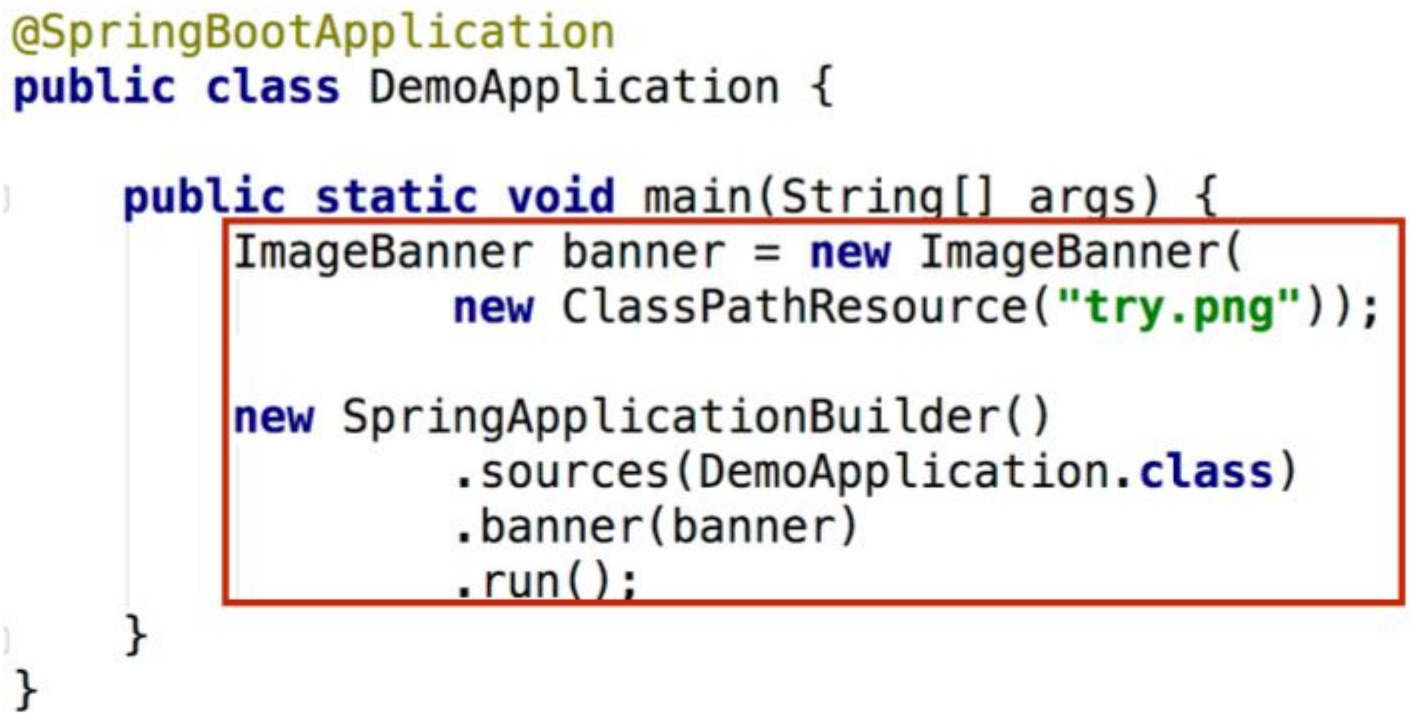
Custom banner (1)

Create file banner.txt or banner.png in resources folder



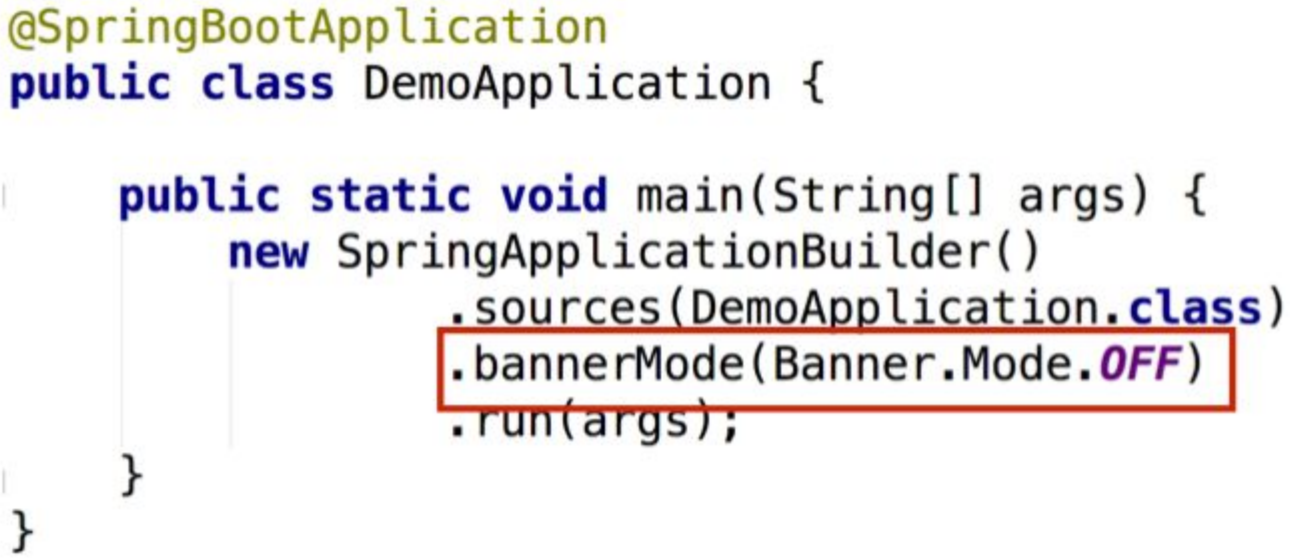
```
▼ src
  ▼ main
    ▼ java
      ▼ com.example.demo
          DemoApplication
    ▼ resources
        static
        templates
        application.properties
        banner.txt
```

Custom banner (2)



```
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        ImageBanner banner = new ImageBanner(
                new ClassPathResource("try.png"));

        new SpringApplicationBuilder()
                .sources(DemoApplication.class)
                .banner(banner)
                .run();
    }
}
```

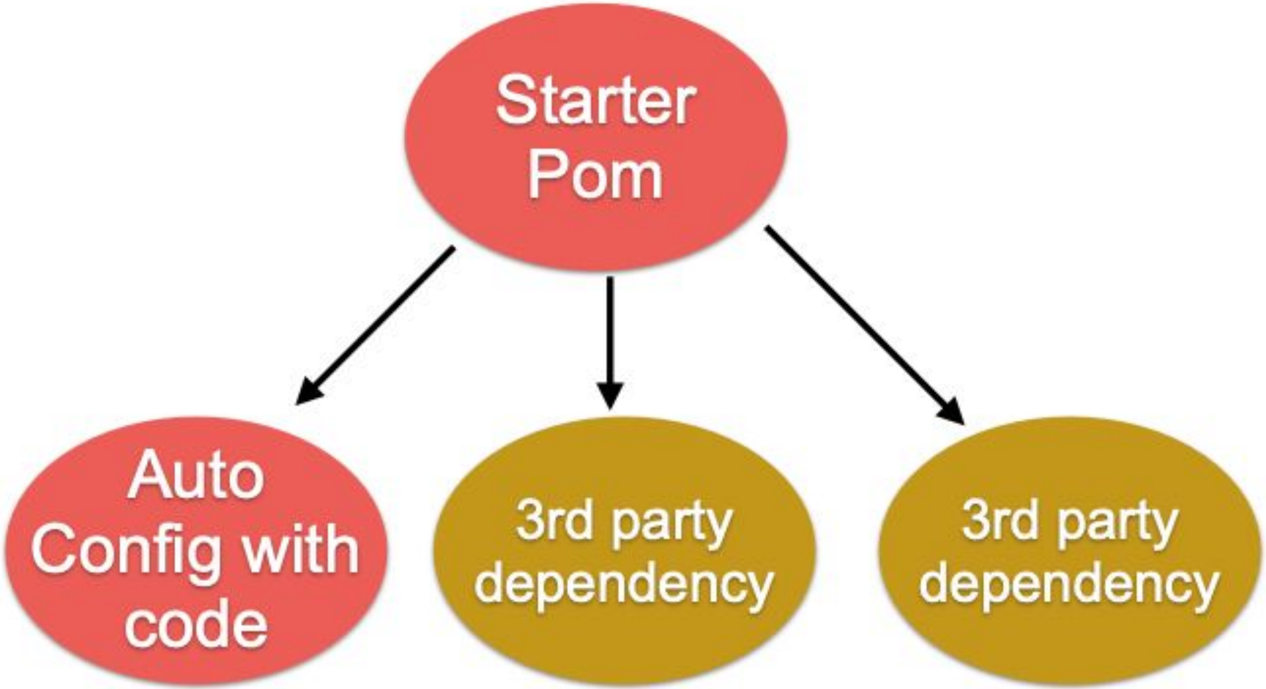
Disable banner



```
@SpringBootApplication
public class DemoApplication {

    public static void main(String[] args) {
        new SpringApplicationBuilder()
                .sources(DemoApplication.class)
                .bannerMode(Banner.Mode.OFF)
                .run(args);
    }
}
```

Anatomy of Starter

Anatomy of Starter



Starter Pom

Auto Config with code

3rd party dependency

3rd party dependency

Configuration management

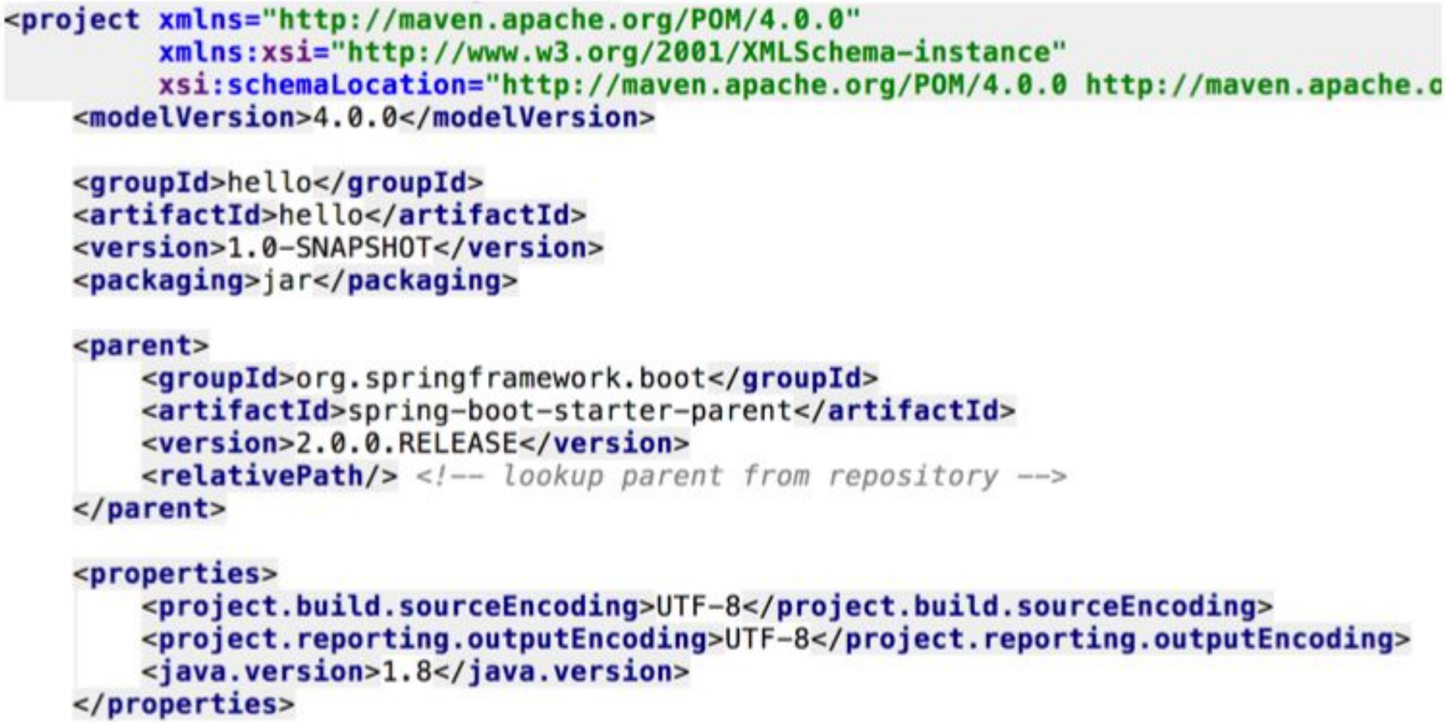
Properties/XML/YAML

Config server (App, Spring cloud config, git)

17 ways !!!

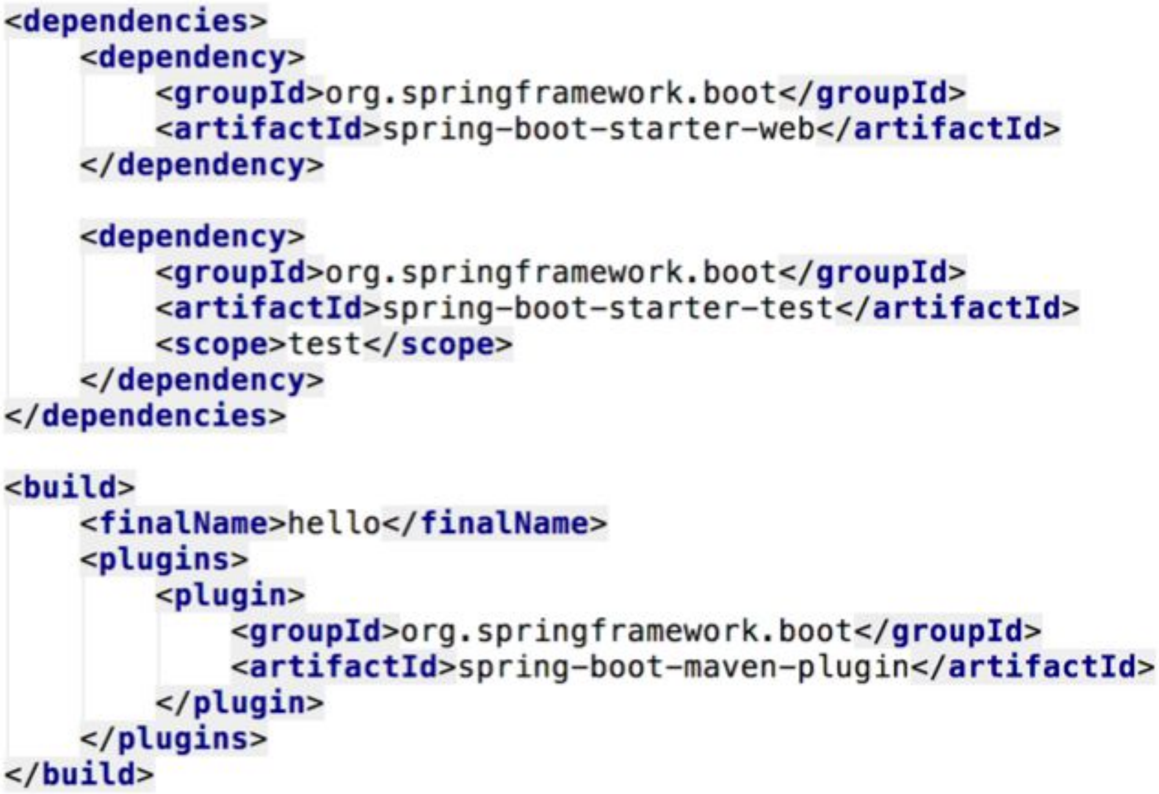
Look at POM.xml

POM.xml (1)



```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.o
    <modelVersion>4.0.0</modelVersion>

    <groupId>hello</groupId>
    <artifactId>hello</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.0.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
    </properties>
```

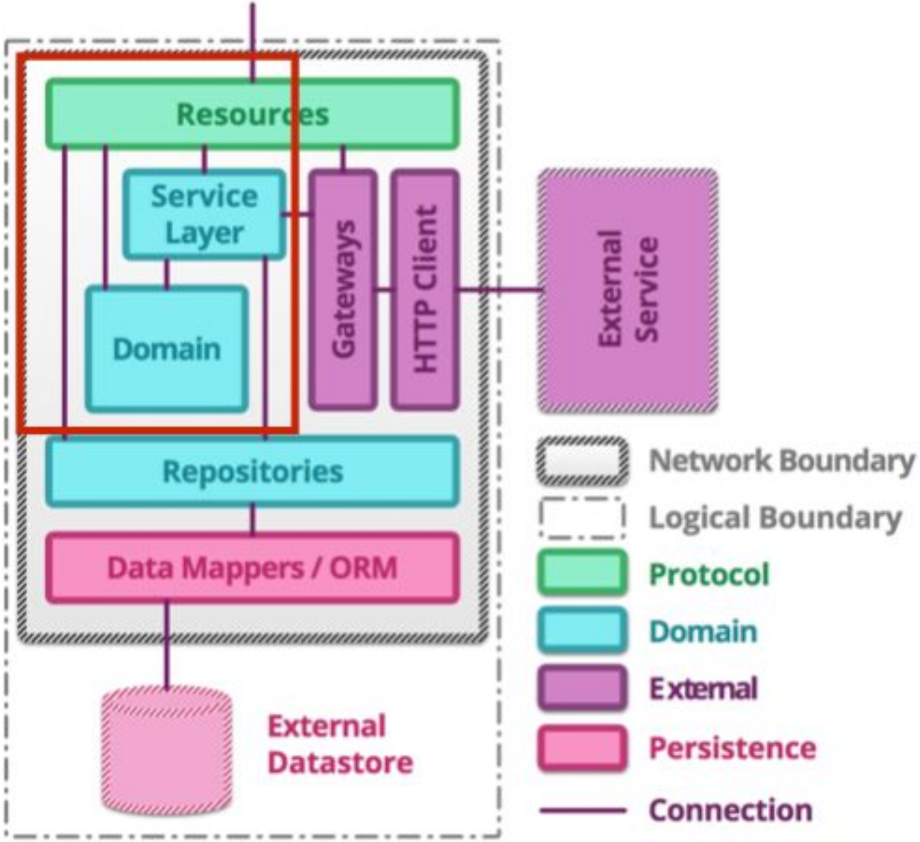
POM.xml (2)



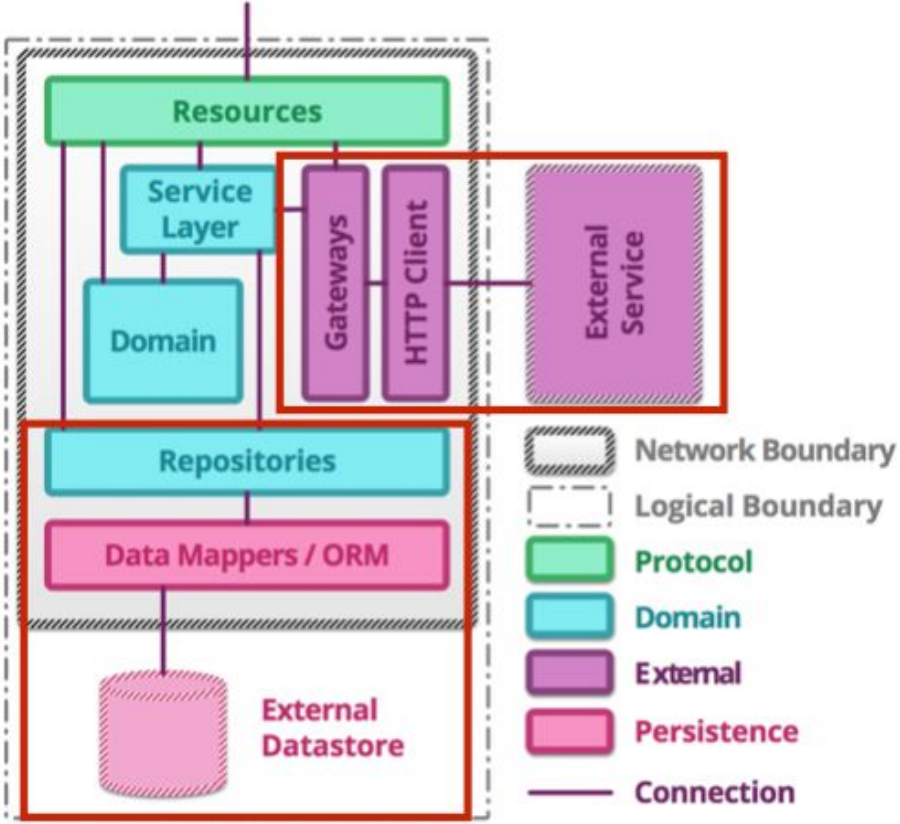
```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <finalName>hello</finalName>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Basic structure of Spring Boot

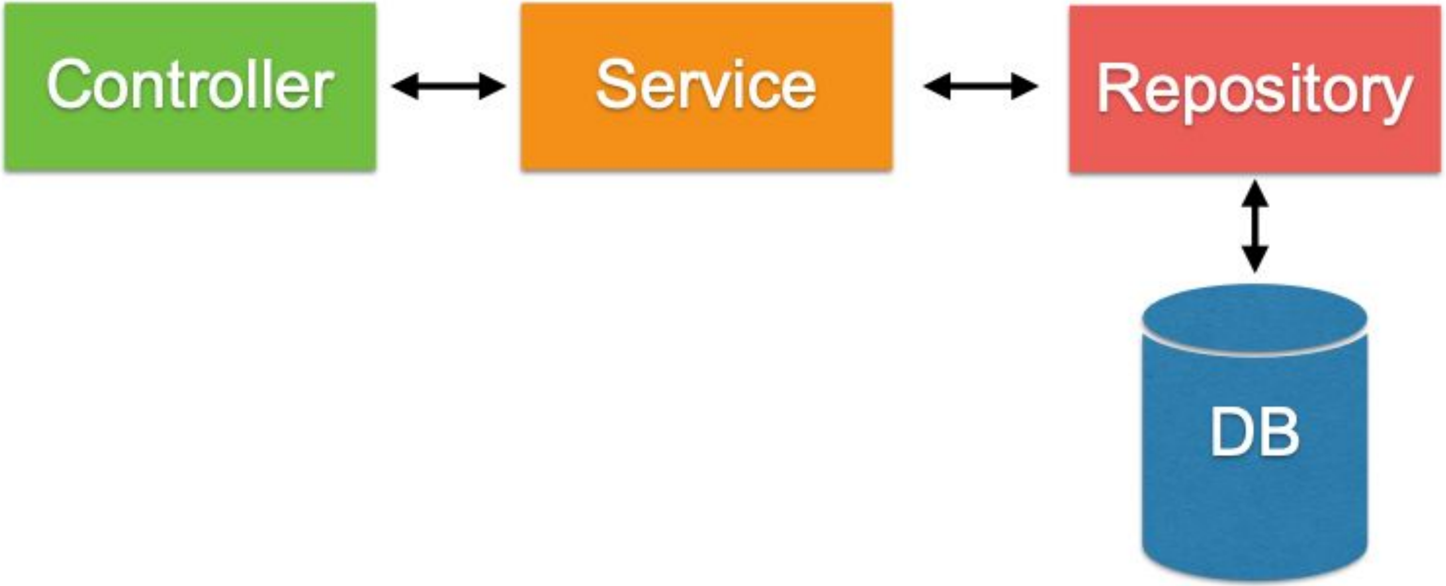
Service Structure



Service Structure



Basic structure of Spring Boot



Controller

Request and Response
Validation input
Delegate to others classes such as service and repository

Service

Control the flow of main business logics

Manage database transaction

Don’t reuse service

Repository

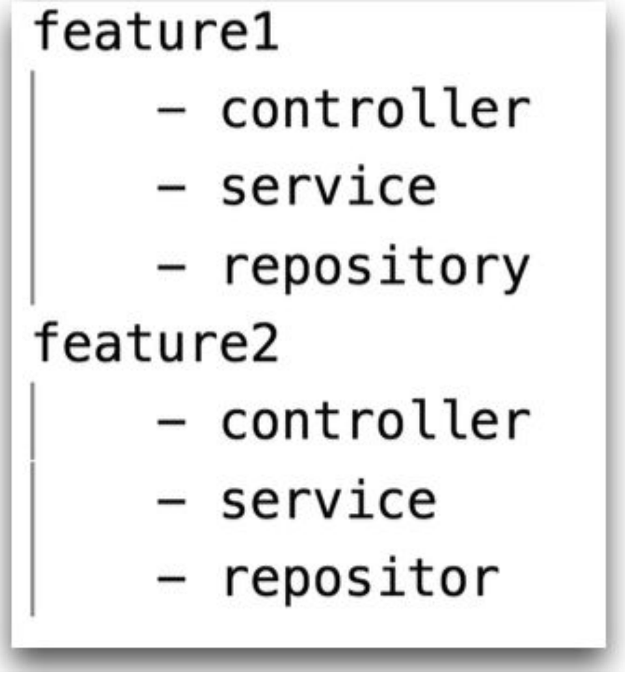
Manage data in data store such as RDBMS and NoSQL

Gateway

Call external service via network such as WebServices and RESTful APIs

Spring Boot Structure (1)

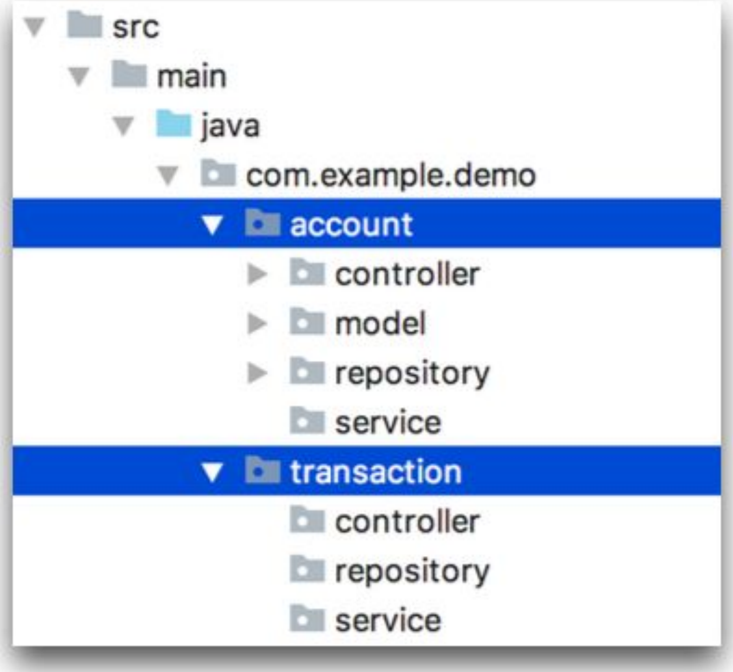
Separate by function/domain/feature



```
feature1
|     – controller
|     – service
|     – repository
feature2
|     – controller
|     – service
|     – repositor
```

Spring Boot Structure (2)

Separate by function/domain/feature



```
▼ src
    ▼ main
        ▼ java
            ▼ com.example.demo
                ▼ account
                    ▶ controller
                    ▶ model
                    ▶ repository
                      service
                ▼ transaction
                      controller
                      repository
                      service
```