

Vývoj pokročilých aplikácií

Framework Spring



doc. Ing. **Jozef Kostolný**, PhD.
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline
jozef.kostolny@fri.uniza.sk

Ing. **Martin Mazúch**
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline
martin.mazuch@fri.uniza.sk



ŽILINSKÁ UNIVERZITA V ŽILINE
Fakulta riadenia
a informatiky

Spring Framework

- k dispozícií od 2003
- autor **Rod Johnson**, prvé spustenie na **SourceForge**
- najpoužívanejší Java EE Framework
- poskytuje prepracovaný programovací a konfiguračný model
- množstvo rozšírení
- Hlavná výhoda:
 - Dependency Injection



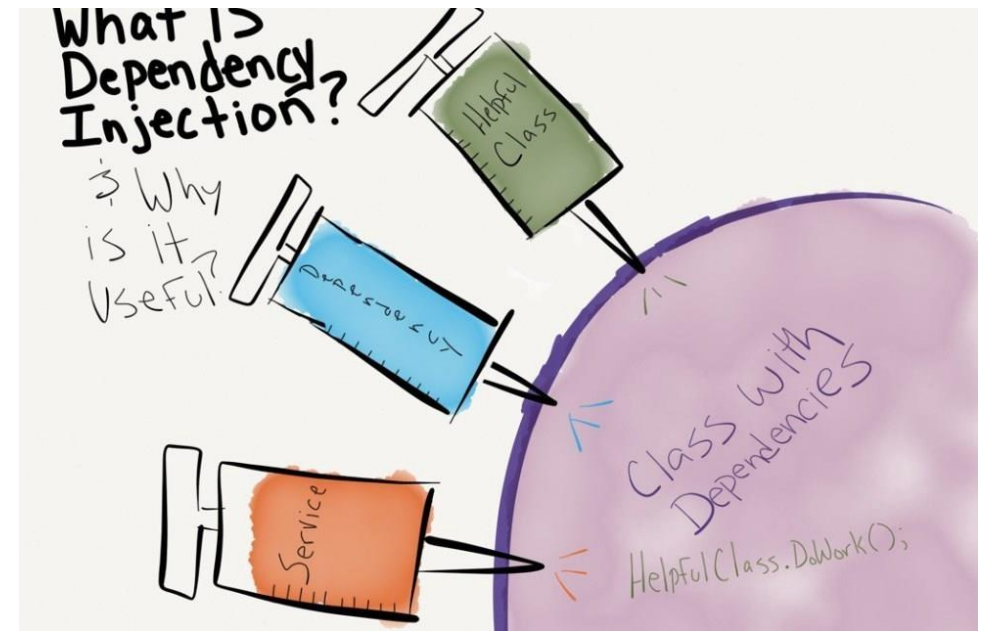
Spring – rýchlejší vývoj

- Jednoduchší vývoj EE aplikácií
- Väzba na POJO nahradená Inversion of Control
 - Kontajnerizovanie, frameworkovanie
- Možnosť voľby implementácie pre architektúru
- Podpora implementovania komponentov pre prístup k údajom – JDBC, ORM – Hibernate
- Abstrakcia a zjednodušenie používania JDBC, JavaMail
- Jednoduchšie písanie Unit testov



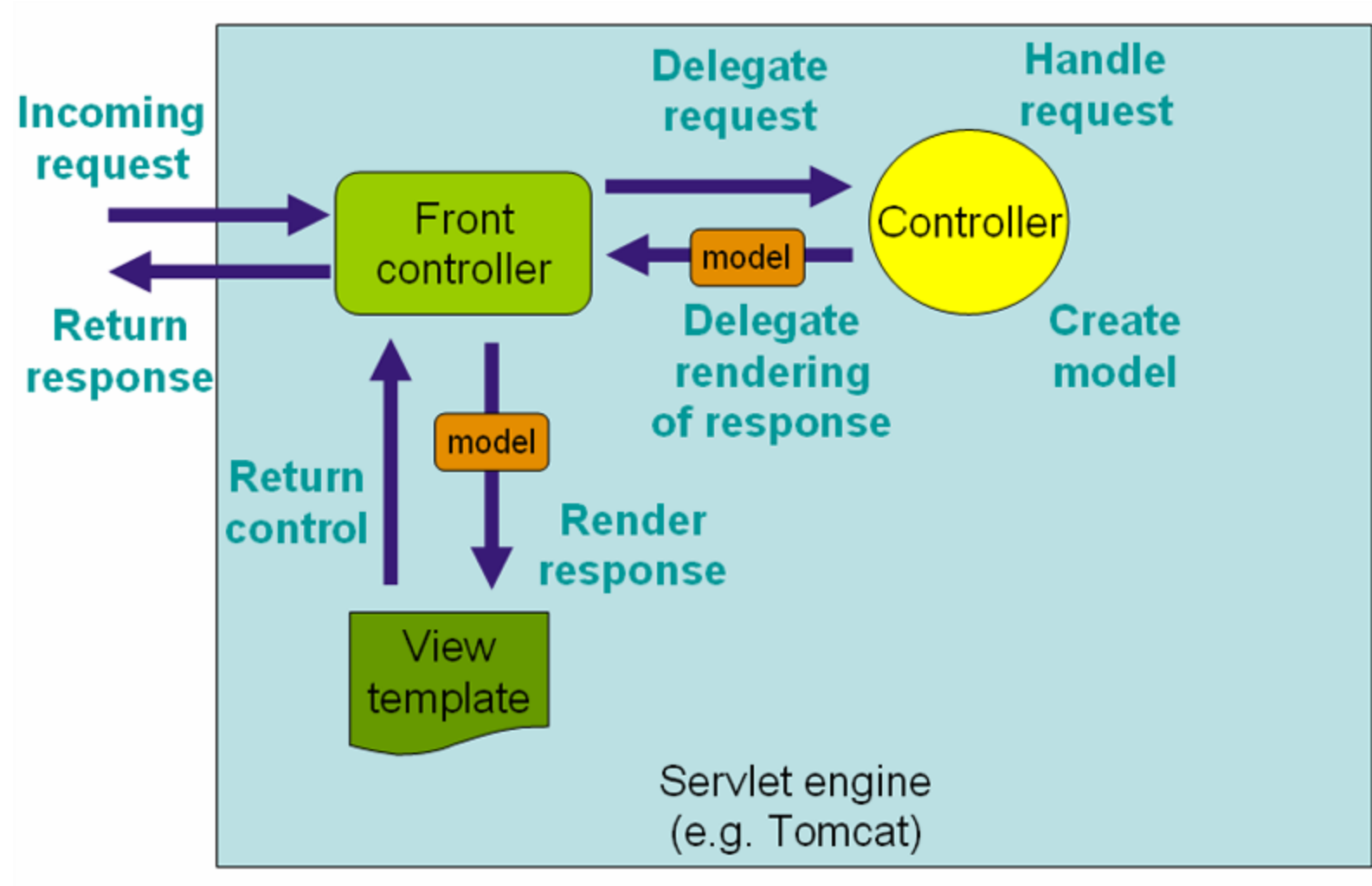
Control a Dependency Injection

- Jadro - návrhový vzor Inversion Control
 - IoC kontajner
- Aplikácia nezodpovedá za vytváranie objektov -> Framework
- Objekty sú JavaBeans



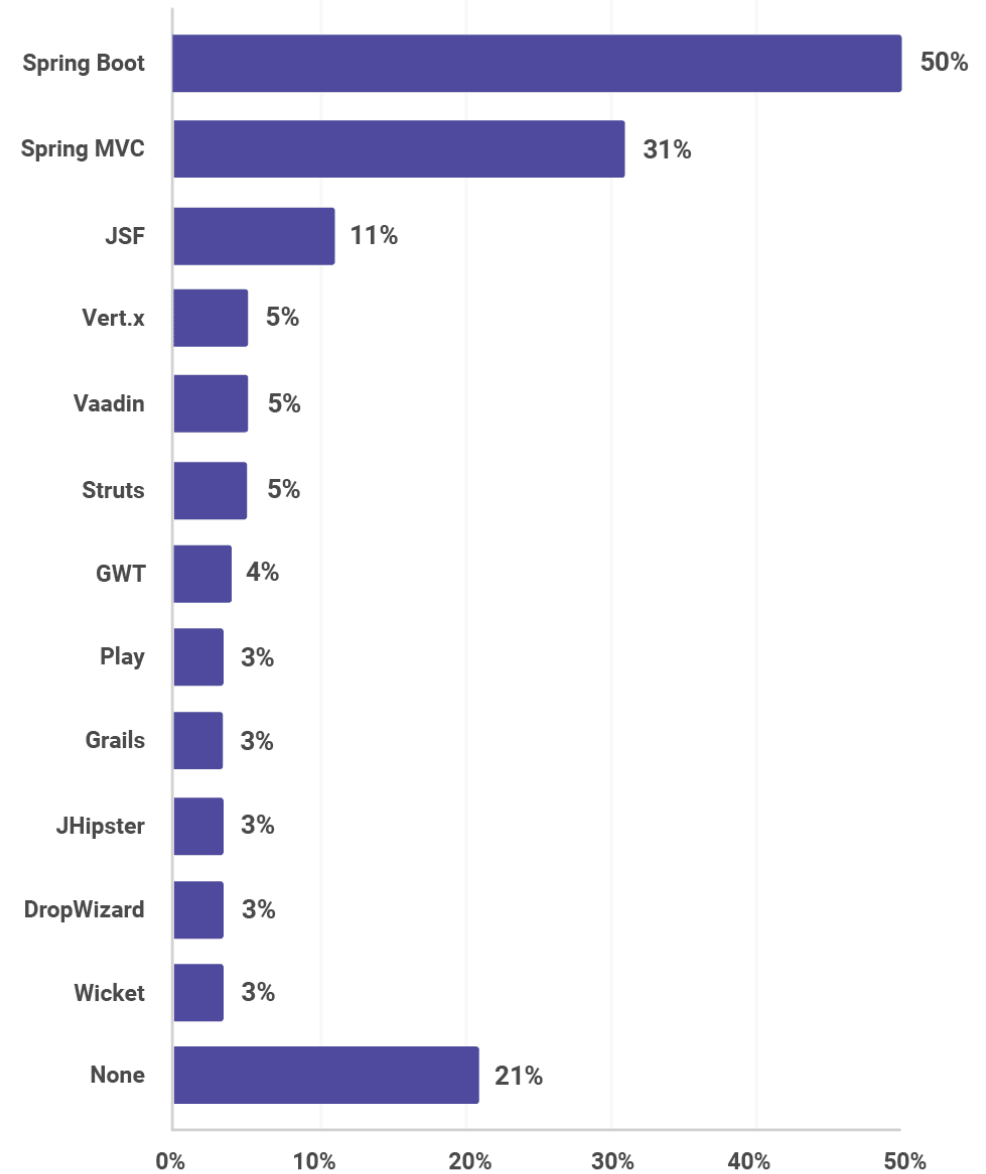
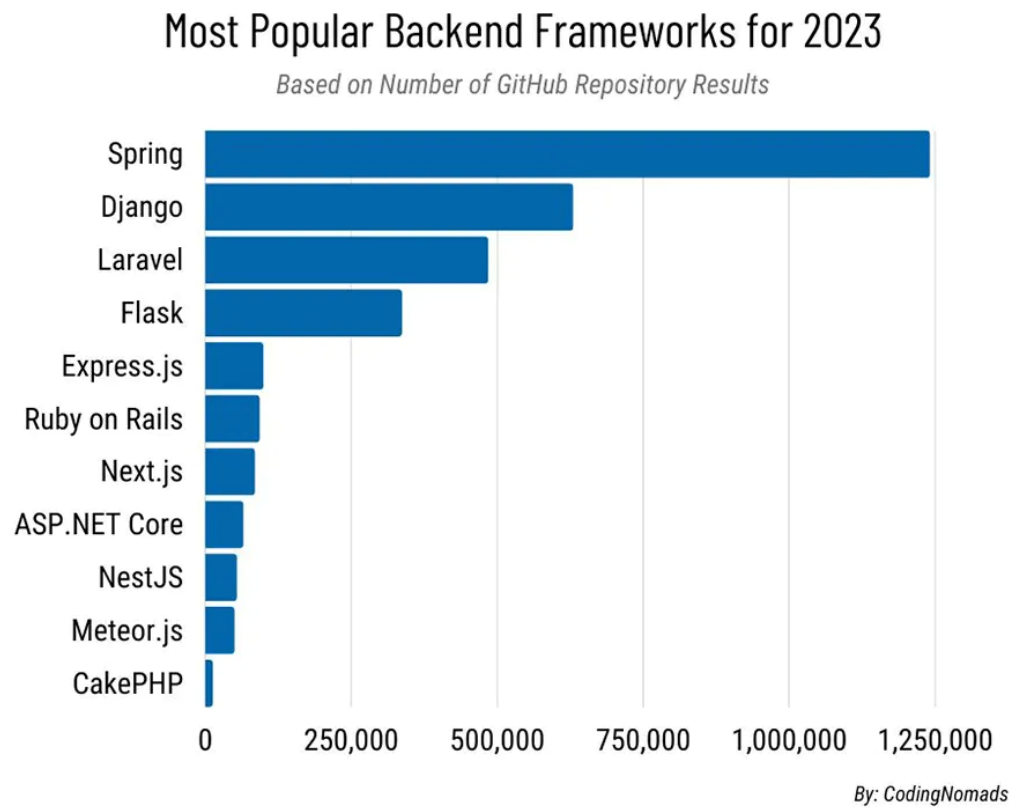
Spring MVC


- aplikácia modelu MVC



Dominancia Spring-u

- najpopulárnejšie server-side frameworky



Multiple responses allowed. 

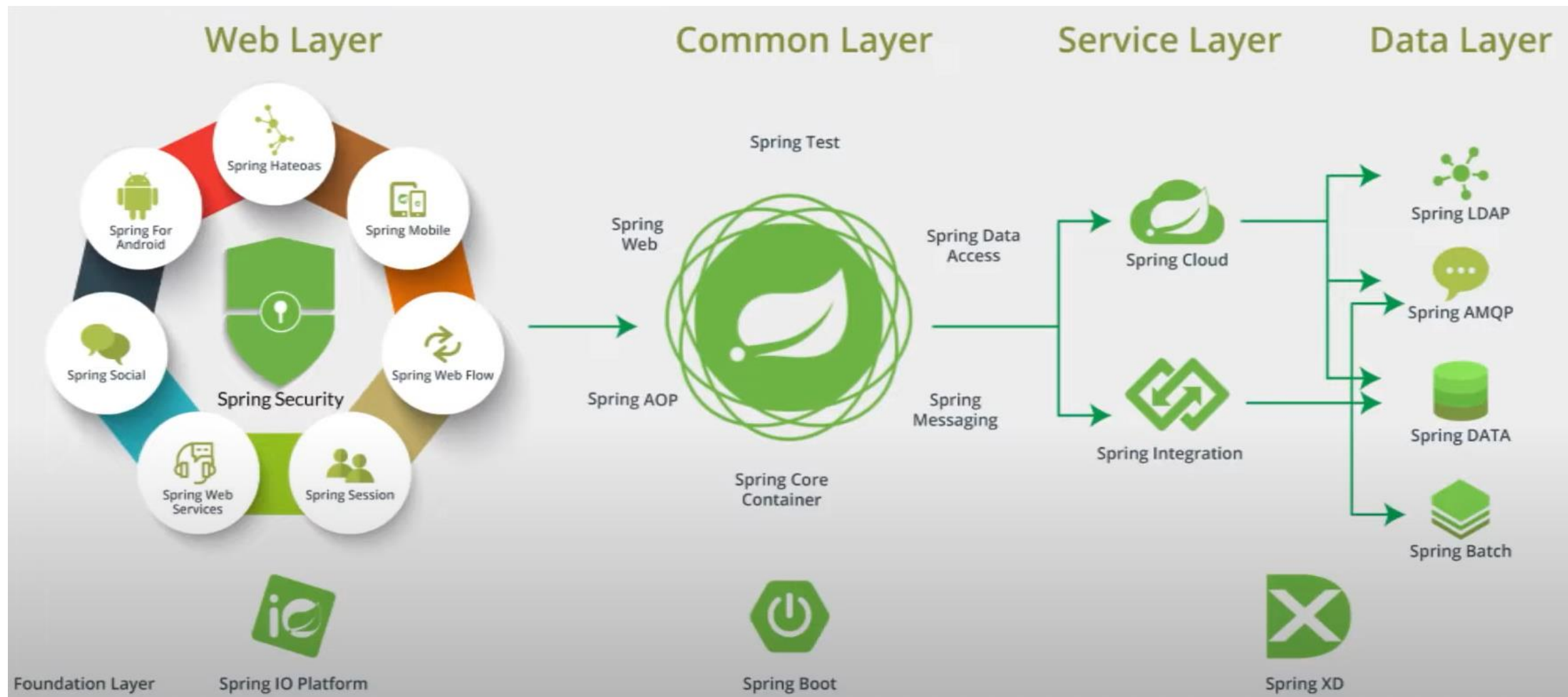


Prednosti Spring

						
1. Fast Development	✓	✗	✓	✓	✗	✓
2. Pre-Built Code	✓	✗	✓	✓	✓	✗
3. Easy Integration with third party	✓	✗	✓	✓	✗	✗
4. Accommodates UI Constraints	✓	✓	✗	✗	✓	✓
5. Efficient Performance	✓	✓	✗	✓	✗	✓



Spring framework „ekosystém“

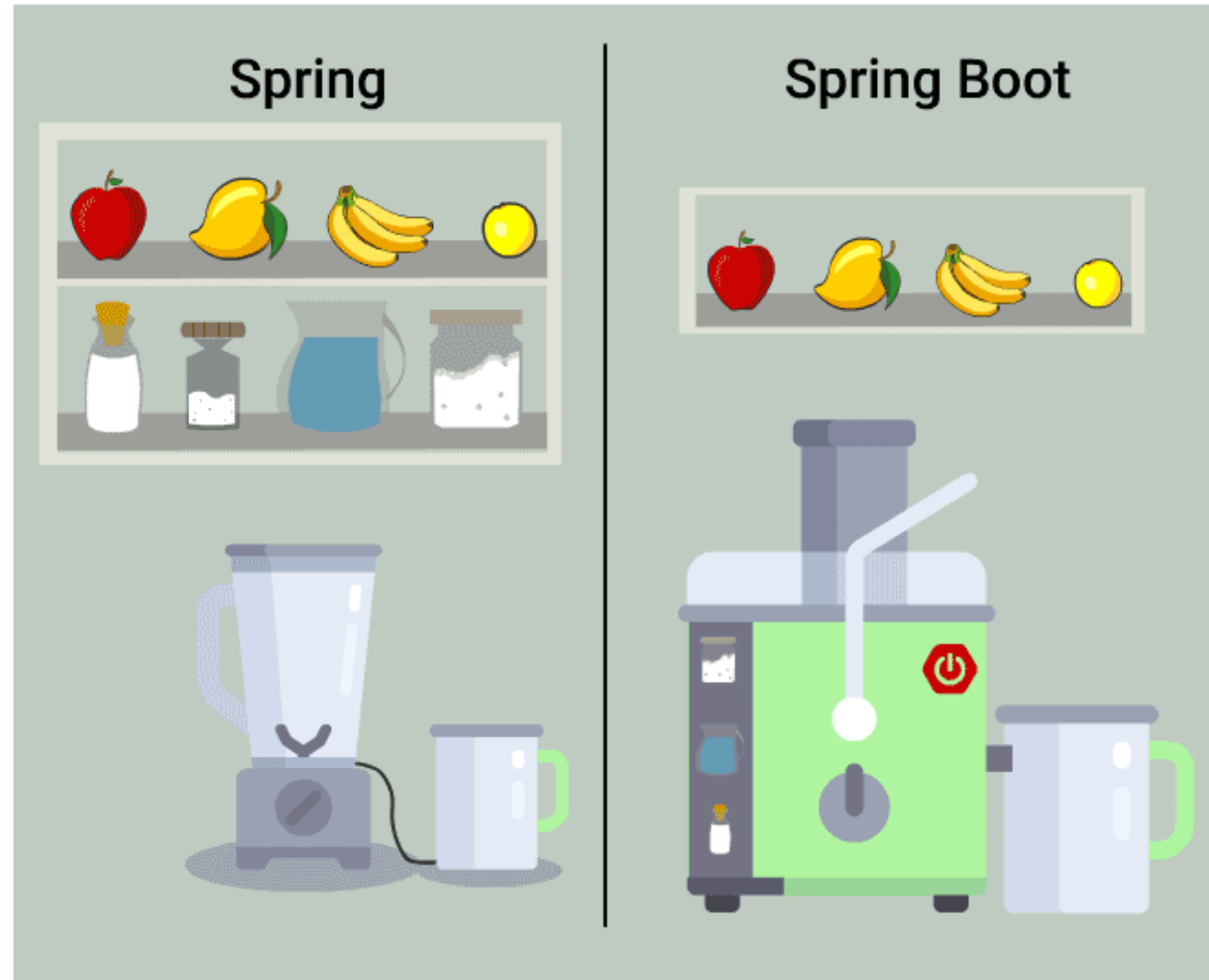


Spring boot

Framework od “Spring tímu” využívajúci jednoduchosť bootstrappingu pre vývoj Spring-ových aplikácií



Spring vs Spring Boot

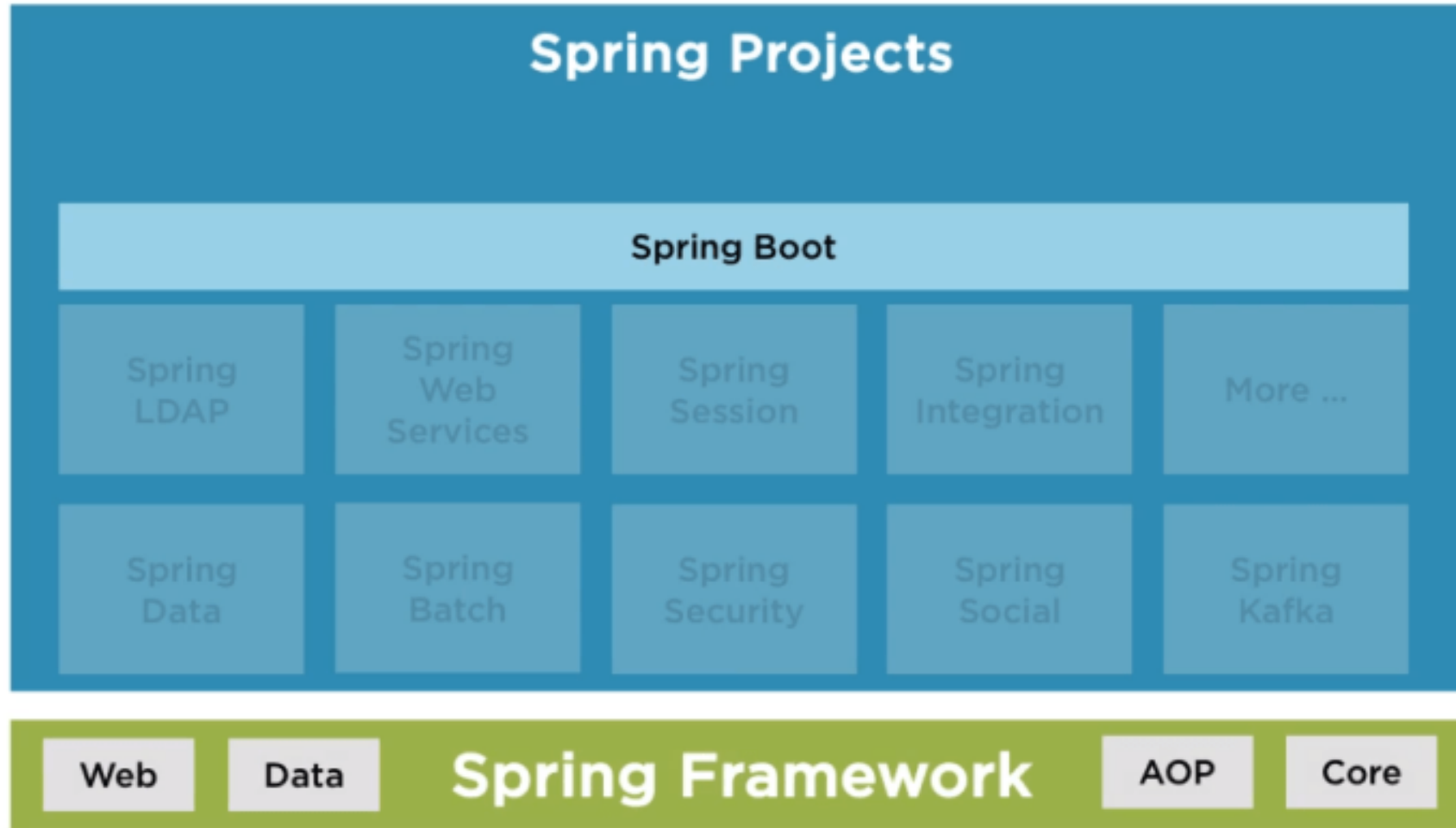


Spring Boot

- Poskytuje:
 - Automatická konfigurácia pre bežnú funkcionálnosť webových aplikácií (*JPA, bezpečnosť, Spring MVC*)
 - Spúšťač závislosti – vyberáme len technológiu
 - Prístup cez príkazový riadok – nie je potrebné vytváranie tradičného projektu
 - Možnosť skúmať bežiacu aplikáciu
 - Možnosť stopovať HTTP, stav vlákien
- Čo nie je:
 - Aplikačný server
 - Negeneruje kód
 - Vkladá servletový kontajner



Štruktúra projektu



Výhody použitia Spring Boot

- Rýchly štart s použitím Spring Initializer
 - <https://start.spring.io/>
- Vývoj:
 - REST API, WebSocket, web, streaming
- Zjednodušená bezpečnosť
- Podpora SQL a NoSQL
- Podpora zabudovaných serverov:
 - Tomcat, Jetty
- Podpora LiveReload, Auto Restart
- Podpora IDE:
 - Spring Tool Suite, IntelliJ IDEA, Netbeans



Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Maven

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 3.3.0 (SNAPSHOT) ☐ 3.3.0 (M2) ☐ 3.2.4 (SNAPSHOT) ☒ 3.2.3
☐ 3.1.10 (SNAPSHOT) ☐ 3.1.9

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 21 ☒ 17

Dependencies

[ADD ...](#)



GENERATE

EXPLORE

SHARE...



Spring Boot hlavné komponenty

- Spring Boot Starters
 - Kombinuje skupinu závislostí do jednej
- Spring Boot AutoConfigurator
 - Zjednodušuje konfiguráciu Springu
- Spring Boot CLI
 - Možnosť spúšťania a testovania aplikácií z cmd
- Spring Boot Actuator
 - Poskytuje EndPointy a Metriky

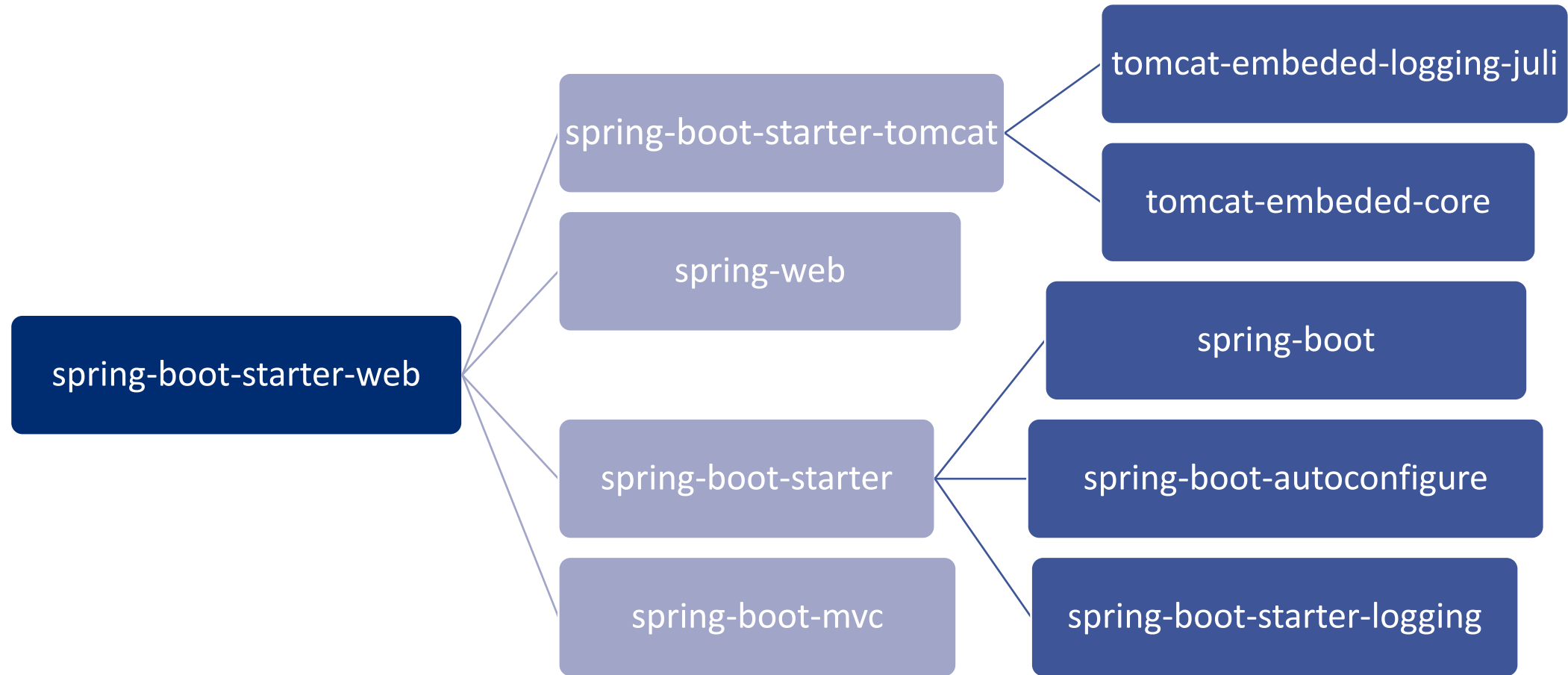


Spring Boot Starter

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-web</artifactId>  
  <version>3.2.3</version>  
</dependency>
```



Spring Boot Starter – zloženie



Spring Boot AutoConfiguration

pom.xml

```
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(
    excludeFilters = {@Filter(
        type = FilterType.CUSTOM,
        classes = {TypeExcludeFilter.class}
    )}
)
public @interface SpringBootApplication
```

14



Spring Boot CLI

- Príkazový riadok
 - Možnosť spustiť a testovať Spring Boot aplikácie

```
MINGW64: c:\_workspace\spring-app\build\libs
```

Owner@THINKSTATION MINGW64 /c/_workspace/spring-app/build/libs
\$ java -jar spring-app-1.0.jar

```

      .
     /\  /____'-----(-)-----\\ \ \ \ \
    ( ) \___| : _| : _| : _| V _| \\ \ \ \ \
     \V ___)| |_)||_| ||_| ||_| (| | ) ) ) )
       |___| : _| ||_| ||_| ||_| \_, | // // //
=====|_|=====|___/=/_/_/_/
:: Spring Boot ::                (v3.1.2)

```

2023-08-06T15:24:02.877-04:00 INFO 15204 --- [main] com.mcnz.spring.SpringAppApplication : Starting SpringAppApplication v1.0 using Java 17.0.7 with PID 15204 (C:_workspace\spring-app\build\libs\spring-app-1.0.jar started by Owner in C:_workspace\spring-app\build\libs)

2023-08-06T15:24:02.886-04:00 INFO 15204 --- [main] com.mcnz.spring : No active profile set, falling back to 1 default profile: "default"

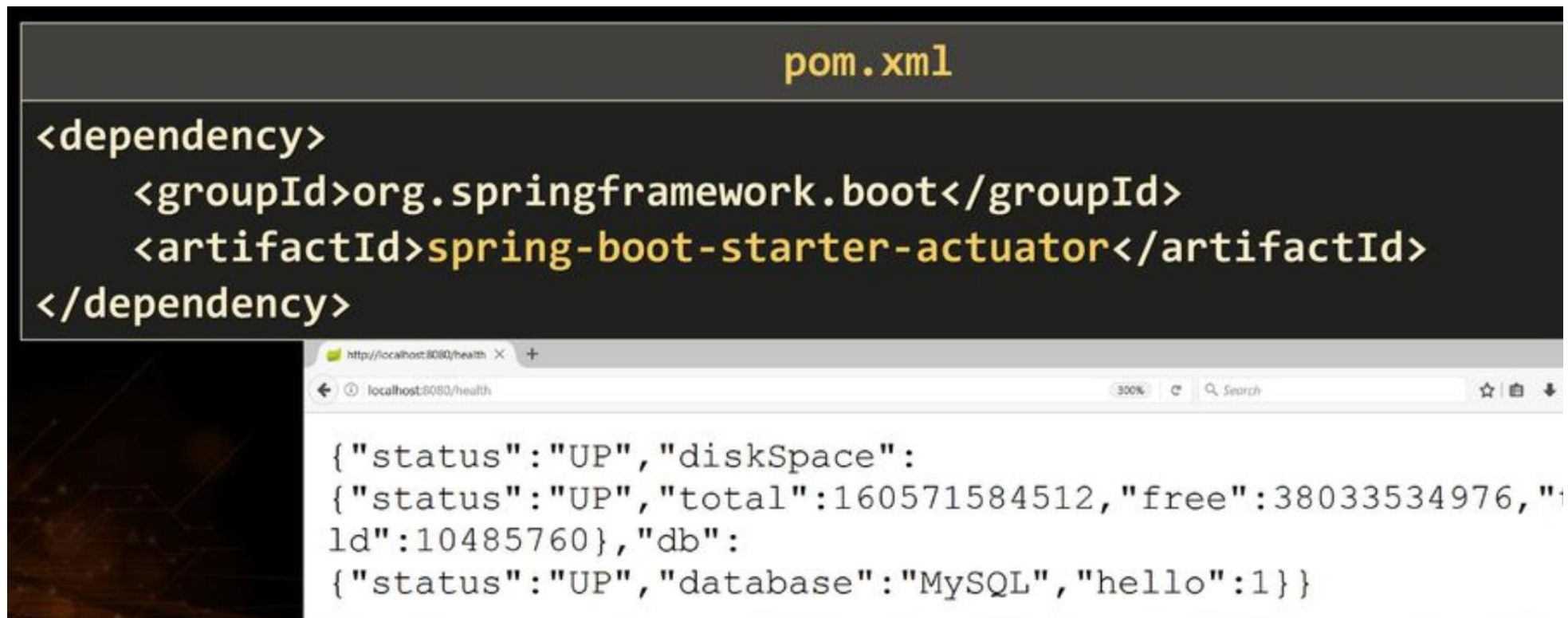
i Spring Java Reconcile

Do you wish to enable additional Java sources reconciling to get Spring specific validations and suggestions?

See [Validations And Quick Fixes](#) for more details.

Spring Boot Actuator

- Prístup k viacerým informáciám o bežiackej aplikácii



The image shows a code editor window with a dark background and a web browser window below it. The code editor displays a snippet of a `pom.xml` file, specifically a `<dependency>` block for the `spring-boot-starter-actuator`. The web browser shows the `http://localhost:8080/health` endpoint, which returns a JSON response indicating the application's health status.

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

```
http://localhost:8080/health{"status":"UP","diskSpace":{"status":"UP","total":160571584512,"free":38033534976,"ld":10485760},"db":{"status":"UP","database":"MySQL","hello":1}}
```



Inversion of Control

UserServiceImpl.java

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

    private UserRepository
    userRepository = new
    UserRepository();
}
```

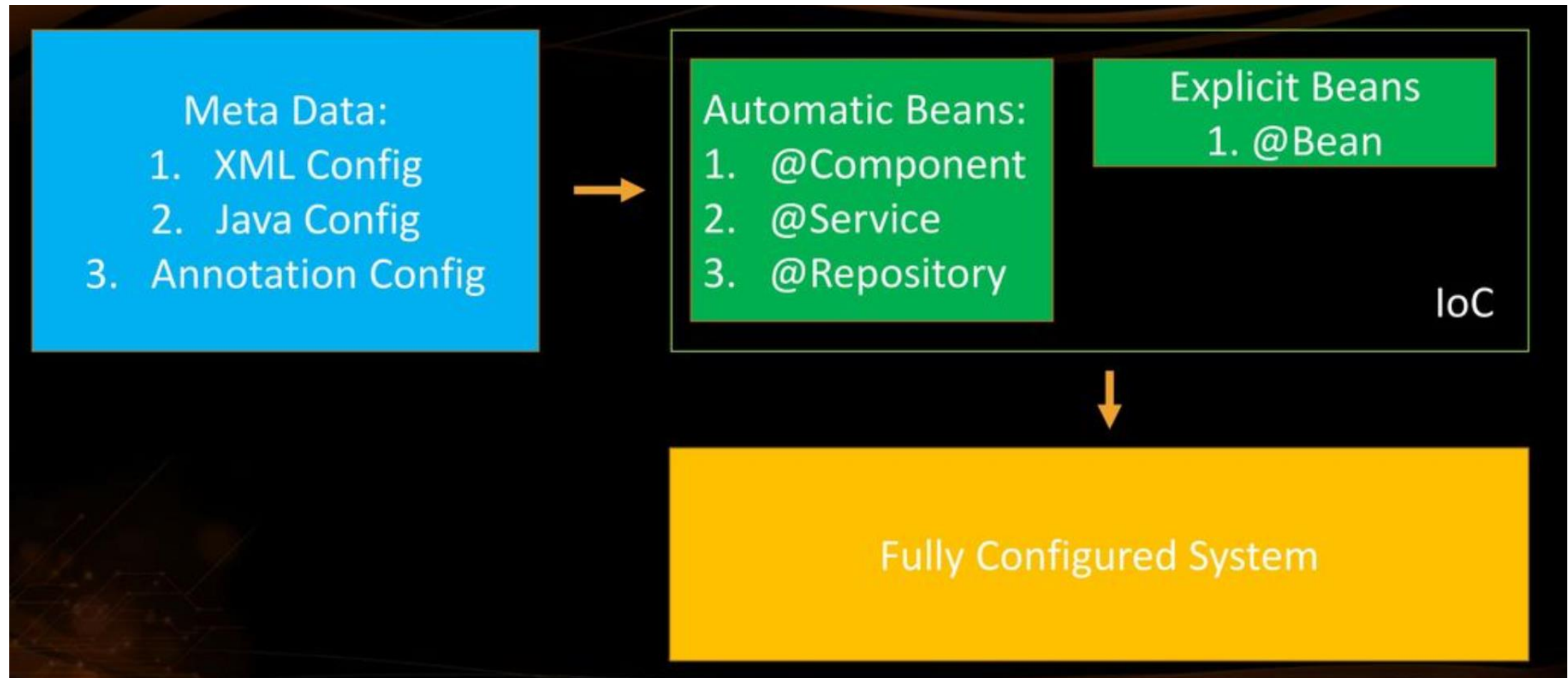
UserServiceImpl.java

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

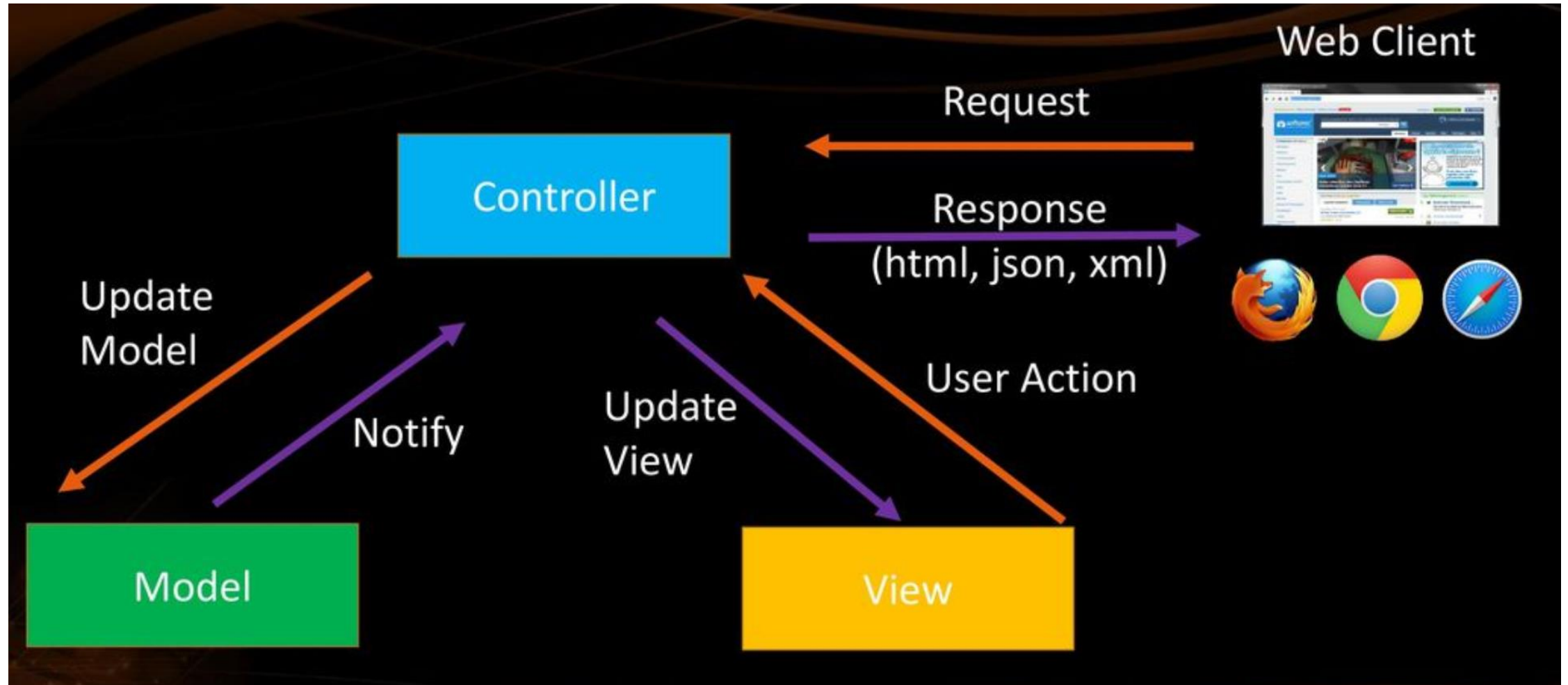
    @Autowired
    private UserRepository
    userRepository;
}
```



Inversion of Control



Spring boot MVC



Kontrolér

The diagram illustrates a Spring MVC Controller and its corresponding web browser output. The controller code is shown in a dark-themed editor window titled `DogController.java`. The code defines a `@Controller` class `DogController` with a `@GetMapping("/dog")` annotated method `getDogHomePage()` that returns the string `"I am a dog page"`. Callout boxes identify the components: **Controller** points to the class, **Request Mapping** points to the `@GetMapping` annotation, **Action** points to the method, **Print Text** points to the return statement, and **Text** points to the returned string. To the right, a web browser window shows the URL `http://localhost:8080/dog` and the rendered output `I am a dog page`.

```
DogController.java
```

```
@Controller
public class DogController {

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

Browser output: `I am a dog page`



Akcie – GET

CatController.java

```
@Controller
public class CatController {

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

Request Mapping

Action

View

Title

localhost:8080/cat

I am a cat html page



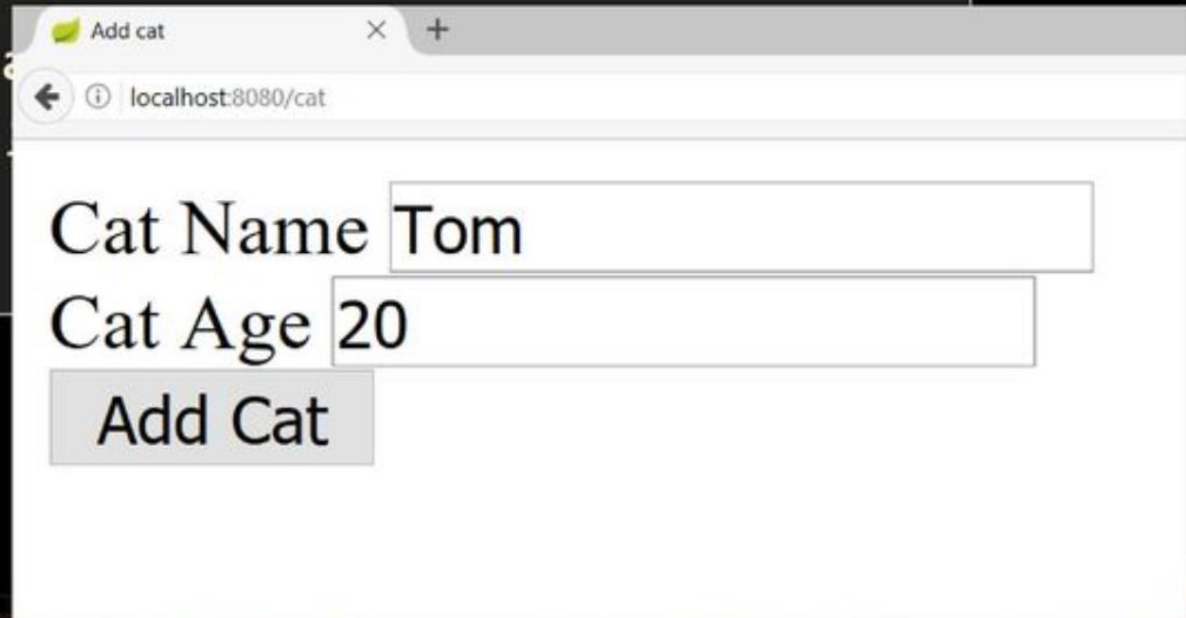
Akcie – POST 1

CatController.java

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

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

Starting route



A screenshot of a web browser window with the title "Add cat". The address bar shows "localhost:8080/cat". The page contains a form with two input fields: "Cat Name" with the value "Tom" and "Cat Age" with the value "20". Below the fields is a button labeled "Add Cat".



Akcie – POST 2

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



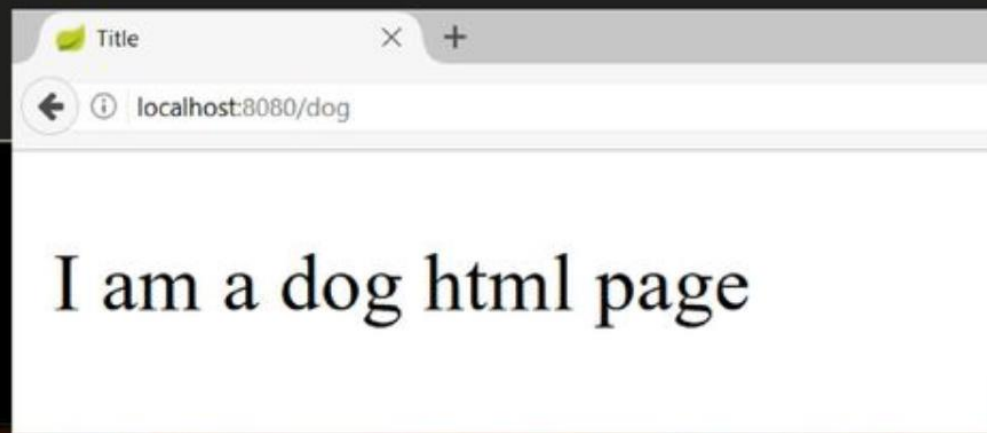
Modely a zobrazenia

DogController.java

```
@Controller
public class DogController {

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

Model and View



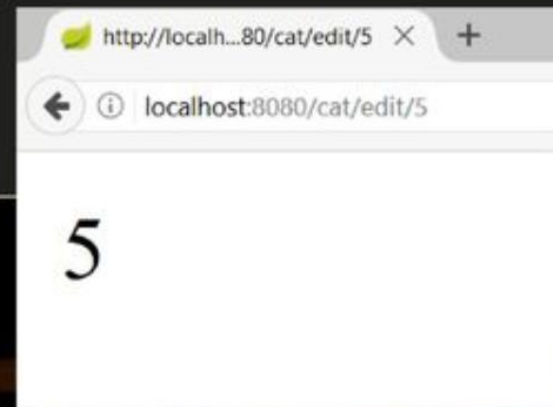
Premenné v odkaze

CatController.java

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

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

Path Variable



Začíname pracovať

Dva režimy:

- Príkazový riadok cez Groovy skript
- Správca závislosti – Maven/Gradle
 - Spustíme projekt z pom.xml





<http://www.patorjk.com/software/taag/>



Inštalácia Spring Boot v1

1. Stiahnuť a rozbaľiť distribúciu

- <https://repo.spring.io/milestone/org/springframework/boot/spring-boot-cli/>

2. pridanie cesty do PATH:

- SET PATH=%PATH%;C:\java\spring-boot\bin



Spustenie Hello World v1

HelloController.groovy :

```
@RestController  
class WebApplication {  
  
    @RequestMapping("/")  
    String home() {  
        return "Hello World!";  
    }  
}
```

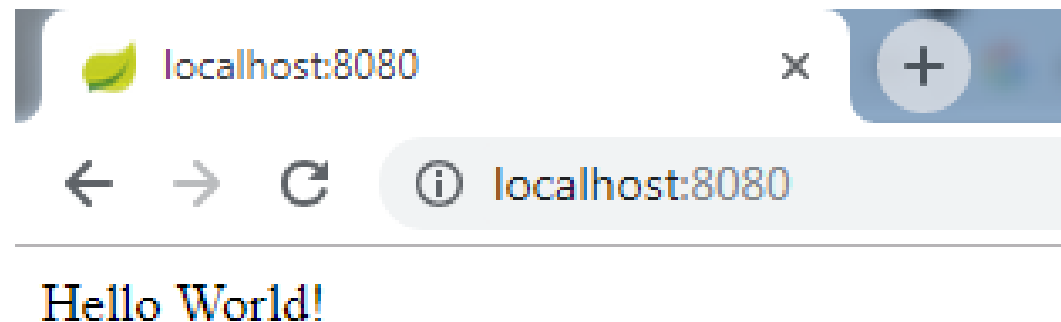


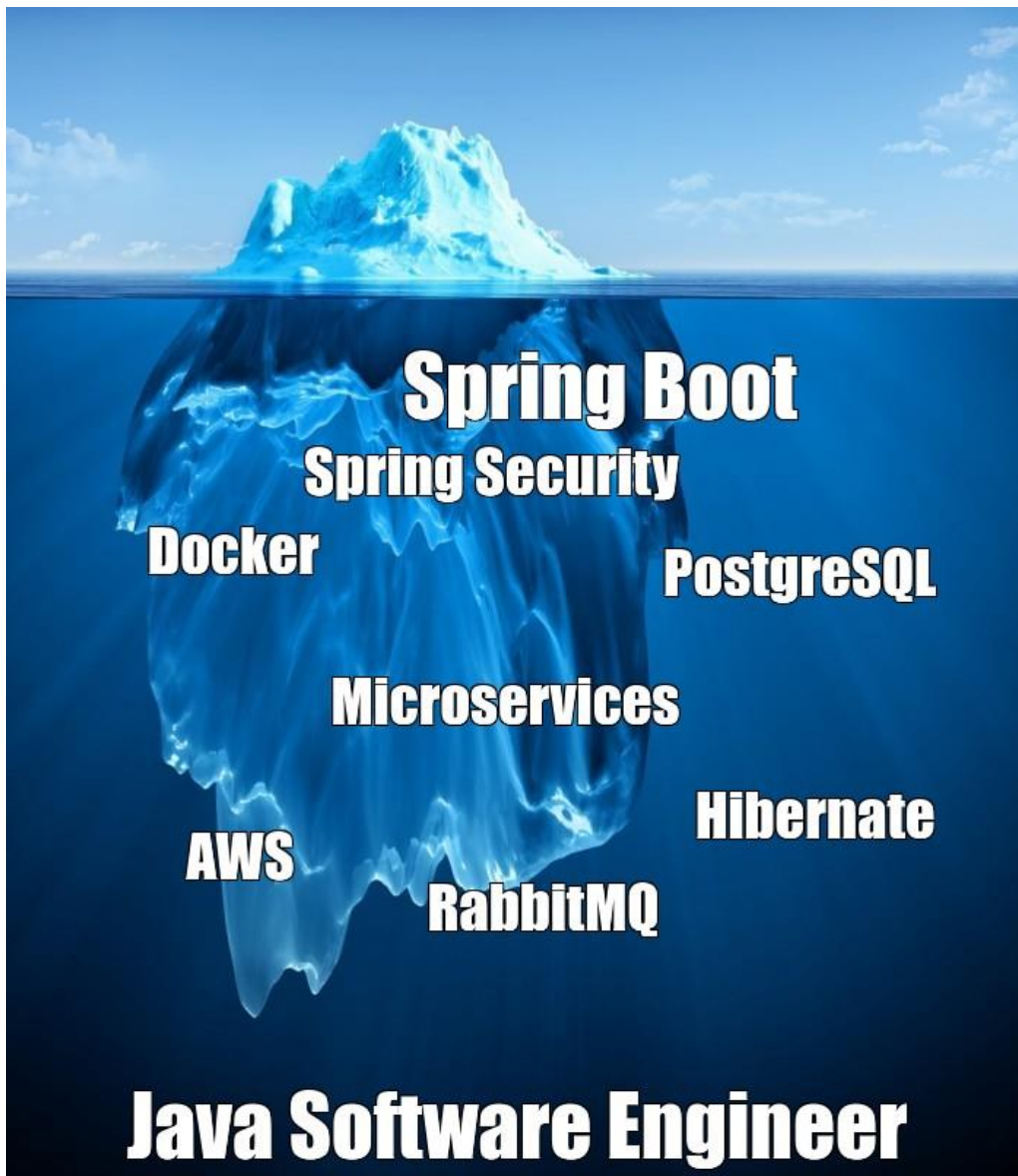
Spustenie v1

```
spring run HelloController.groovy
```

Automatické spustenie servera:

- <http://localhost:8080/>





Ďakujem za pozornosť

doc. Ing. **Jozef Kostolný**, PhD.
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline
jozef.kostolny@fri.uniza.sk

Ing. **Martin Mazúch**
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline
martin.mazuch@fri.uniza.sk