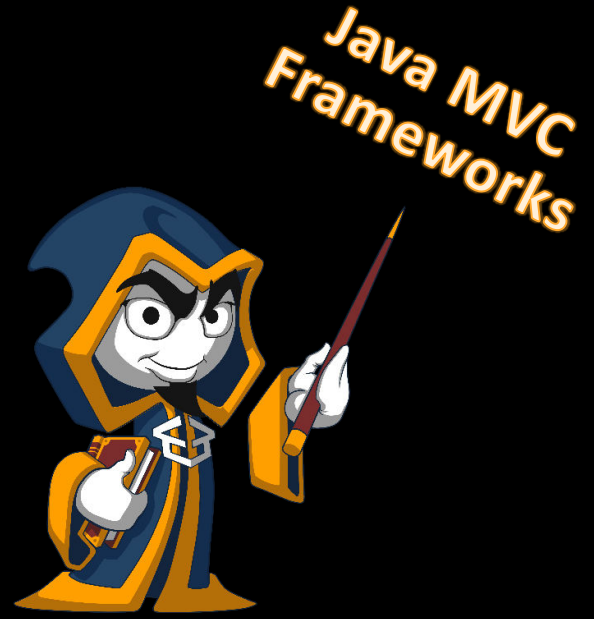
Java MVC Frameworks

Spring Boot Introduction

SoftUni Foundation



spring boot

SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Contents

Questions

SoftUni Foundation

sli.do

#java-web

spring

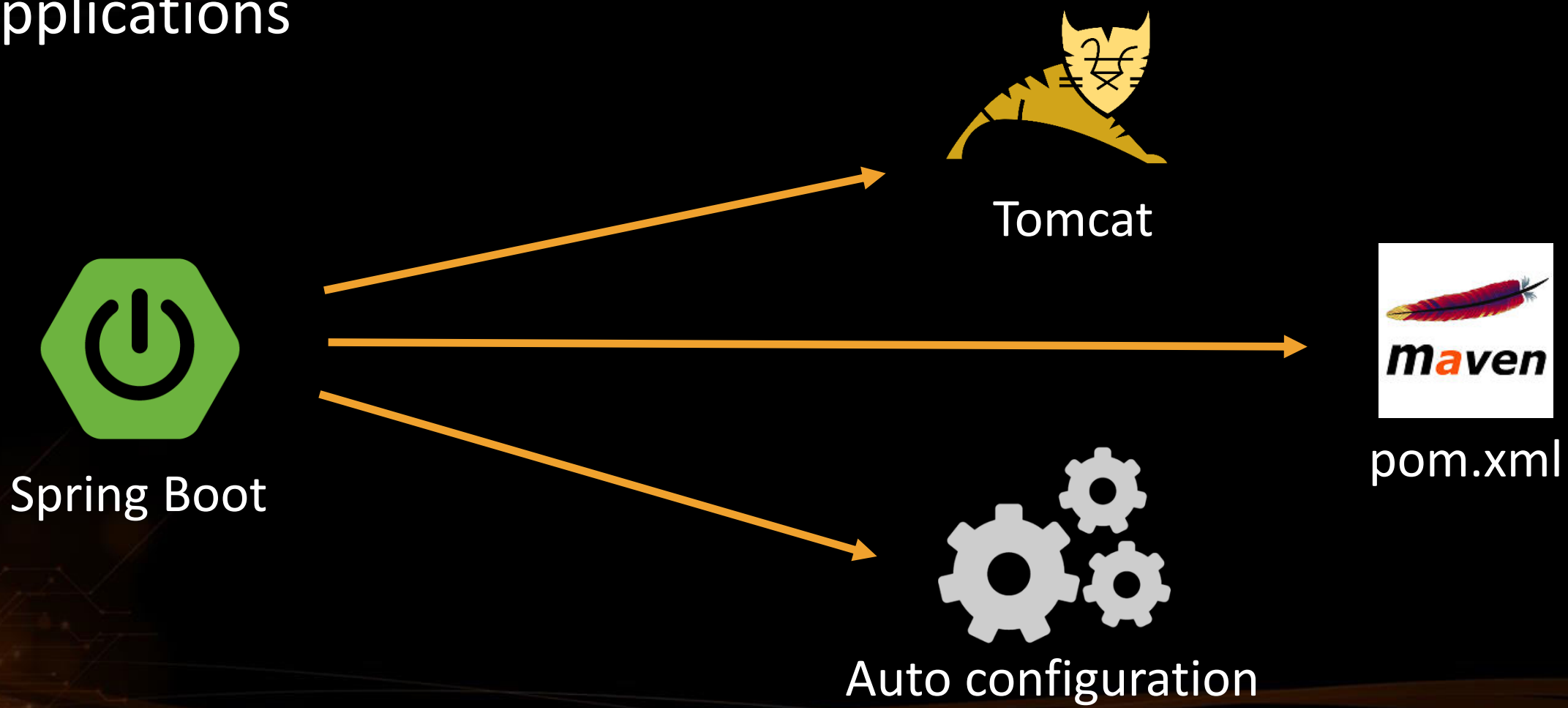
by Pivotal™

What is Spring Boot?

Spring Boot

SoftUni Foundation

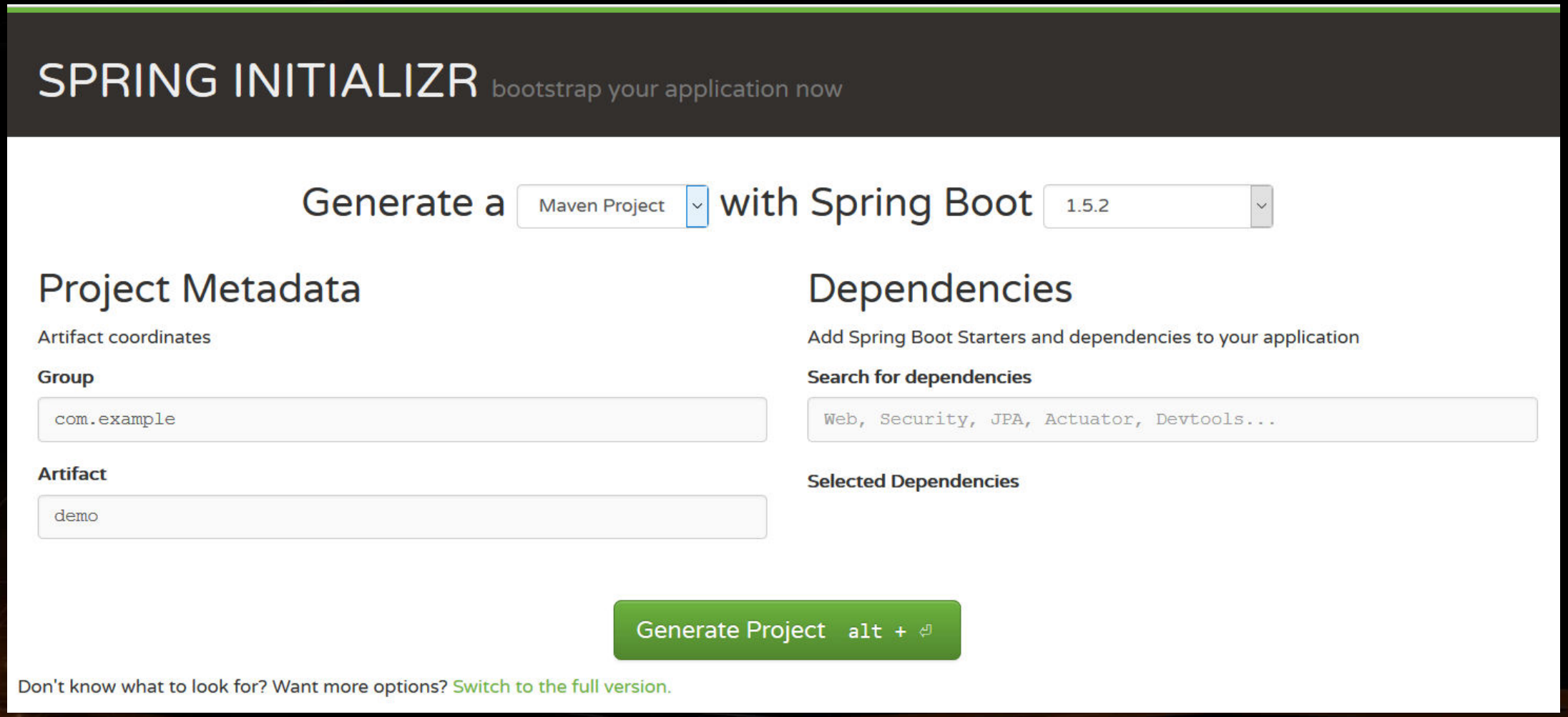
- Opinionated view of building production-ready Spring applications



Creating Spring Boot Project

SoftUni Foundation

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



Spring Dev Tools

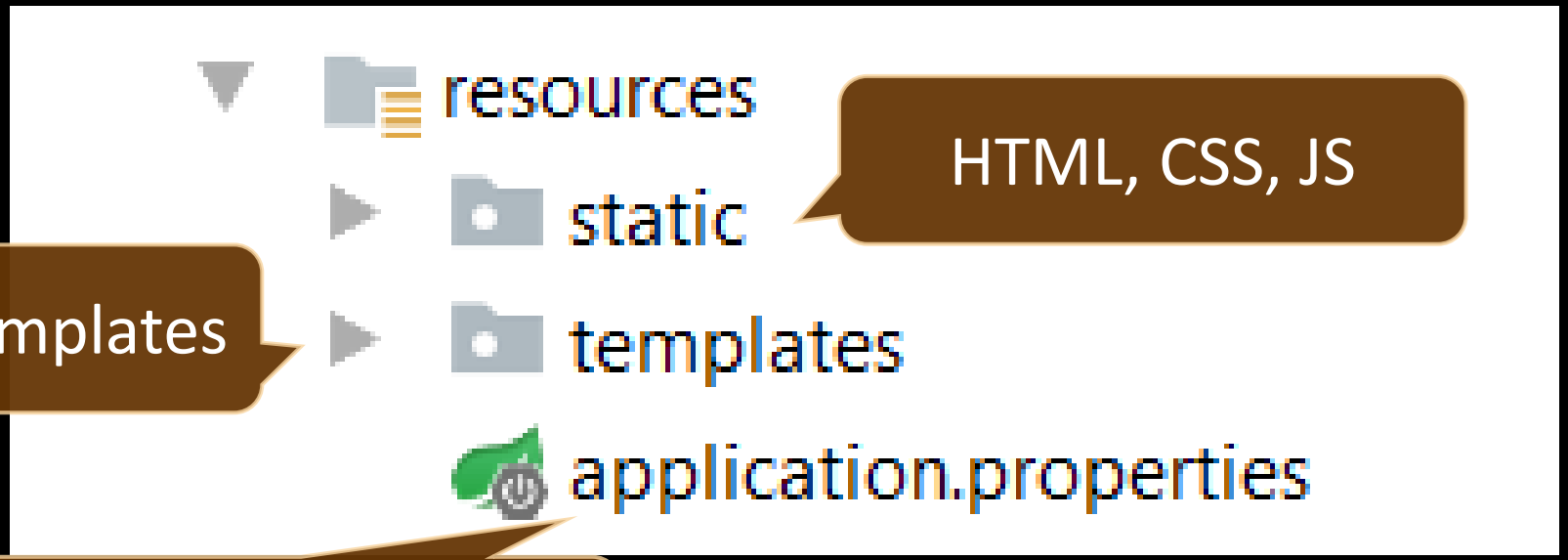
SoftUni Foundation

- 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

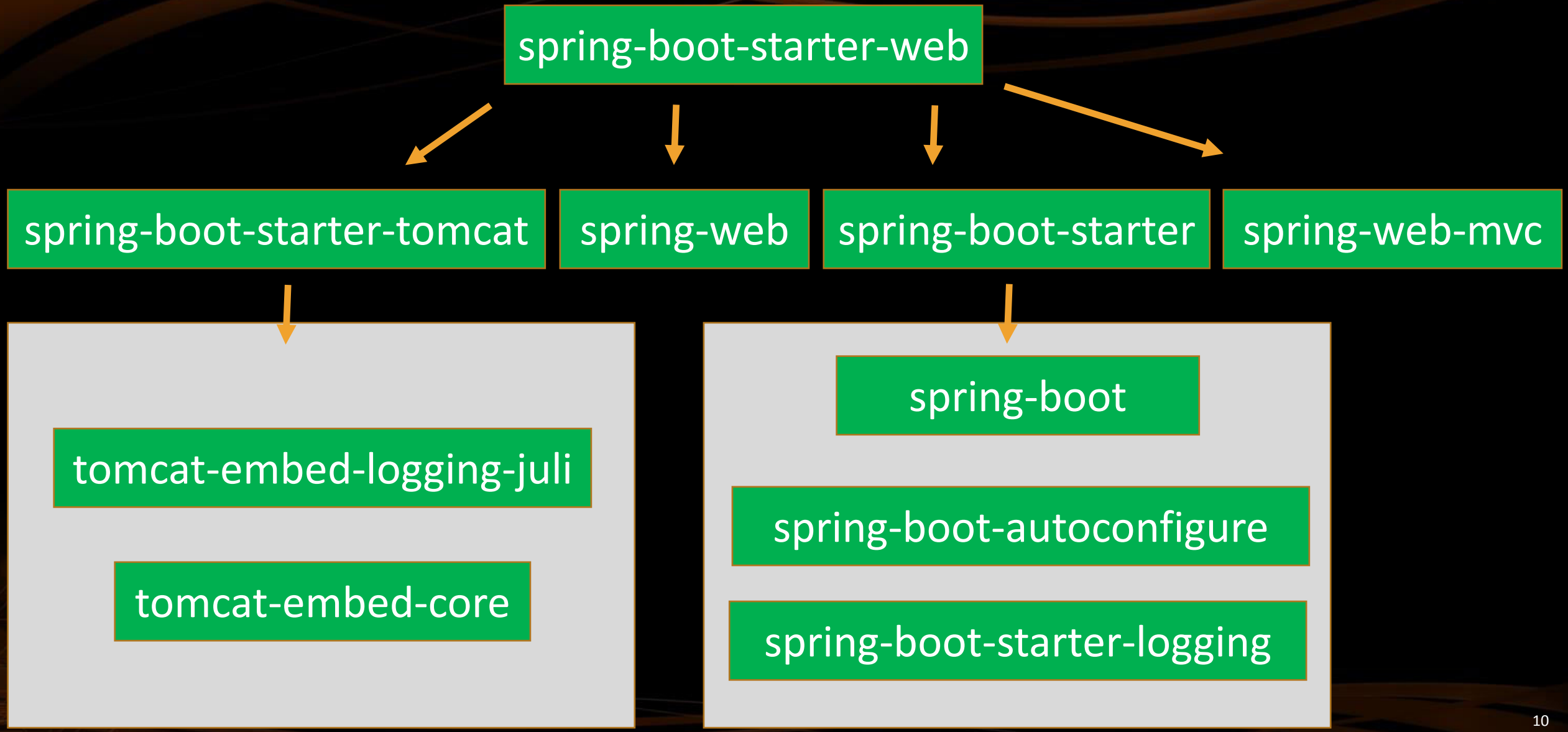


Spring Boot Main Components

SoftUni Foundation

- 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

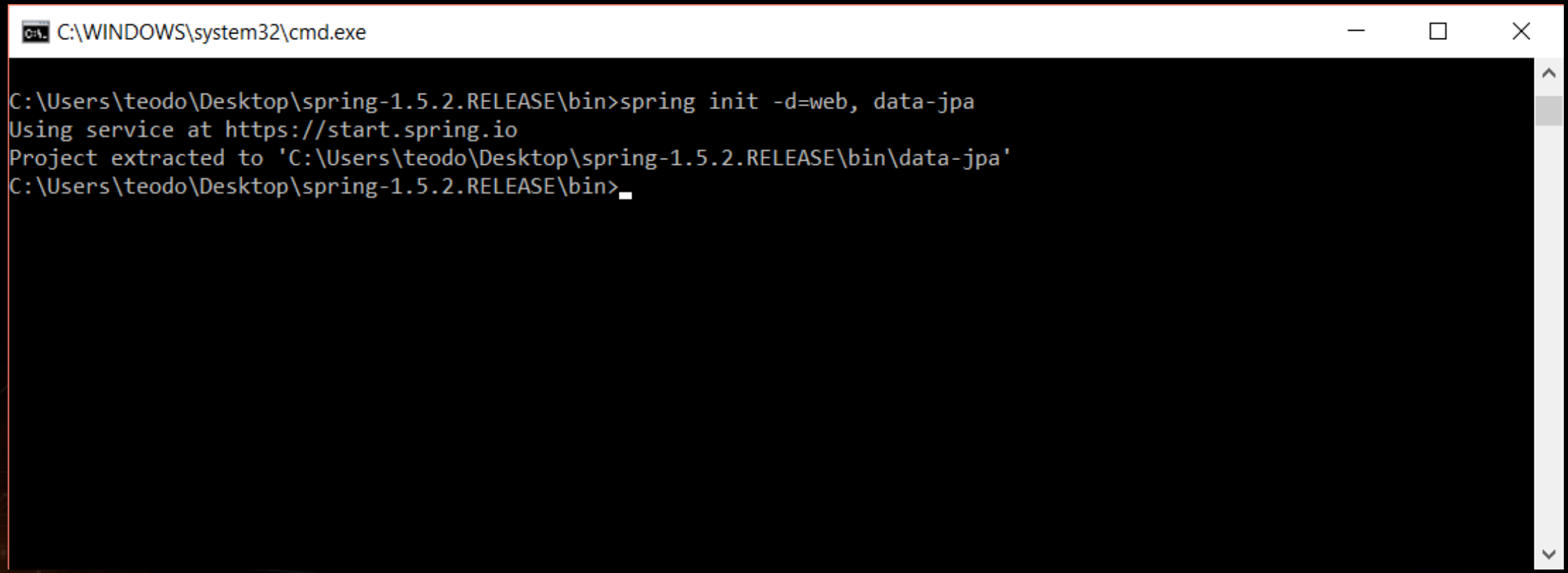
Spring Boot Starters



Spring Boot CLI

SoftUni Foundation

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



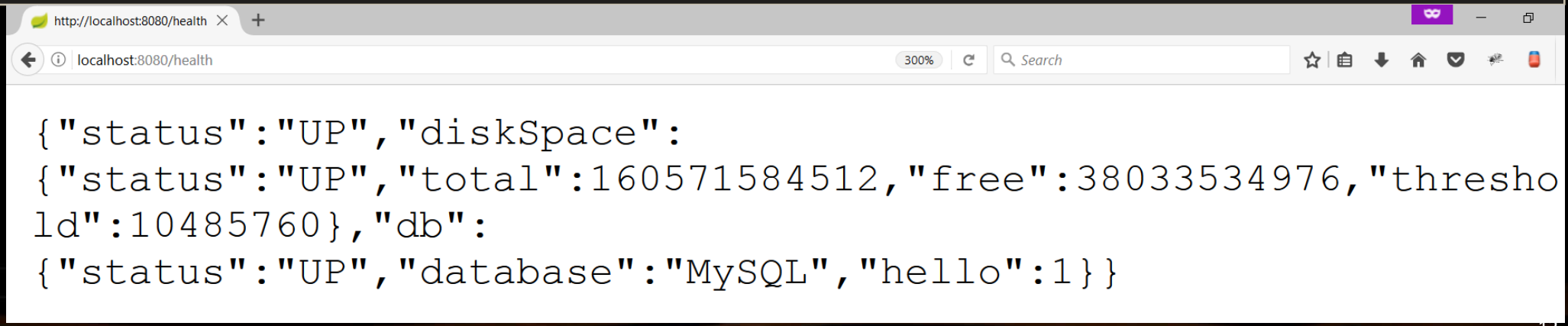
Spring Boot Actuator

SoftUni Foundation

- Expose different types of information about the running application

pom.xml

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



Inversion of Control

SoftUni Foundation

- Spring provides Inversion of Control and Dependency Injection

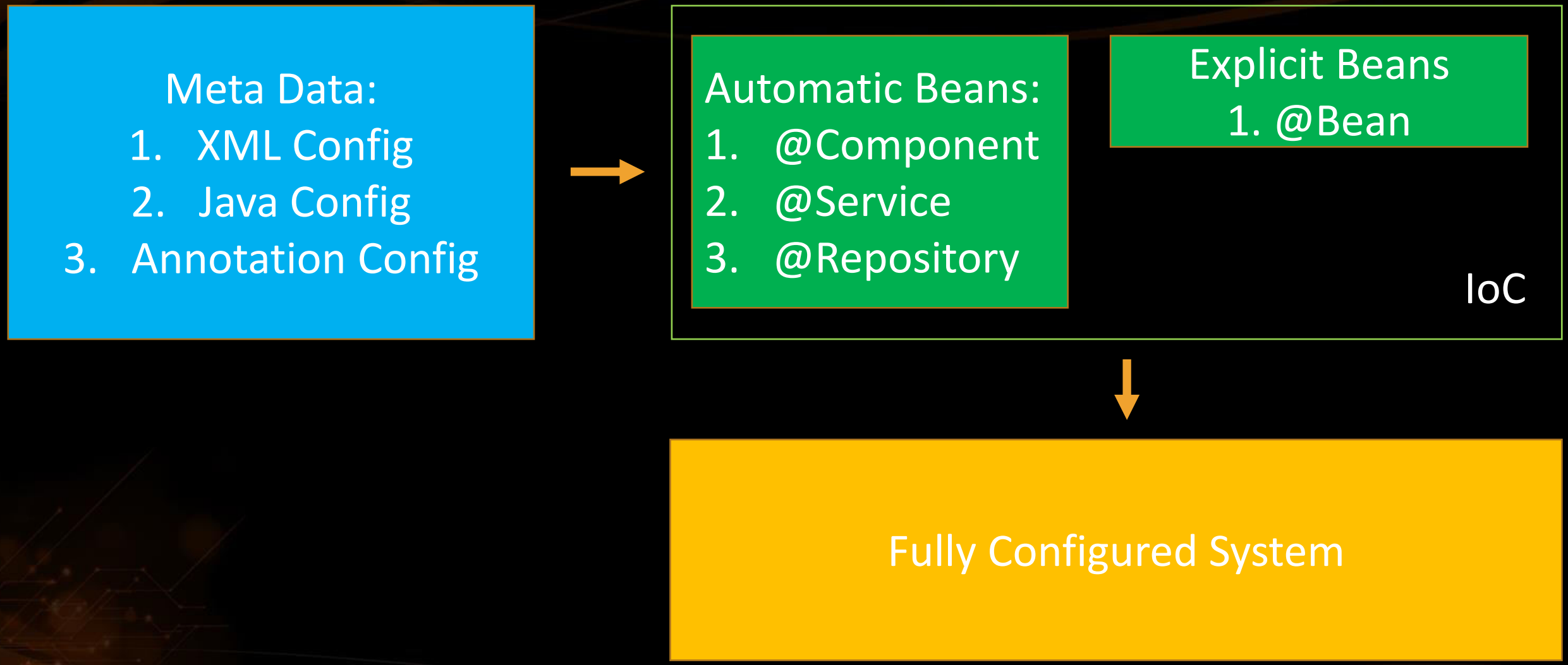
UserServiceImpl.java

```
//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

SoftUni Foundation

- 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

SoftUni Foundation

Dog.java

```
@SpringBootApplication
public class MainApplication {

    …

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

Bean Declaration

Get Bean from Application Context

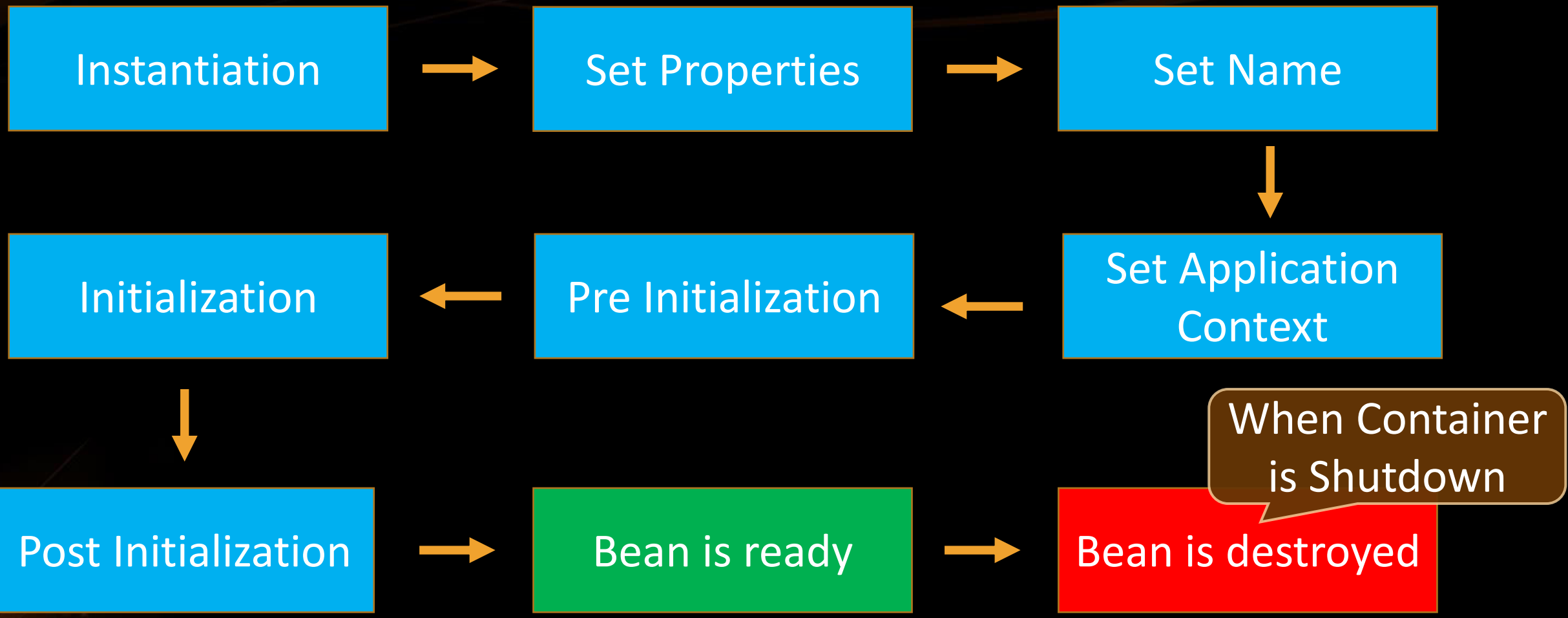
SoftUni Foundation

MainApplication.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().getSimpleNam
    }
}
```

```
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)

SoftUni Foundation

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)

SoftUni Foundation

MainApplication.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..");
    }
}
```

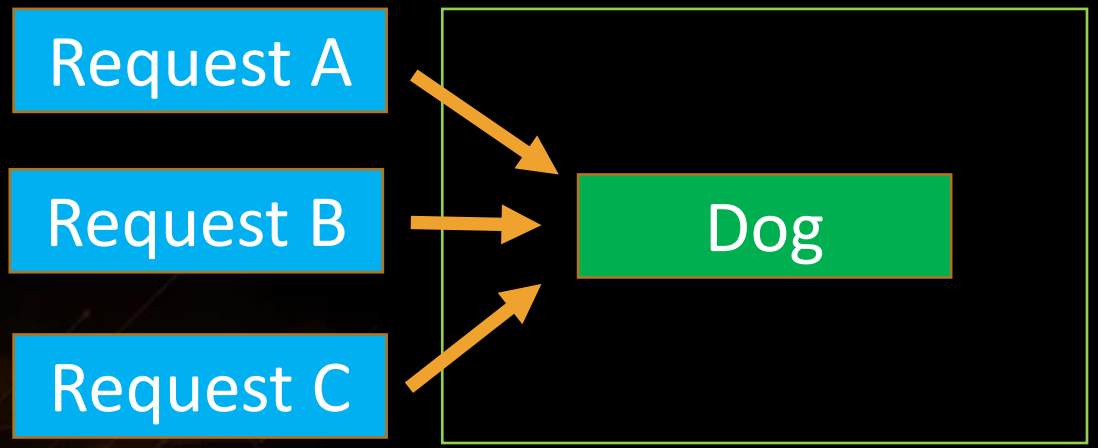
```
Instantiation
Initializing..
Destroying..
```

Bean Scope

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

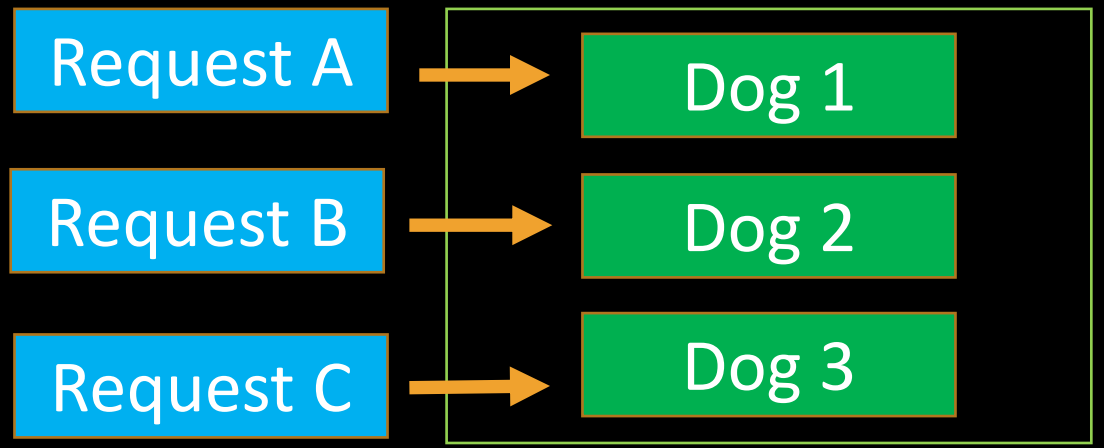
Mostly used as State-less

Singleton



Mostly used as State-full

Prototype

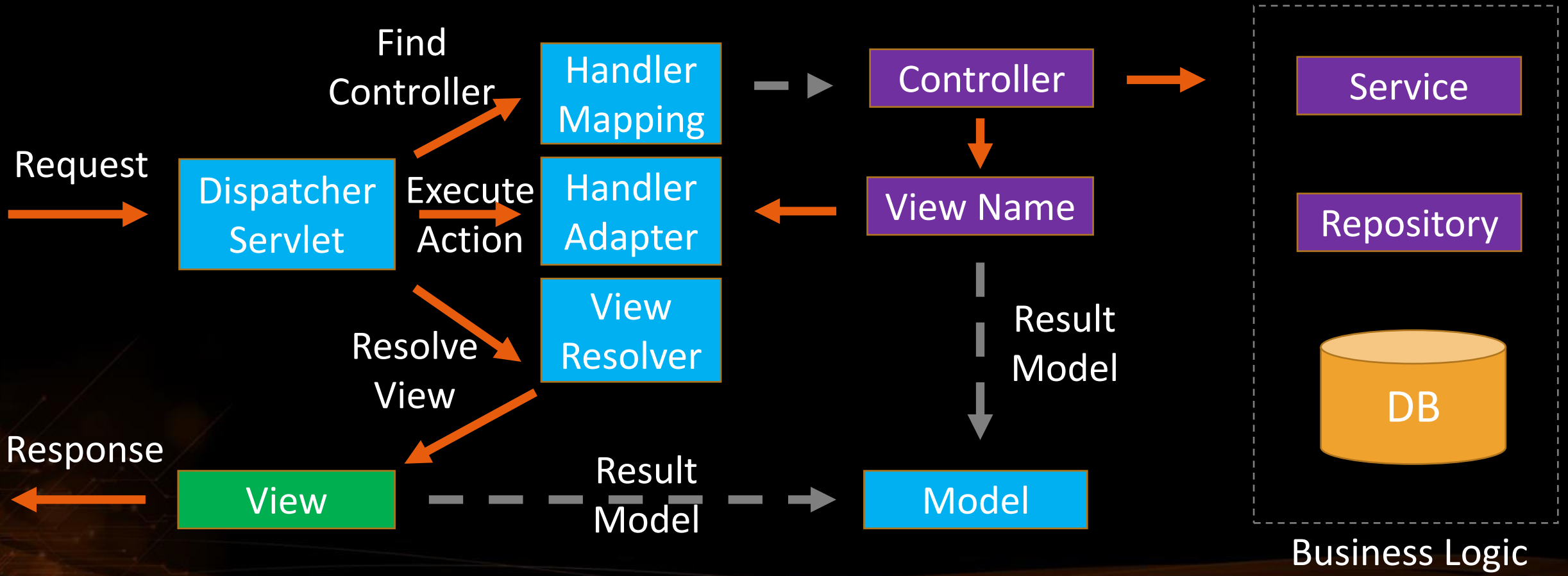


SoftUni Foundation

What is Spring MVC?

What is Spring MVC?

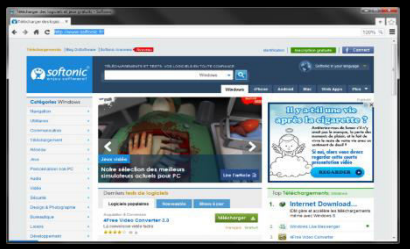
- Model-view-controller (MVC) framework is designed around a DispatcherServlet that dispatches requests to handlers



MVC – Control Flow

SoftUni Foundation

Web Client



Request

Response
(html, json, xml)

Controller

Update
Model

Notify

Update
View

User Action

Model

View

Controllers

SoftUni Foundation

Controller

DogController.java

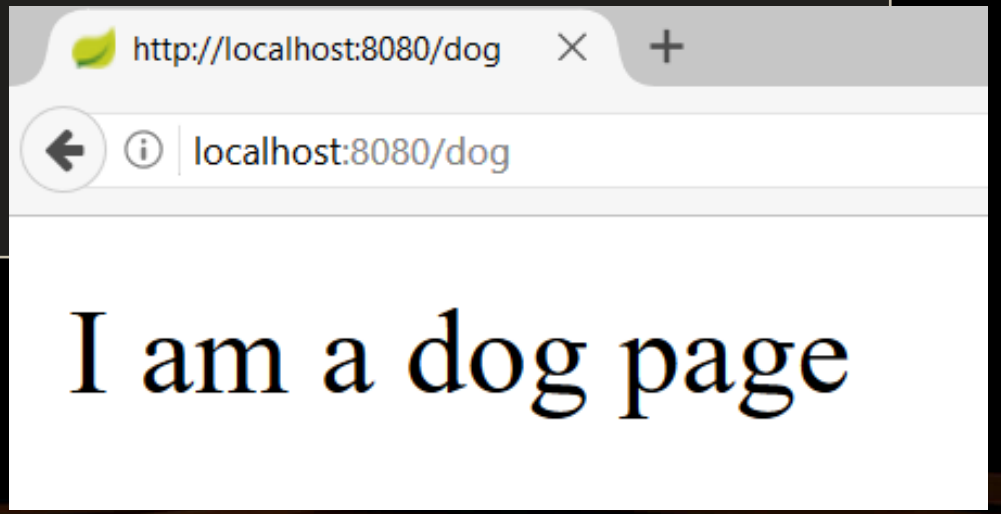
```
@Controller
public class DogController {

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

Request Mapping

Action

Print Text

Text



Actions – Get Requests

SoftUni Foundation

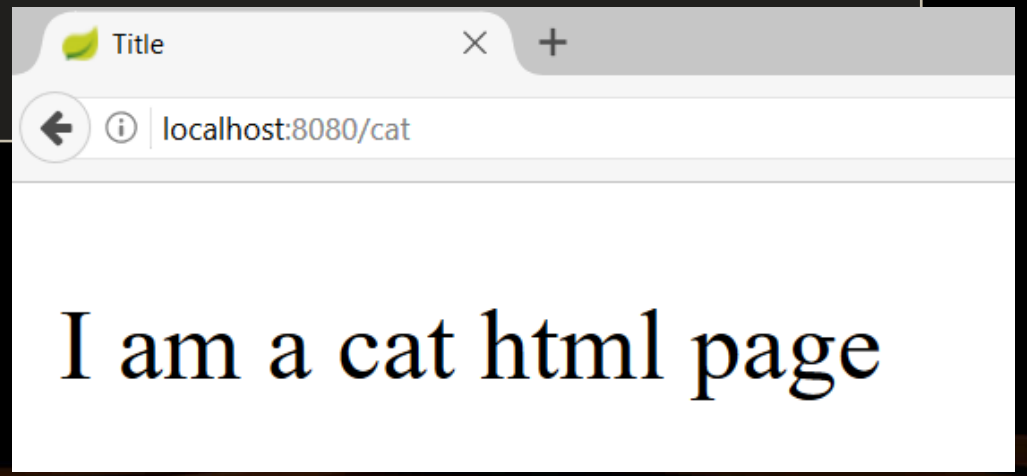
CatController.java

```
@Controller
public class CatController {

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

Request Mapping

Action

View



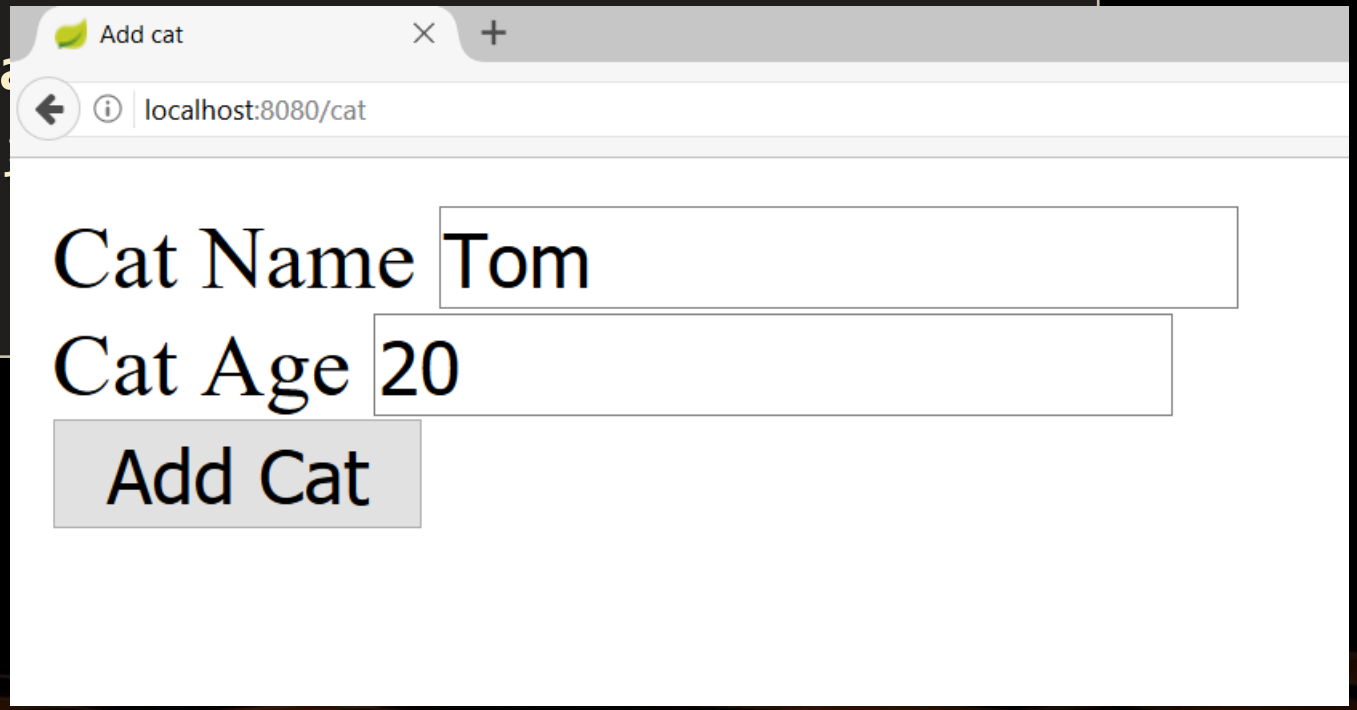
Actions – Post Requests (1)

SoftUni Foundation

CatController.java

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

    @GetMapping("")
    public String getHomeCatPa
        return "new-cat.html"
    }
}
```

Starting route



Actions – Post Requests (1)

SoftUni Foundation

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
```

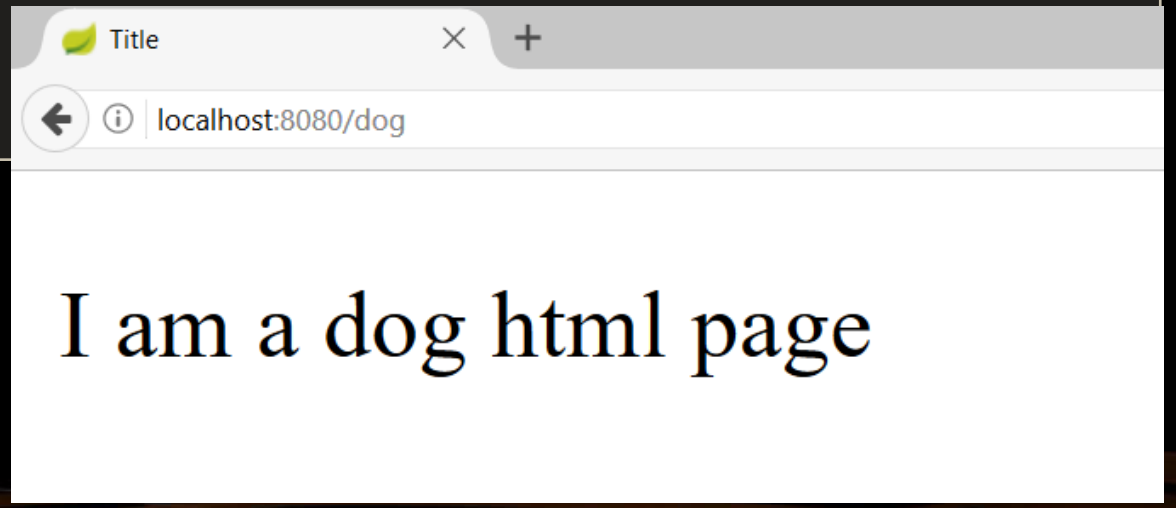
Models and Views

SoftUni Foundation

DogController.java

```
@Controller
public class DogController {

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

Model and View



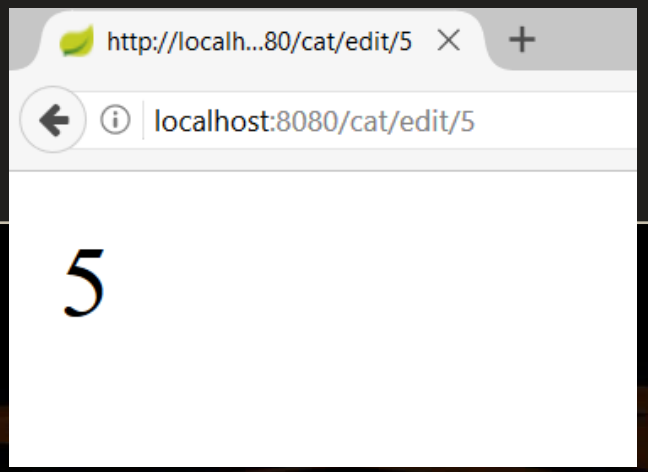
Path Variables

SoftUni Foundation

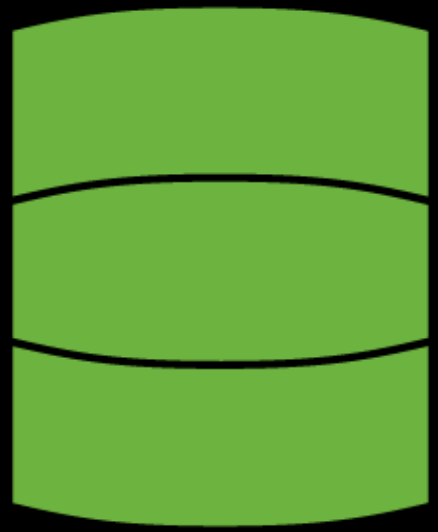
CatController.java

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

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

Path Variable

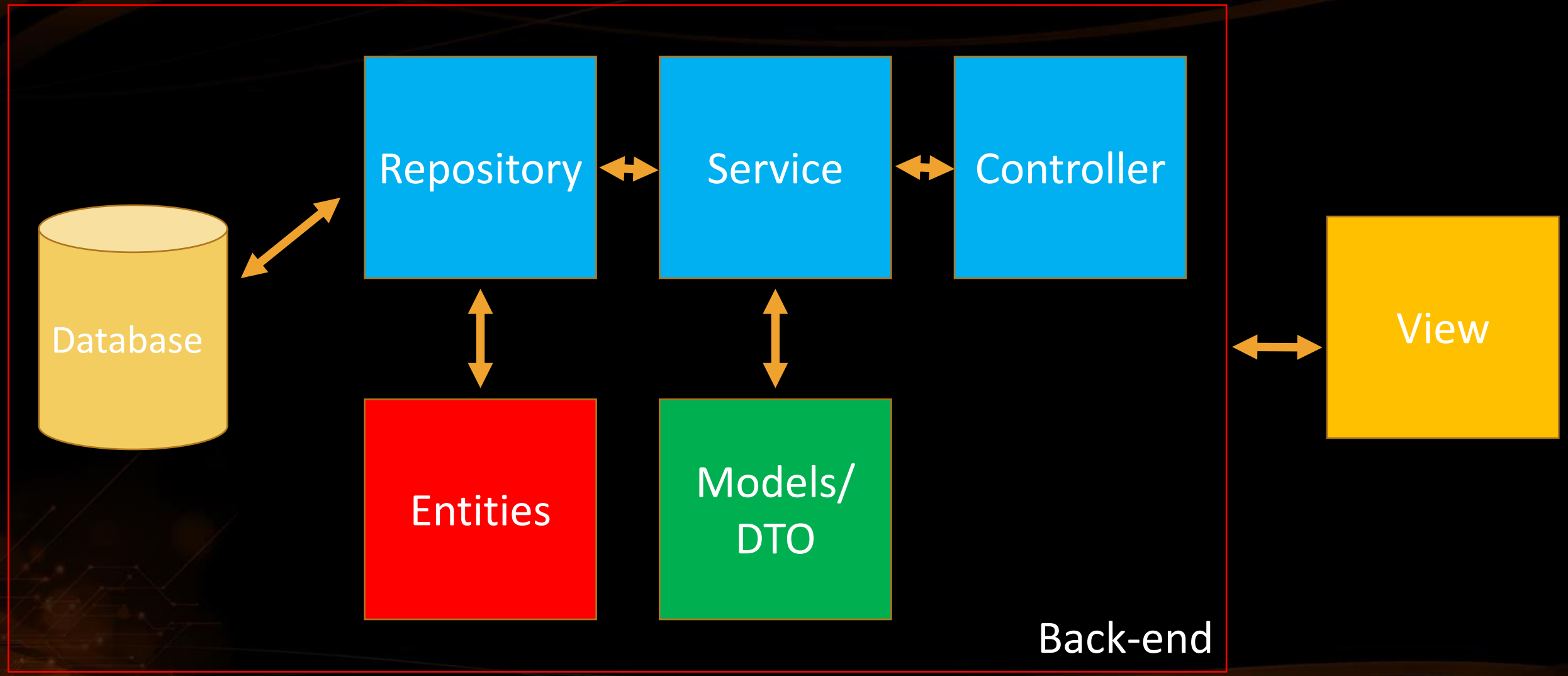


Spring MVC Demo



Spring Data

Overall Architecture



Application Properties

SoftUni Foundation

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

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

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

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

Repositories

SoftUni Foundation

- Persistence layer that works with entities

CatRepository.java

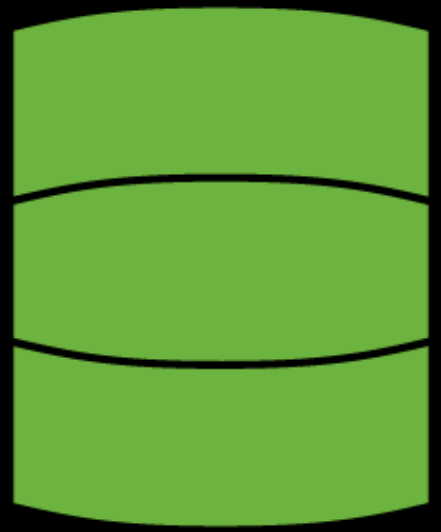
```
@Repository
public interface CatRepository extends CrudRepository<Cat, Long> {
}
```

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
    }
}
```



Spring Data Demo

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

Java MVC Frameworks – Spring Boot

SoftUni Foundation

Questions?

https://softuni.bg/courses/java-mvc-frameworks-spring

License

SoftUni Foundation

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

Software University

SoftUni Foundation

FORUM