Java MVC Frameworks

Spring Boot Introduction

SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Contents

Questions

Software University

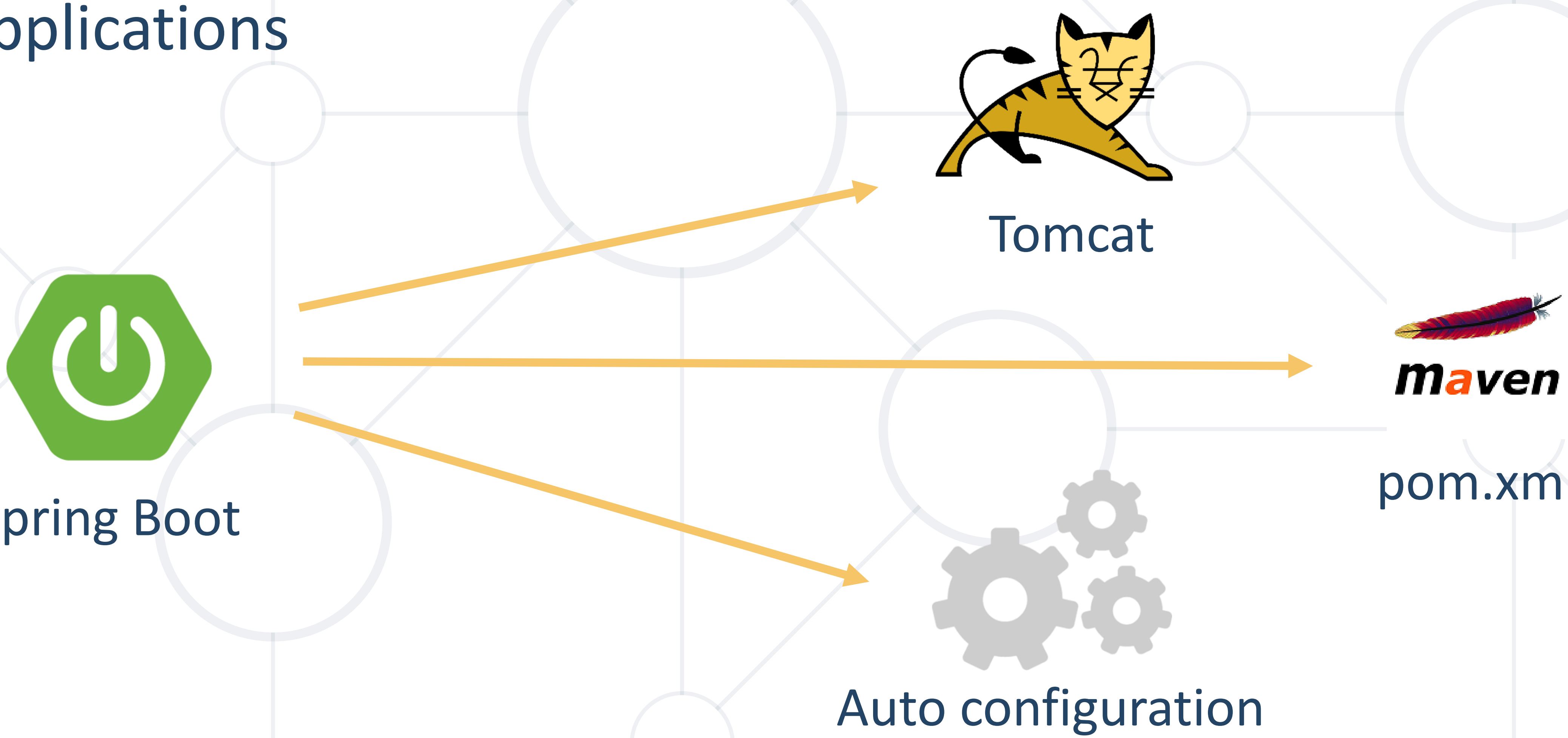
sli.do

#java-web

What is Spring Boot?

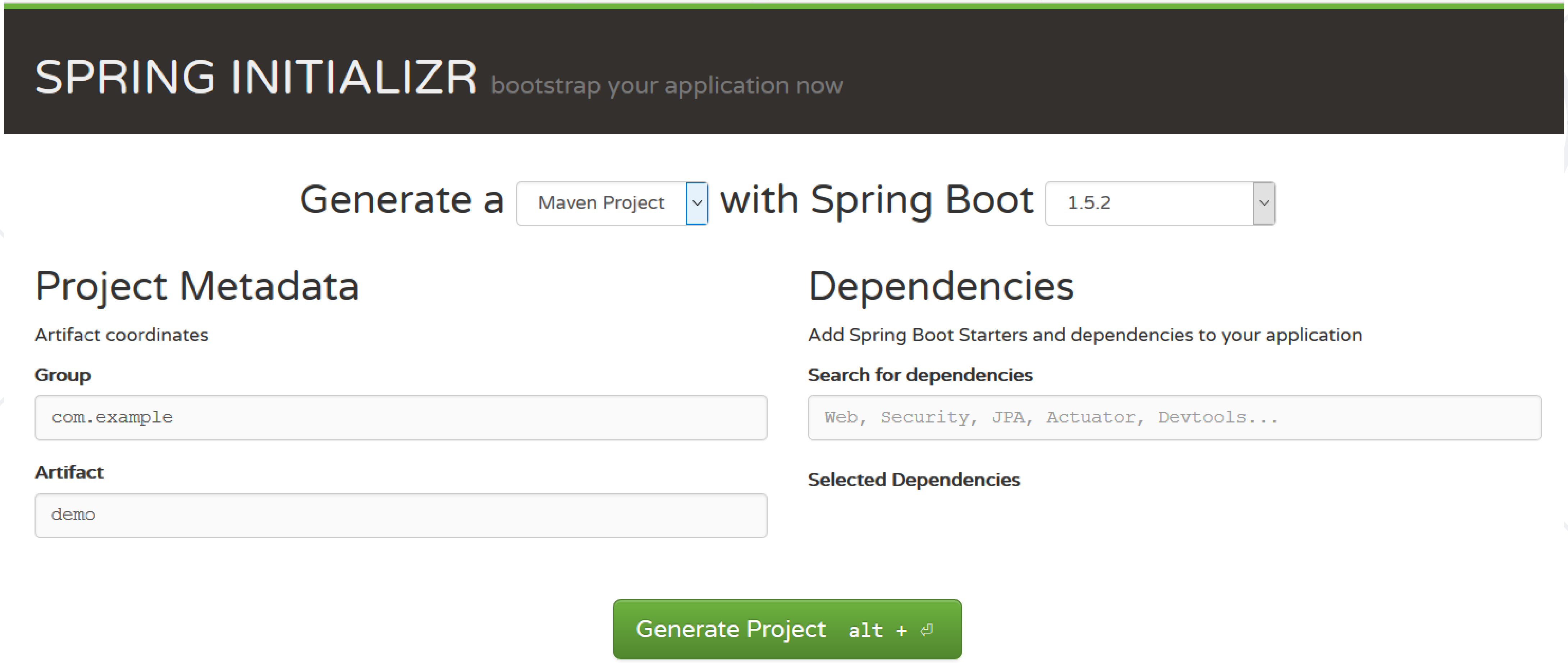
Spring Boot

- Opinionated view of building production-ready Spring applications



Creating Spring Boot Project

- Just go to https://start.spring.io/



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Spring Boot 1.5.2

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.

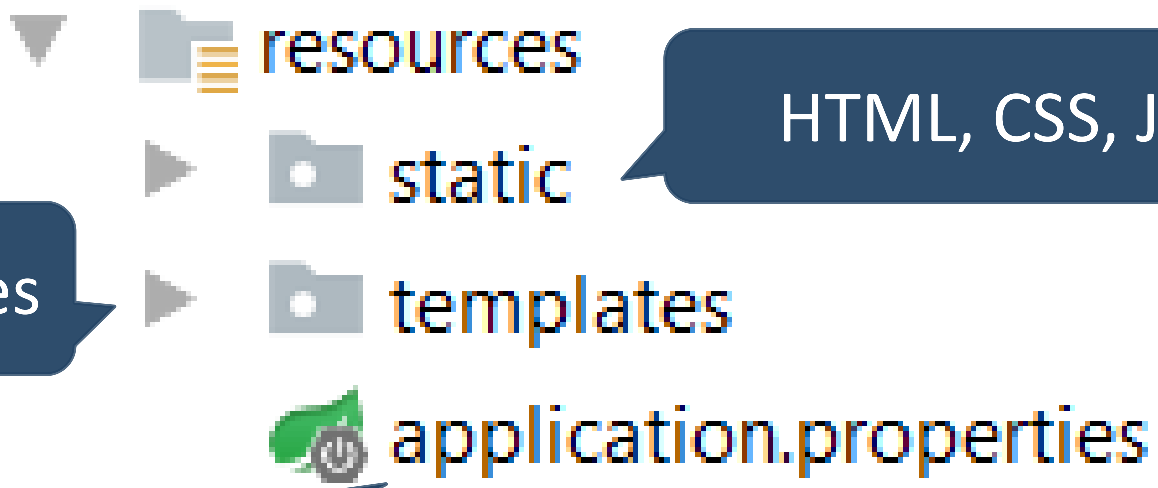
Spring Dev Tools

- Additional set of tools that can make the application development faster and more enjoyable

pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
</dependency>
```

Spring Resources



HTML, CSS, JS

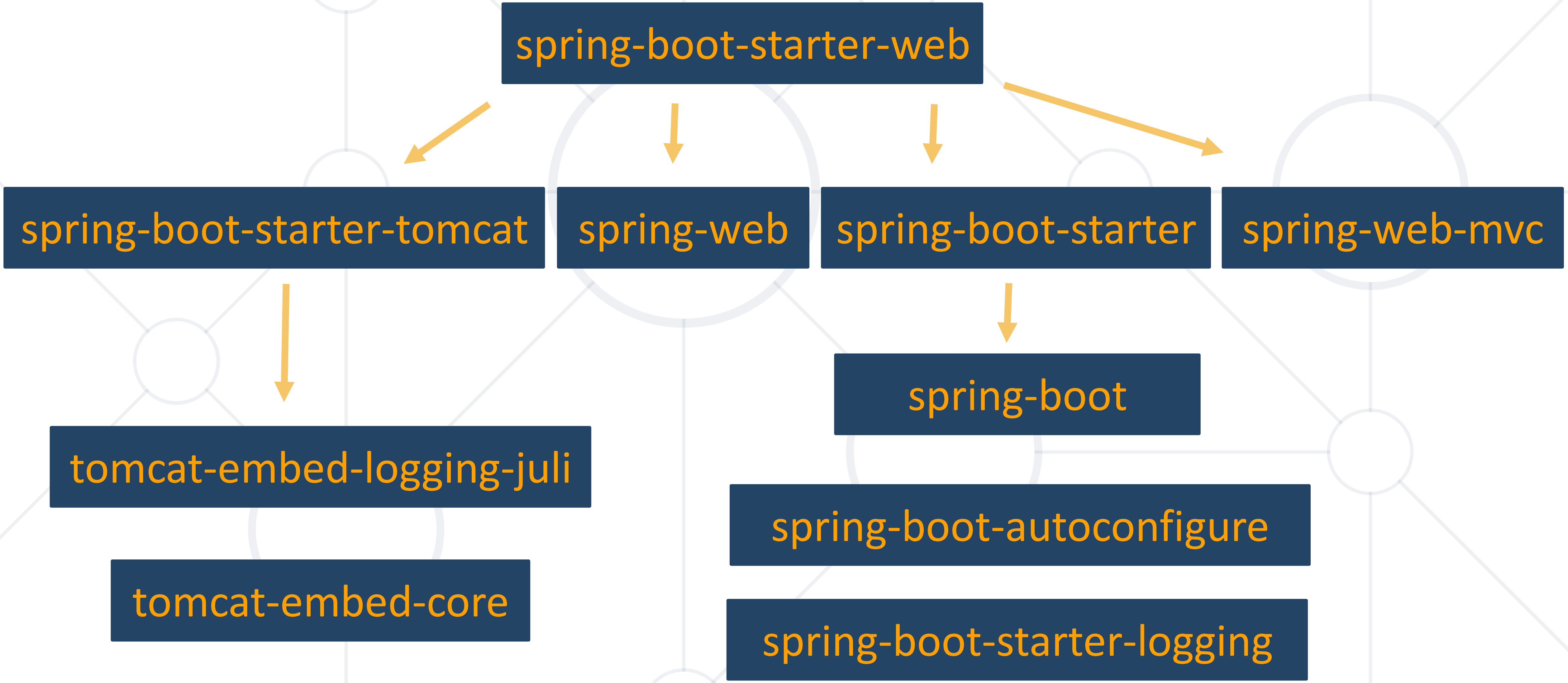
Thymeleaf templates

Application properties

Spring Boot Main Components

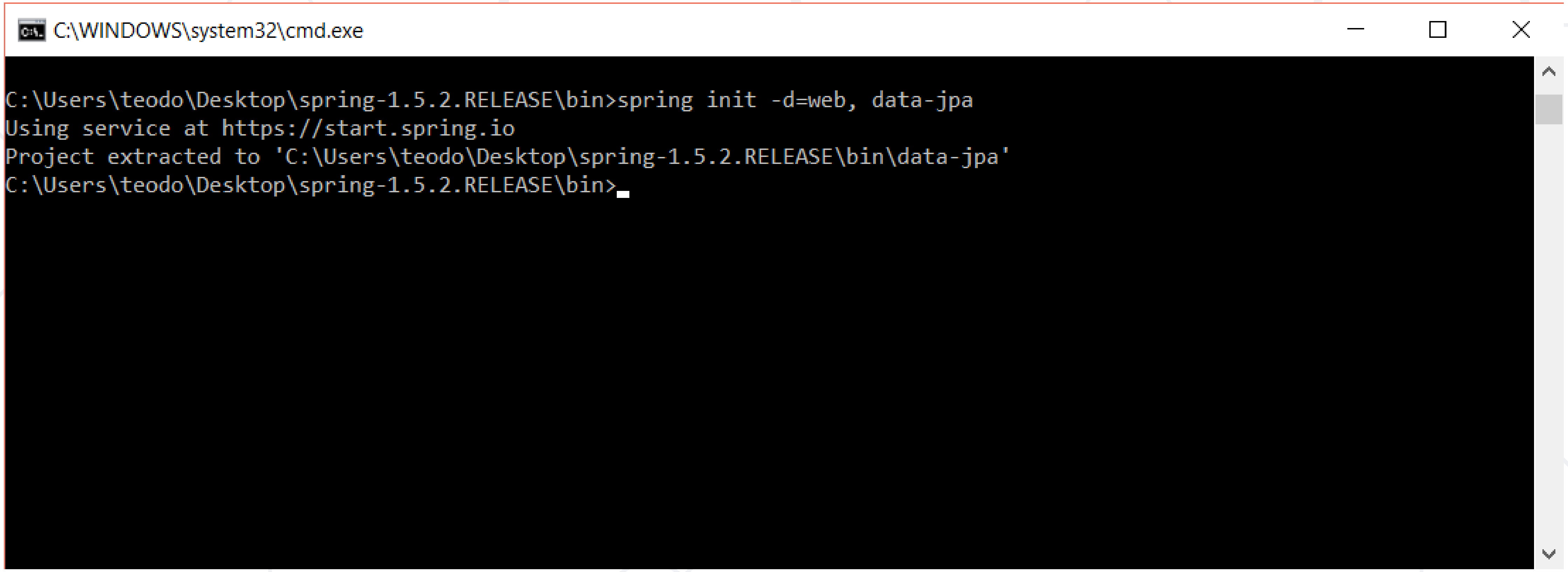
- Four main components:
  - Spring Boot Starters - combine a group of common or related dependencies into single dependency
  - Spring Boot Auto-Configuration - reduce the Spring Configuration
  - Spring Boot CLI - run and test Spring Boot applications from command prompt
  - Spring Boot Actuator – provides EndPoints and Metrics

Spring Boot Starters


spring-boot-starter-web
spring-boot-starter-tomcat
spring-web
spring-boot-starter
spring-web-mvc
tomcat-embed-logging-juli
tomcat-embed-core
spring-boot
spring-boot-autoconfigure
spring-boot-starter-logging

Spring Boot CLI

- Command Line Interface - Spring Boot software to run and test Spring Boot applications



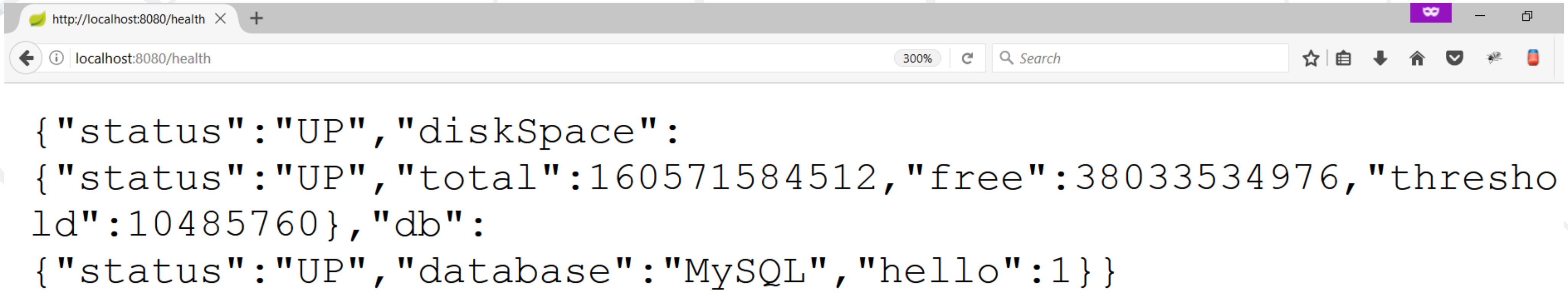
```
C:\Users\teodo\Desktop\spring-1.5.2.RELEASE\bin>spring init -d=web, data-jpa
Using service at https://start.spring.io
Project extracted to 'C:\Users\teodo\Desktop\spring-1.5.2.RELEASE\bin\data-jpa'
C:\Users\teodo\Desktop\spring-1.5.2.RELEASE\bin>
```

Spring Boot Actuator

- Expose different types of information about the running application

pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```



```
{"status":"UP","diskSpace":
{"status":"UP","total":160571584512,"free":38033534976,"thresho
ld":10485760},"db":
{"status":"UP","database":"MySQL","hello":1}}
```

Inversion of Control

- Spring provides Inversion of Control and Dependency Injection

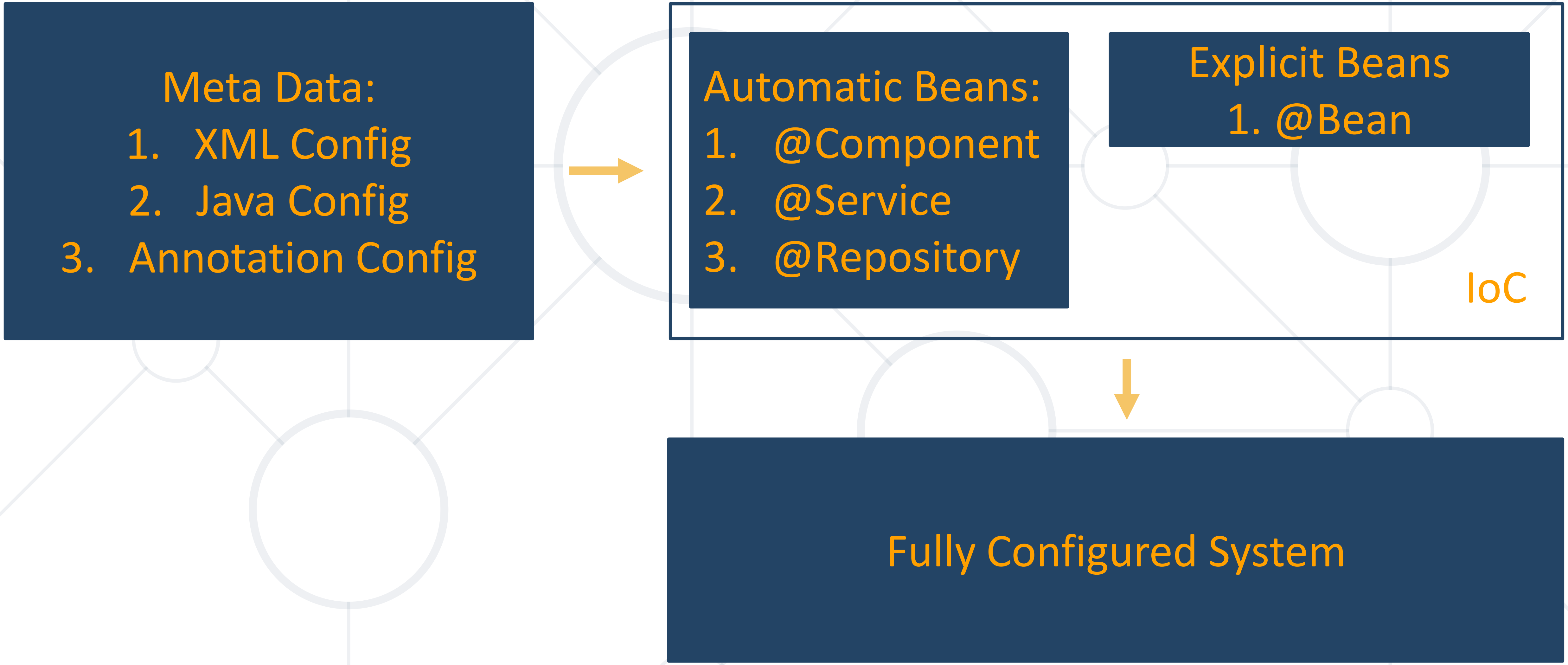
UserServiceImpl.java

```
//Tradiotional Way
public class UserServiceImpl implements
UserService {

private UserRepository userRepository = new
UserRepository();

}
```

UserServiceImpl.java

```
//Dependency Injection
@Service
public class UserServiceImpl implements
UserService {

@Autowired
private UserRepository userRepository;
}
```

Spring IoC


Meta Data:
1. XML Config
2. Java Config
3. Annotation Config

Automatic Beans:
1. @Component
2. @Service
3. @Repository

Explicit Beans
1. @Bean

IoC

Fully Configured System

Beans

- Object that is instantiated, assembled, and otherwise managed by a Spring IoC container

Dog.java

```
public class Dog implements Animal {

    private String name;

    public Dog() {}

    //GETTERS AND SETTERS
}
```

Bean Declaration

Dog.java

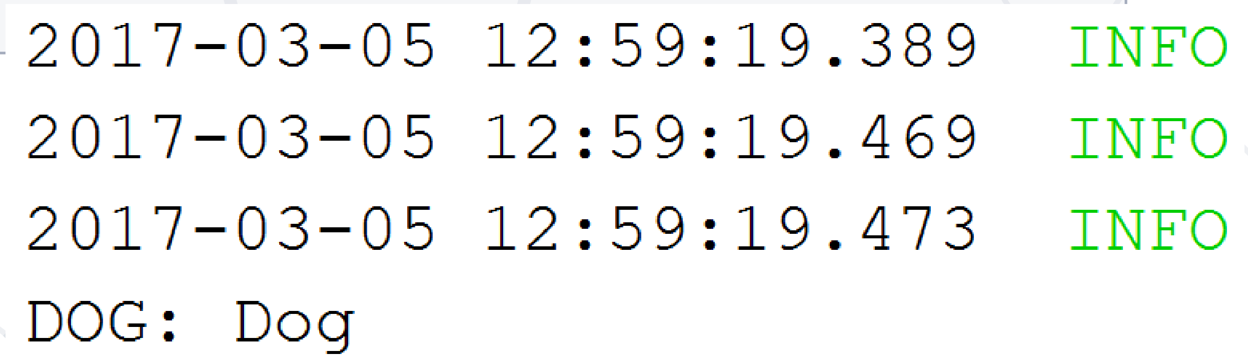
```
@SpringBootApplication
public class MainApplication {

    …

    @Bean
    public Animal getDog(){
        return new Dog();
    }
}
```

Bean Declaration

Get Bean from Application Context

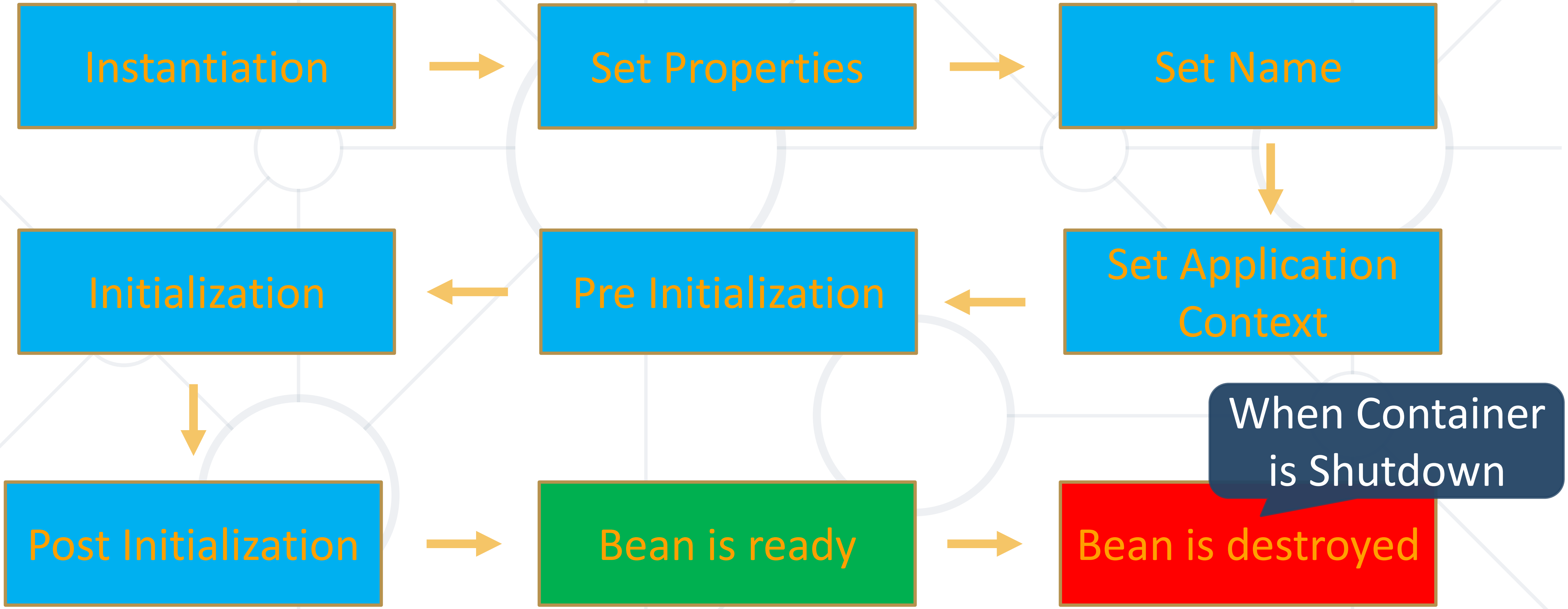
MainApplication.java

```java
@SpringBootApplication
public class MainApplication {

  public static void main(String[] args) {
    ApplicationContext context = SpringApplication.run(MainApplication.class,
args);
    Animal dog = context.getBean(Dog.class);
    System.out.println("DOG: " + dog.getClass().getSimpleName());
    }
}
```



```
2017-03-05 12:59:19.389  INFO
2017-03-05 12:59:19.469  INFO
2017-03-05 12:59:19.473  INFO
DOG: Dog
```

Bean Lifecycle



Bean Lifecycle Demo (1)

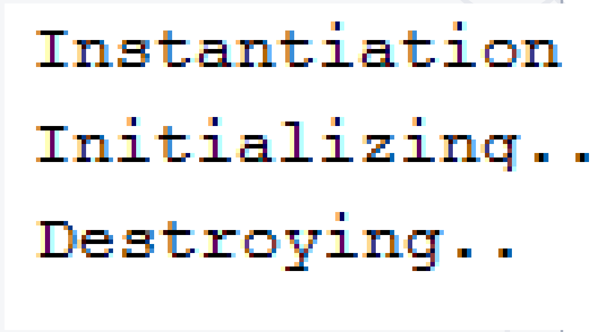
MainApplication.java

```
@SpringBootApplication
public class MainApplication {

 public static void main(String[] args) {
        ApplicationContext context =
SpringApplication.run(MainApplication.class, args);
        ((AbstractApplicationContext)context).close();
 }
 @Bean(destroyMethod = "destroy", initMethod = "init")
 public Animal getDog(){
     return new Dog();
 }
}
```

Bean Lifecycle Demo (2)

MainApplication.java

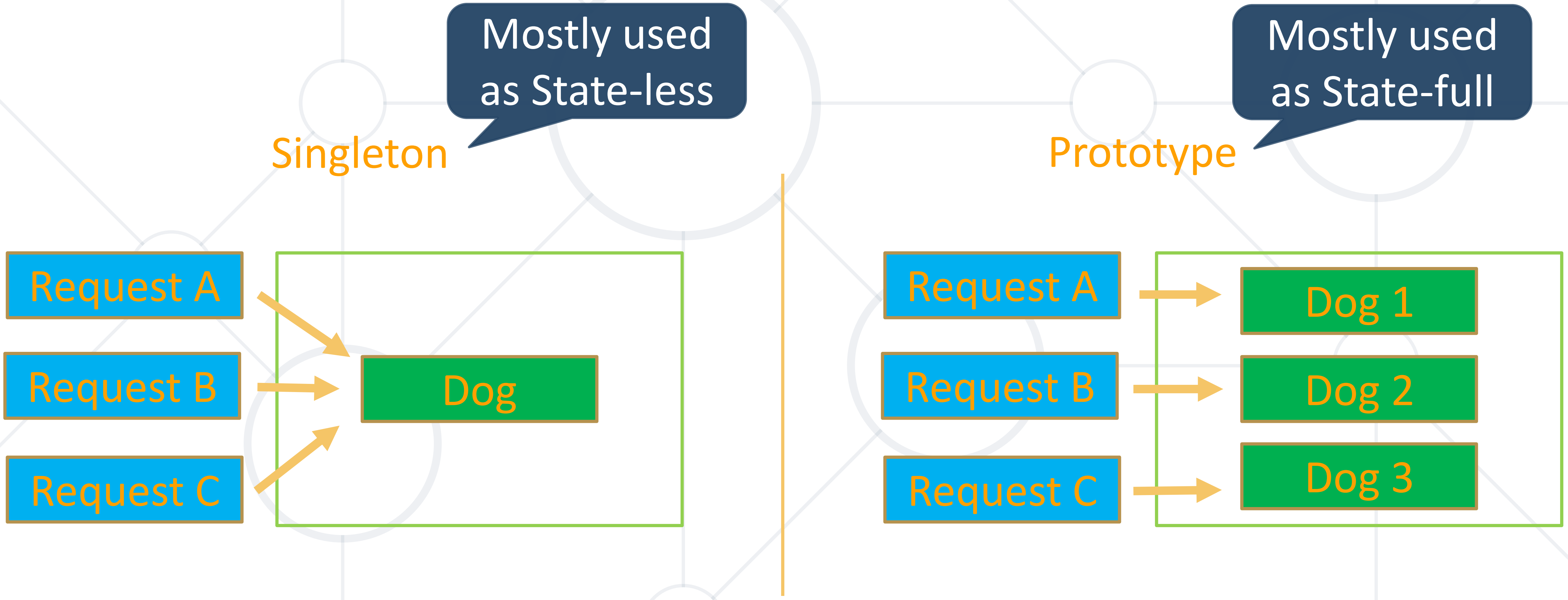
```java
public class Dog implements Animal {

    public Dog() {
        System.out.println("Instantiation");
    }

    public void init(){
        System.out.println("Initializing..");
    }

    public void destroy(){
        System.out.println("Destroying..");
    }
}
```



```
Instantiation
Initializing..
Destroying..
```

Bean Scope

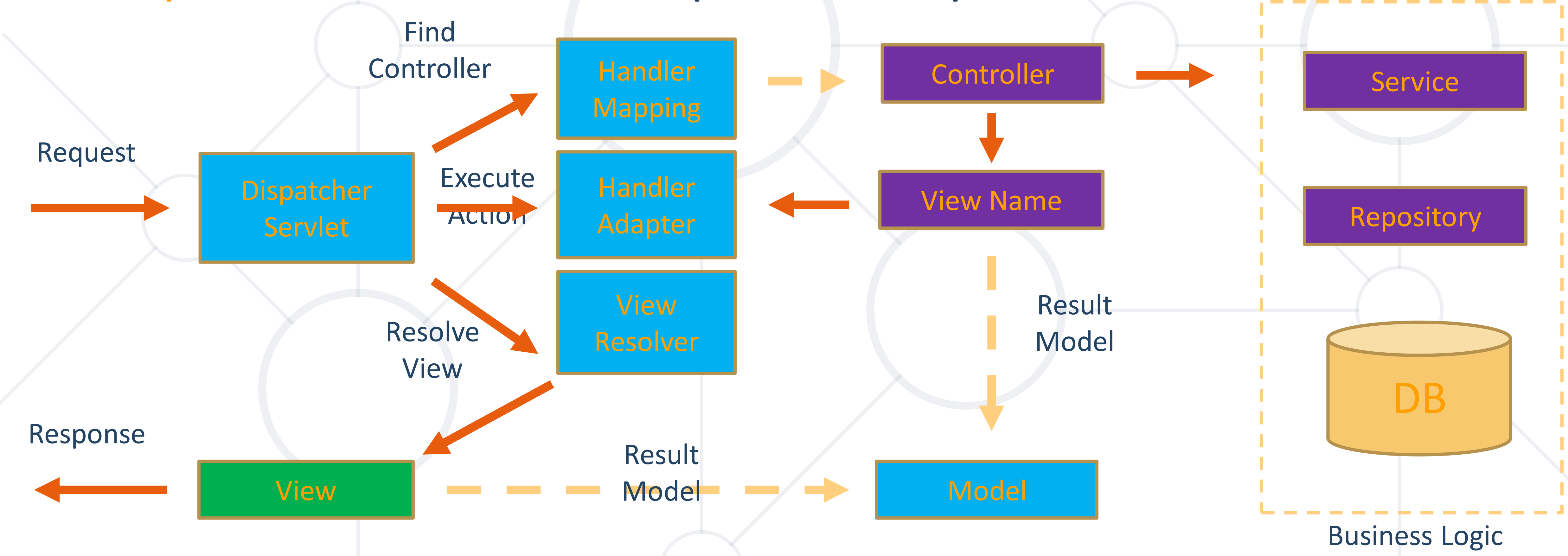
- The default one is Singleton. It is easy to change to Prototype



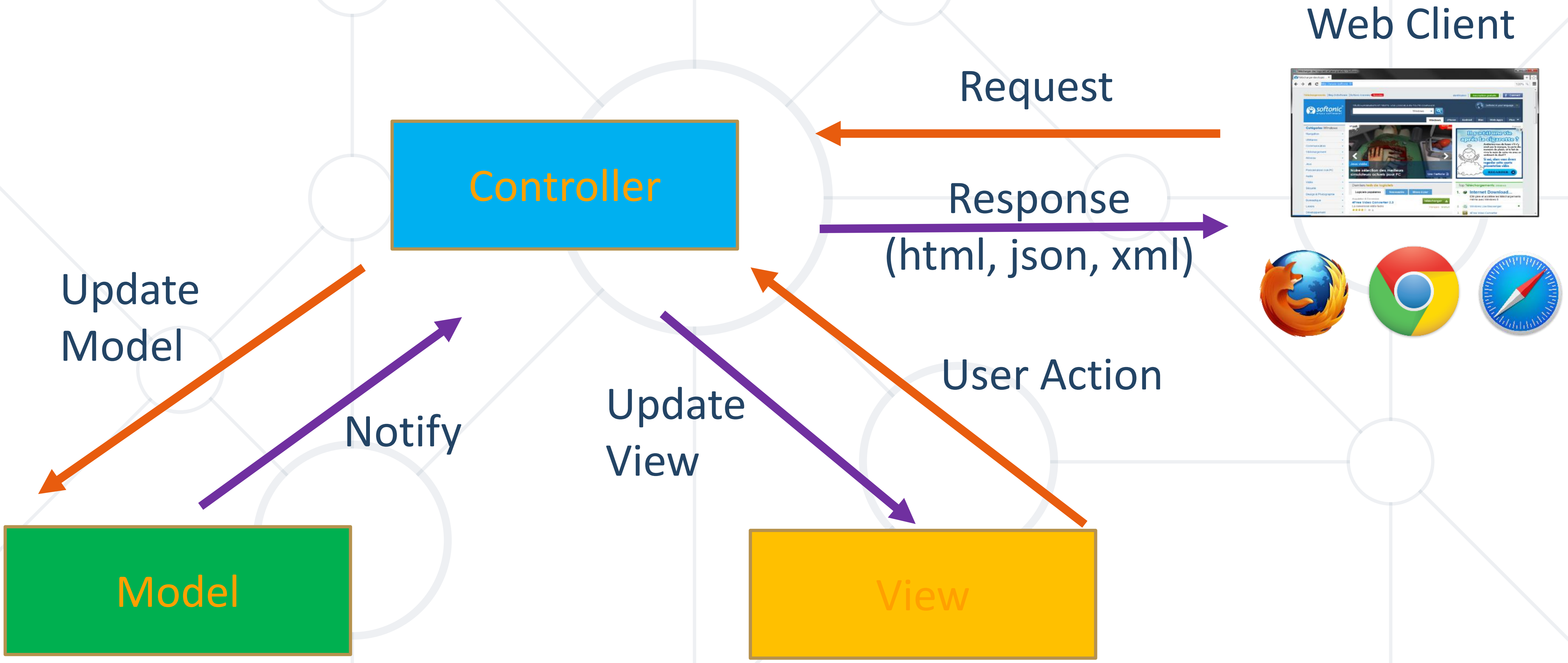
What is Spring MVC?

What is Spring MVC?

- Model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers



MVC – Control Flow



Controllers

Controller

DogController.java

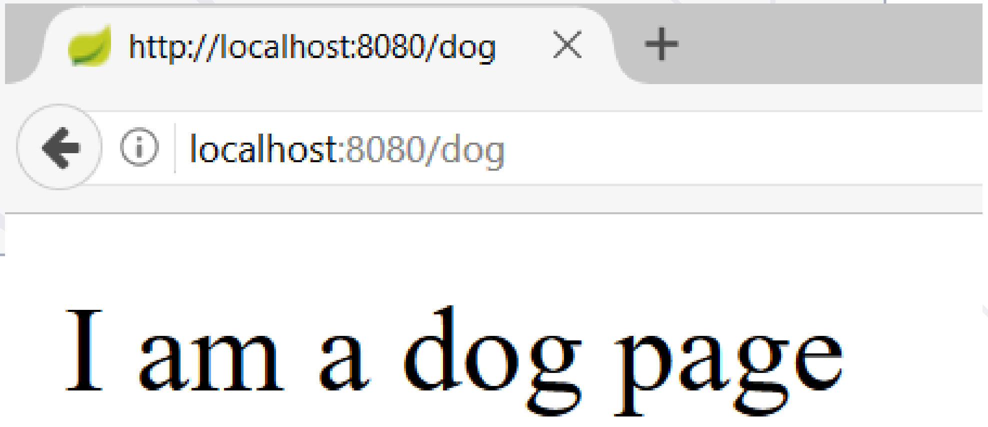
```
@Controller
public class DogController {

    @GetMapping("/dog")
    @ResponseBody
    public String getDogHomePage(){
        return "I am a dog page";
    }
}
```

Request Mapping

Action

Print Text

Text



I am a dog page

Actions – Get Requests

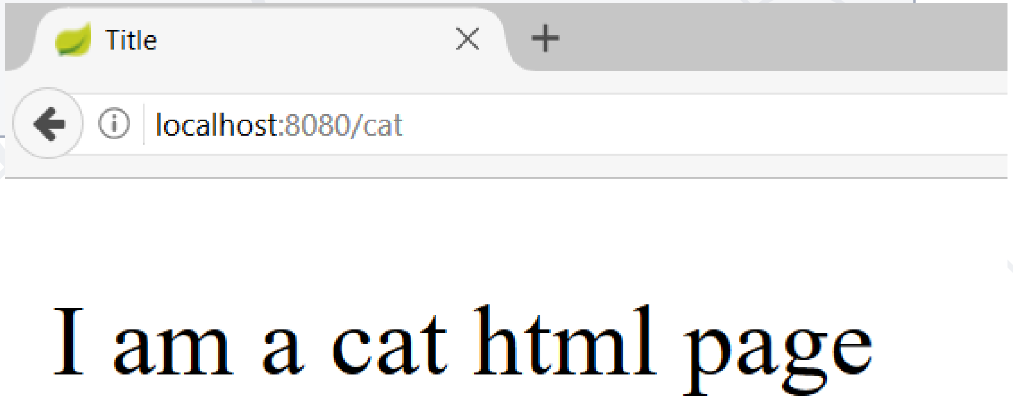
CatController.java

```
@Controller
public class CatController {

    @GetMapping("/cat")
    public String getHomeCatPage(){
        return "cat-page.html";
    }
}
```

Request Mapping

Action

View



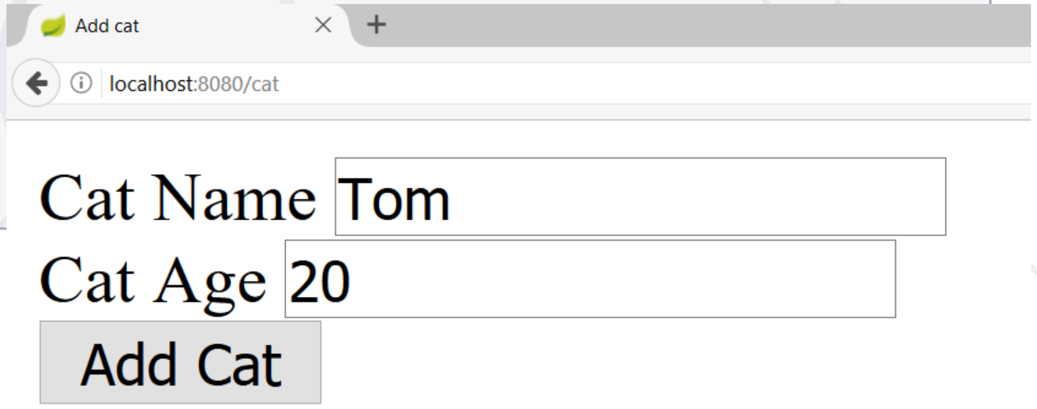
I am a cat html page

Actions – Post Requests (1)

CatController.java

```
@Controller
@RequestMapping("/cat")
public class CatController {

    @GetMapping("")
    public String getHomeCatPage(){
        return "new-cat.html";
    }
}
```

Starting route



Add cat

localhost:8080/cat

Cat Name	Tom
Cat Age	20

Actions – Post Requests (1)

CatController.java

```
@Controller
@RequestMapping("/cat")
public class CatController {

     @PostMapping("")
    public String addCat(@RequestParam String catName,
@RequestParam int catAge){
        System.out.println(String.format("Cat Name: %s, Cat
Age: %d", catName, catAge));
        return "redirect:/cat";
    }
}
```

Request param

Redirect

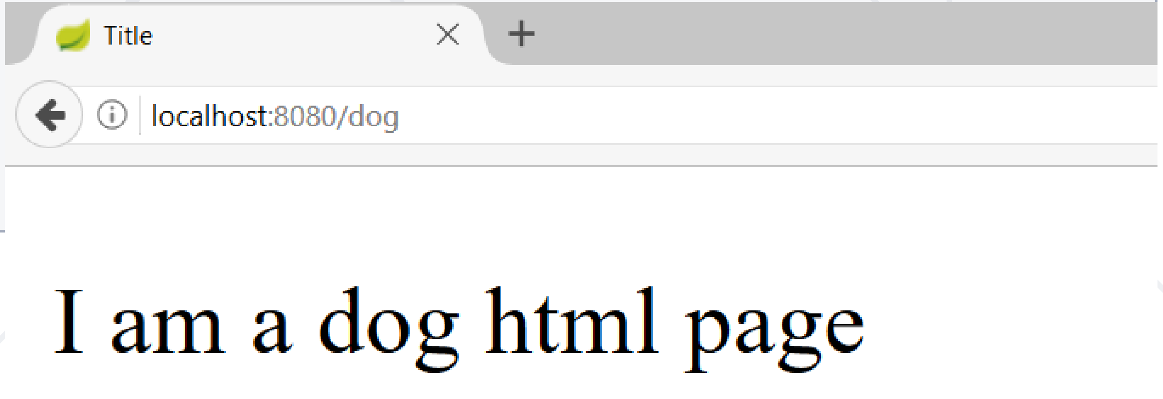
```
Cat Name: Tom, Cat Age: 20
```

Models and Views

DogController.java

```
@Controller
public class DogController {

    @GetMapping("/dog")
    public ModelAndView getDogHomePage(ModelAndView modelAndView){
        modelAndView.setViewName("dog-page.html");
        return modelAndView;
    }
}
```

Model and View



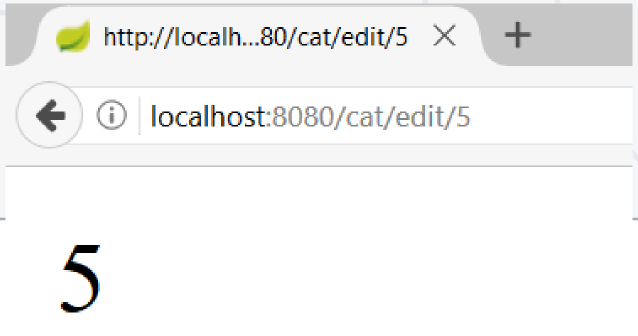
I am a dog html page

Path Variables

CatController.java

```
@Controller
@RequestMapping("/cat")
public class CatController {

    @GetMapping("/edit/{catId}")
    @ResponseBody
    public String editCat(@PathVariable long catId){
        return String.valueOf(catId);
    }
}
```

Path Variable

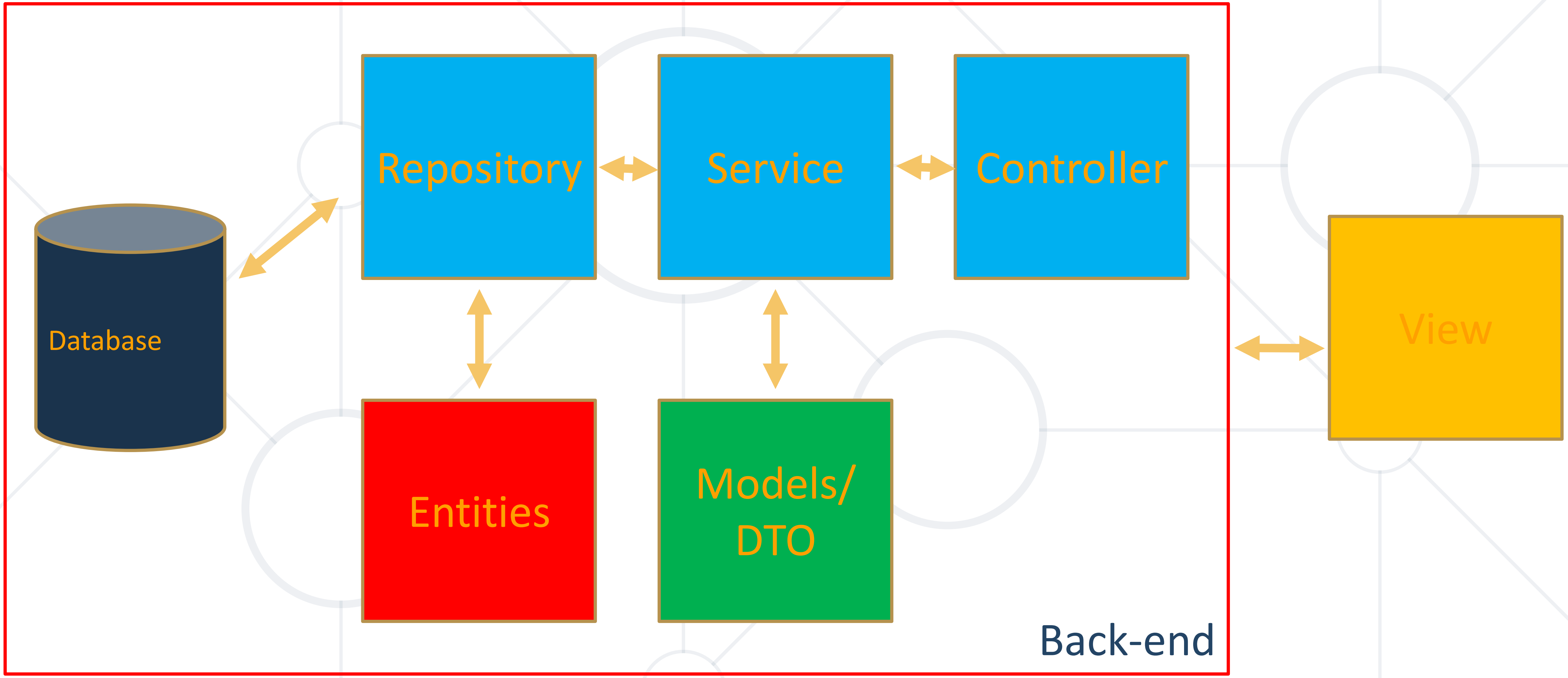


localhost:8080/cat/edit/5

5

Spring Data

Overall Architecture



Application Properties

application.properties

```
#Data Source Properties
spring.datasource.driverClassName=com.mysql.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/cat_store?useSSL=
false&createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=1234
#JPA Properties
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL
5InnoDBDialect
spring.jpa.properties.hibernate.format_sql=TRUE
spring.jpa.hibernate.ddl-auto=update
```

Entities

- Entity is a lightweight persistence domain object

Cat.java

```java
@Entity
@Table(name = "cats")
public class Cat {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;

    private String name;
    //GETTERS AND SETTERS
}
```

Repositories

- Persistence layer that works with entities

CatRepository.java

```
@Repository
public interface CatRepository extends CrudRepository<Cat, Long> {
}
```

Services

- Business Layer. All the business logic is here.

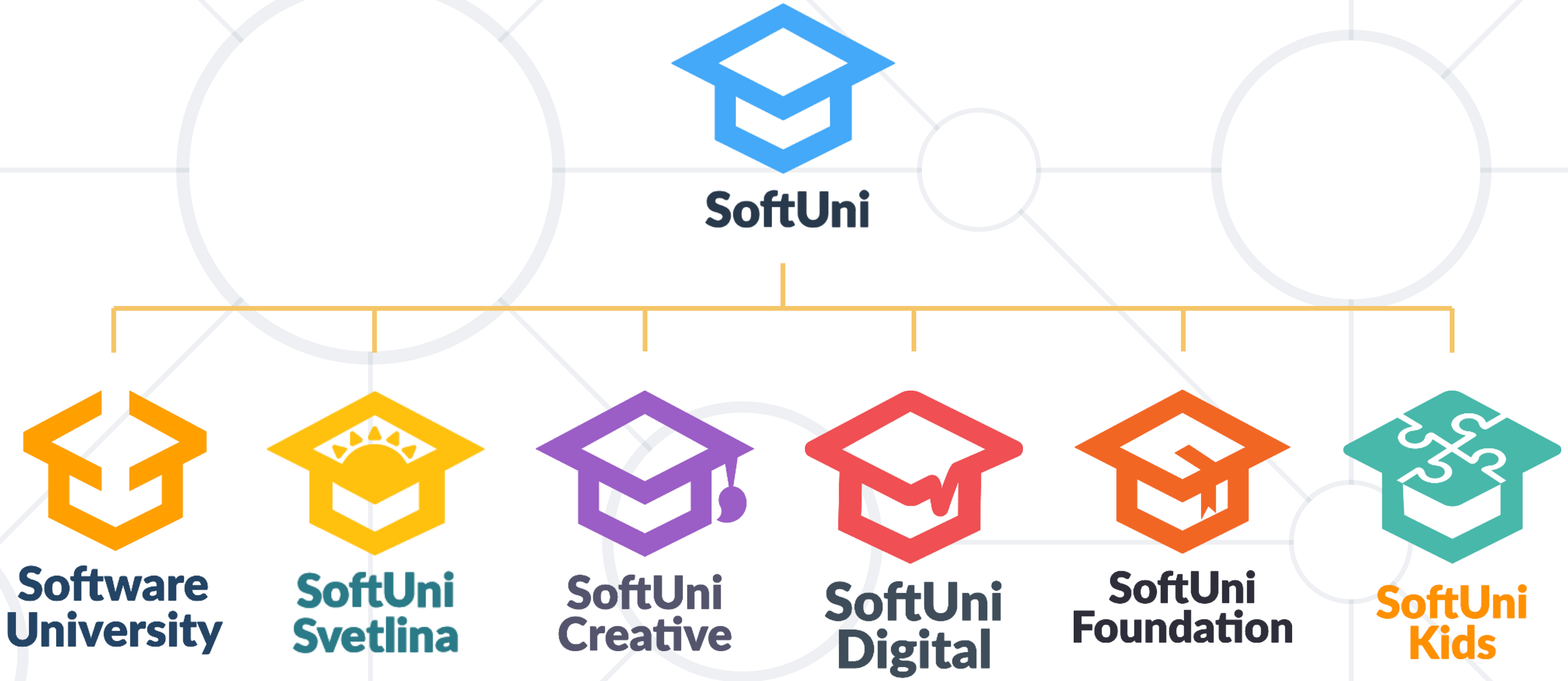
CatService.java

```
@Service
public class CatServiceImpl implements CatService {

    @Autowired
    private CatRepository catRepository;

    @Override
    public void buyCat(CatModel catModel) {
        //TODO Implement the method
    }
}
```

Summary

- Spring Boot - Opinionated view of building production-ready Spring applications
- Spring MVC - MVC framework that has three main components:
  - Controller - controls the application flow
  - View - presentation layer
  - Model - data component with the main logic
- Spring Data - Responsible for database related operations

Questions?


SoftUni
Software University
SoftUni Svetlina
SoftUni Creative
SoftUni Digital
SoftUni Foundation
SoftUni Kids

SoftUni Diamond Partners

Software University

XSsoftware

SBTech we know sports

telenor

SoftwareGroup doing it right

NETPEAK

SmartIT

Postbank Решения за твоето утре

SUPER HOSTING .BG

INDEAVR Serving the high achievers

INFRAGISTICS

LIEBHERR

æternity

SoftUni Organizational Partners

Software University

IS ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License