

# Vývoj pokročilých aplikácií

Framework Spring Boot – pokročilejšie techniky



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

# Obsah

- Výhody Spring Boot pri vývoji
- Príklady aplikácie šablón web aplikácii
- Bezpečnostné odporúčania



# Spring Boot

- Najpoužívanejší Java framework pre microservisy
- Pomáha s konfiguráciou Spring-u
- Jednoduché konfigurovanie a pridávanie Dependencies
- má svoje názory
  - Rozumné „default“ nastavenia
- Je ho možné prispôbovať
  - Vlastný názor bez možnosti zmien?
  - Úprava podľa vlastných predstáv
    - Pri začiatkovej konfigurácii
    - Aj pri vývojovom cykle
  - Manažment závislosti
    - Jednoduchá úprava napr. Maven \*.pom



# Auto-konfigurácia

- Jednoduché fungovanie
- Aktivované hneď po identifikácii *jar*

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

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

- *Možné vypnúť :*
  - `@EnableAutoConfiguration(exclude={ClassNotToAutoconfigure.class})`



# Spring inicializátor

- Spúšťanie nových projektov
- <https://start.spring.io/>

New Project

**Project Metadata**

Group:

Artifact:

Type:

Language:

Packaging:

Java Version:

Version:

Name:

Description:

Package:

Spring Initializr  
Bootstrap your application

[Github](#) | [Twitter](#)

**Project**

**Maven Project** **Gradle Project**

**Language**

**Java** **Kotlin** **Groovy**

**Spring Boot**

2.2.0 M1 2.2.0 (SNAPSHOT) 2.1.4 (SNAPSHOT) **2.1.3** 1.5.19

**Project Metadata**

Group

Artifact

Name

Description

Package Name

Packaging  
**Jar** **War**

Java Version  
**11** **8**



# Vlastné vs všeobecne používané

- Možnosť vytvorenia vlastnej konfigurácie
  - Problémy pri štandardnej schéme
  - Výhodné pre open source projekty
- Výhody:
  - Správna réžia
  - Údržba
  - Viacero projektov s rozdielnou konfiguráciou



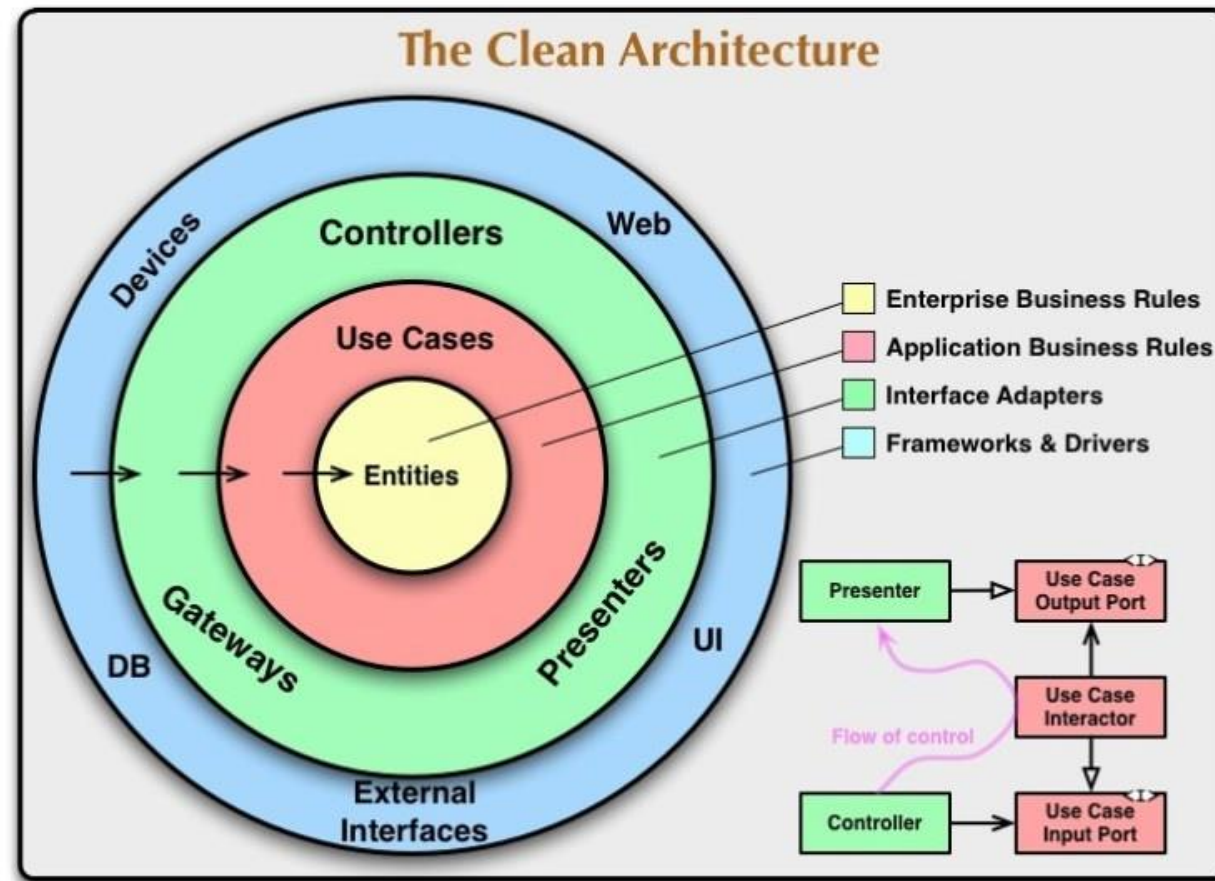
# Správna štruktúra kódu

- Veľká sloboda pri písaní
- Základné pravidlá:
  - Default package
    - používať pomenovaný balíček
  - Vlastná trieda *Application.Java*
    - Ponechať vo vrchnej vrstve priečinkov
  - Controler/Services
    - Ponechať ich s modulom, na ktorý sú previazané
    - Oddeliť a ponechať všetky spolu



# Abstrakcia databázy

- Pri návrhu DB logiky abstrahovať od Servisov
- Neorientovať sa na daný typ DB
- Príprava pre možnosť zmeny typu DE
- Rýchly vývoj
- Rýchla adaptácia





# Oddelenie Spring Boot kódu

- Ochrana business logiky
- Lákavé miešať so Spring Boot kódom
- Oddeliť:
  - Možnosť opätovného využitia
  - Vytvorenie/pretvorenie do knižníc



# Spracovanie výnimiek

Je to naozaj potrebné?

*SIMPLY EXPLAINED*



# Spracovanie výnimiek

Je to naozaj potrebné?

Jednotné spracovanie

Spring Boot riešenie:

1. *HandlerExceptionResolver*
  - Pre globálne riešenie
2. *@ExceptionHandler*
  - Vhodné pre špecifické prípady

Čítanie:

- Error Handling for REST with Spring
- <https://www.baeldung.com/exception-handling-for-rest-with-spring>

*SIMPLY EXPLAINED*



# Logovací framework

- Lepšie riešenie ako *System.out.println()*
- Bez konfigurácie v Spring Boot
- Vytvorenie inštancie triedy:

```
Logger logger = LoggerFactory.getLogger(MyClass.class);
```



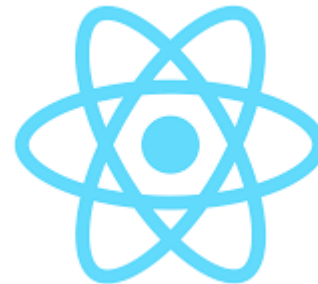
# Testovanie

- Pri Spring Boot sa väčšinou nerobí
- Pri väčších projektoch
  - príchod ďalšieho vývojára
  - Aktualizovanie staršieho kódu
  - Integrovanie s inou službou



# Frontend frameworky so Spring Boot

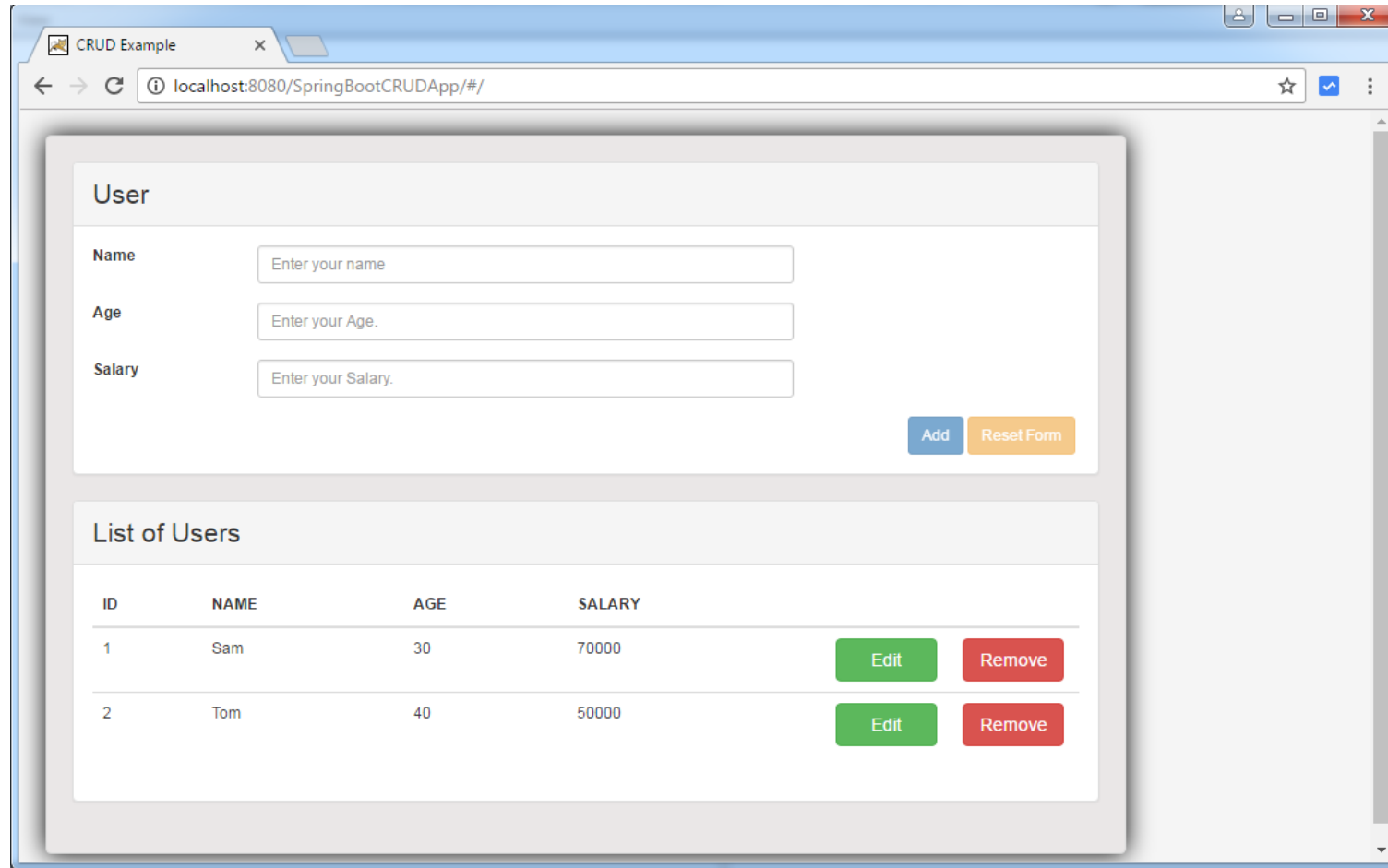
- Angular
- Vue.js
- React.js
- Thymeleaf
- Vaadin



# Príklad Web aplikácii



# Spring Boot + Angular JS



CRUD Example

localhost:8080/SpringBootCRUDApp/#/

### User

Name

Age

Salary

Add Reset Form

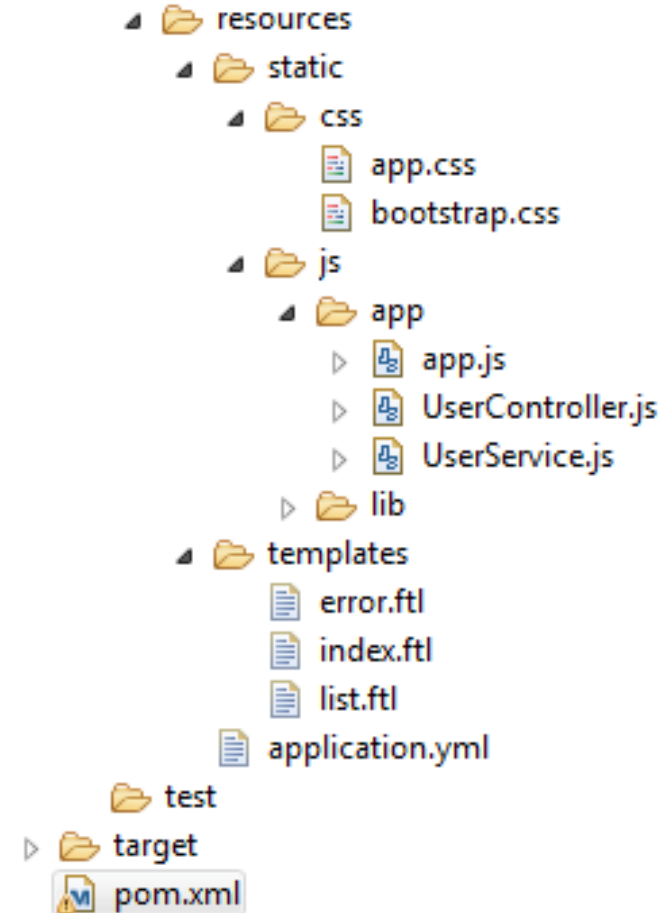
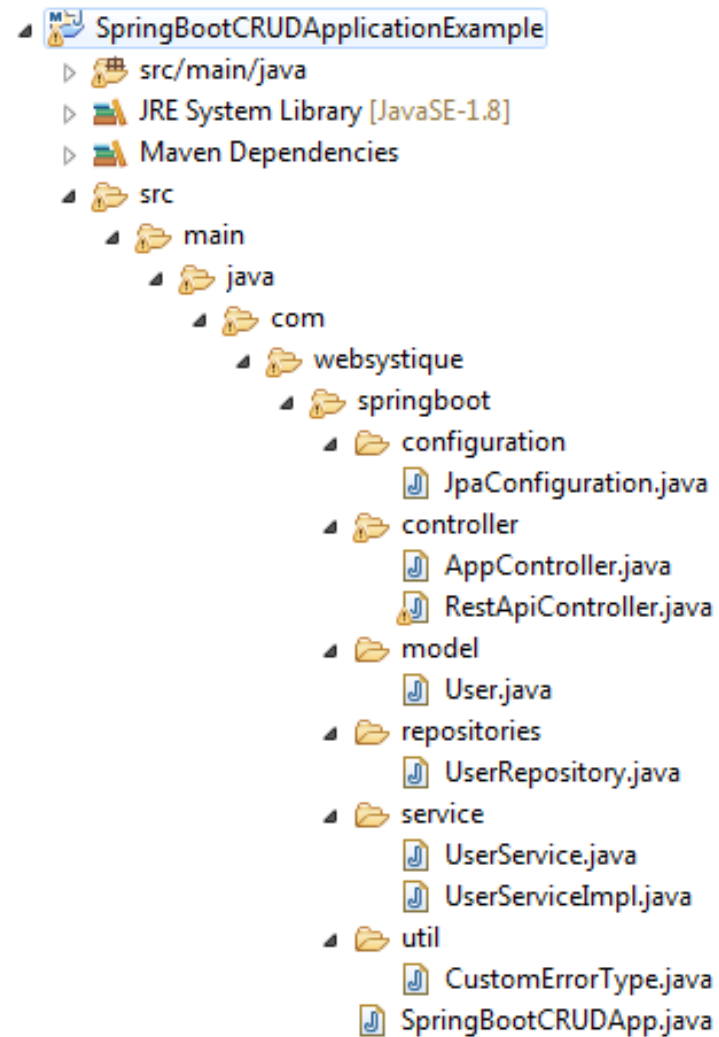
### List of Users

ID	NAME	AGE	SALARY	
1	Sam	30	70000	<button>Edit</button> <button>Remove</button>
2	Tom	40	50000	<button>Edit</button> <button>Remove</button>





# Spring Boot + Angular JS



# Spring Boot + Thymeleaf

Studentlist

localhost:8080

Students

Name	Email	Actions
Raja C	raj@gmail.com	<div>Delete</div>
Manoj K	manoj@gmail.com	
Praveen P	prav@gmail.com	
Sekar S	sekar@gmail.com	

Add Student

Add Student

localhost:8080/add

New student

Rajasekar

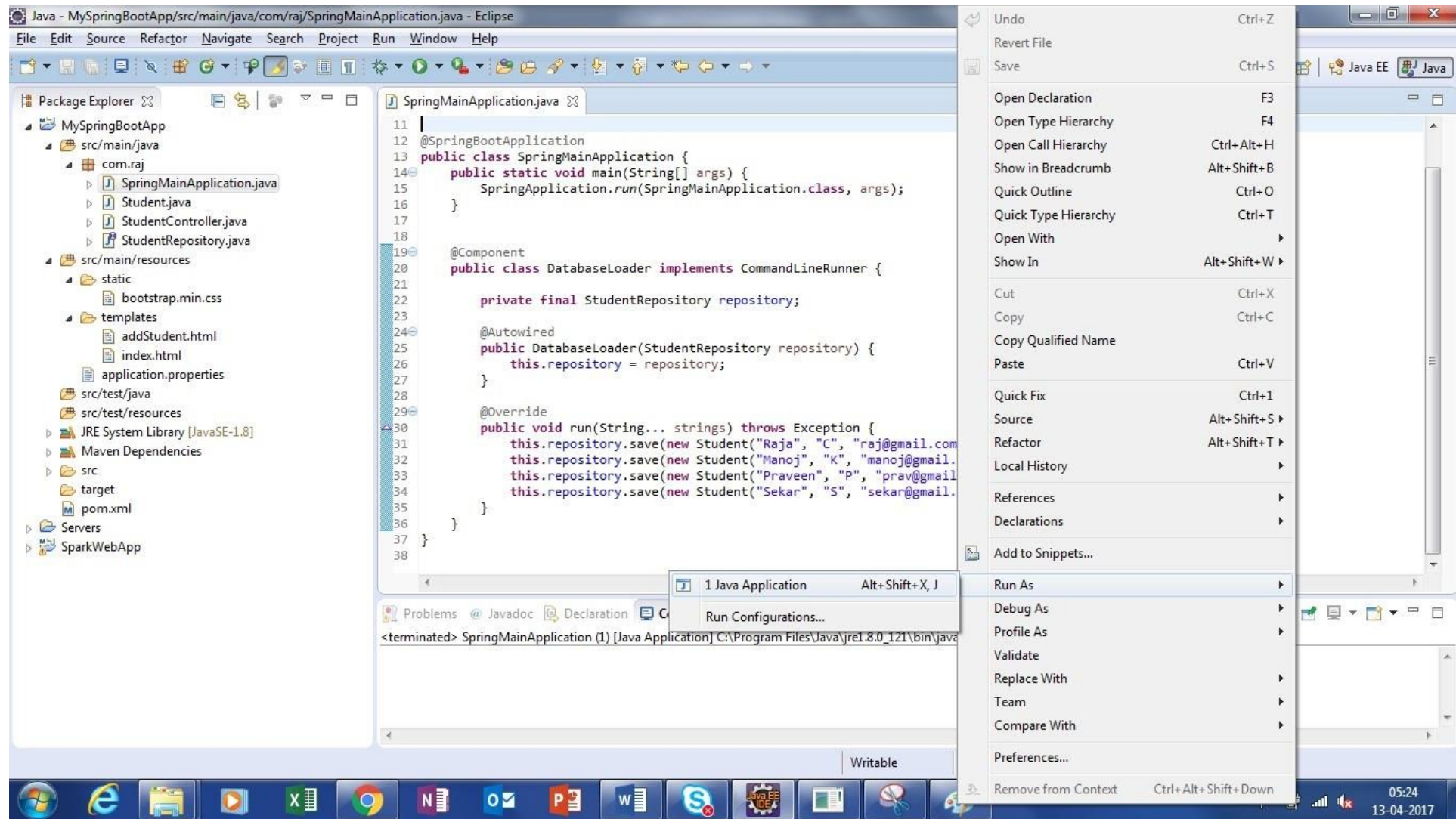
C

sekar@gmail.com

Save



# Spring Boot + Thymeleaf



# Thymeleaf 3.X.X

- Podpora starších templátov
- Úprava konfigurácie
- Podpora HTML5
- Možnosť výberu XML alebo HTML



<https://www.thymeleaf.org/doc/articles/thymeleaf3migration.html>



# Spring Boot + Thymeleaf s Bootstrap

## AdminLTE



# Spring Boot + Thymeleaf s Bootstrap

Gentella Alela!



<https://colorlib.com/polygon/gentella/index.html>



ŽILINSKÁ UNIVERZITA V ŽILINE  
Fakulta riadenia  
a informatiky



# Spring Boot Security

## *Best Practices*



# Používanie HTTPS

Možné problémy:

- TLS/SSL certifikáty – drahé
- Protokol HTTPS - pomalé





# Používanie HTTPS

## Riešenie:

- Aplikovanie HTTPS
- Let's Encrypt

## V Spring Boot:

- Zabezpečenie pripojenia aplikácie
- **WebSecurityConfigurerAdapter**

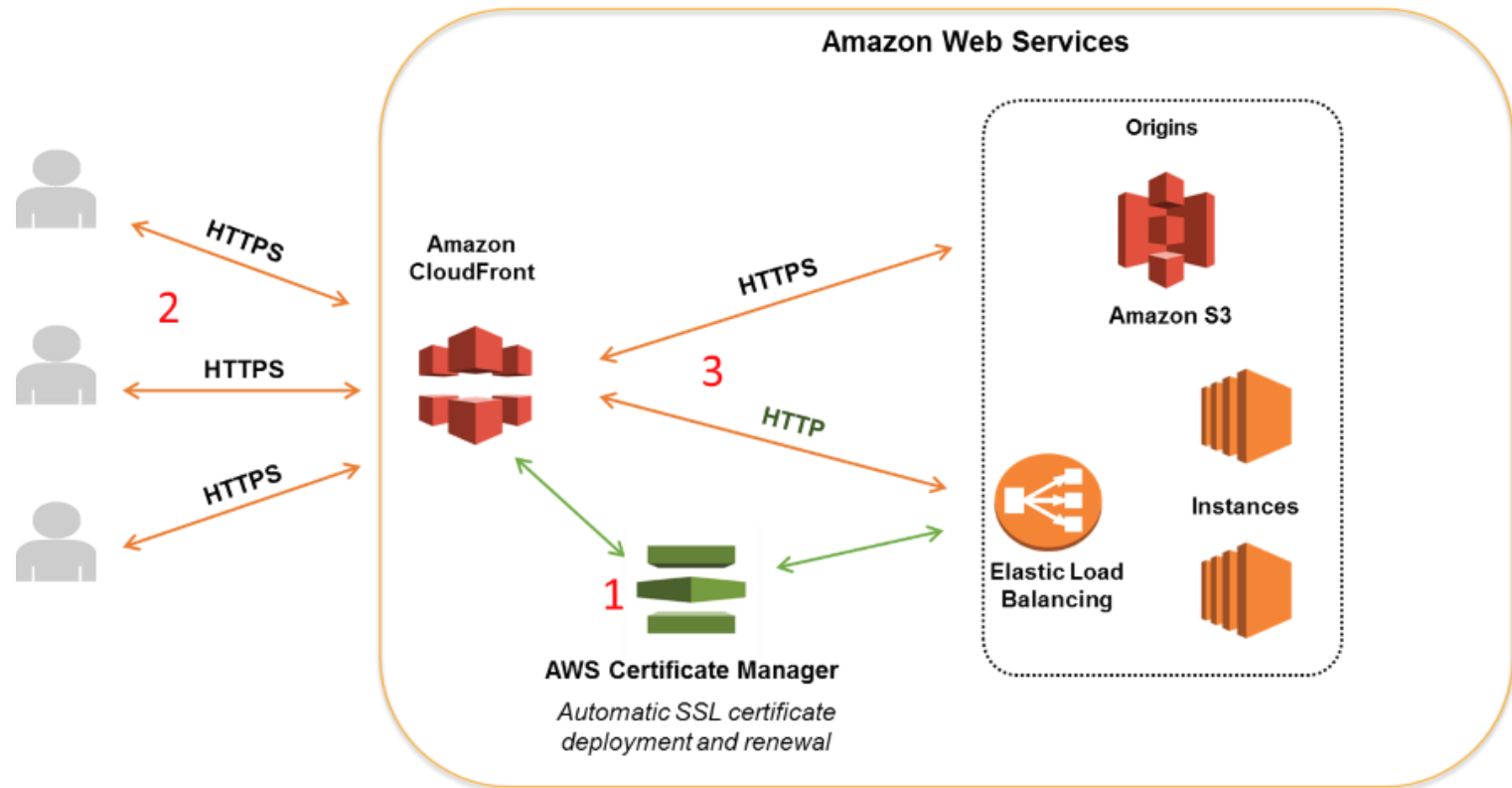
```
@Configuration
public class WebSecurityConfig extends
WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.requiresChannel().requiresSecure();
    }
}
```



# Správa SSL

- Nutné generovanie nových SSL
- Amazon Certificate Manager
  - Jednoduchá správa
  - podpora Let's Encrypt
  - Žiadna konfigurácia



- Konfigurácia automatické manažmentu iných služieb
  - Napr: HEROKU : <https://www.heroku.com>
  - Testovanie cloudu
  - <https://devcenter.heroku.com/articles/automated-certificate-management>



# Testovanie použitých Dependencies

- Vieme aké Dependencies využívame?
- Koľko je ich reálne využitých?
- Útoky najmä na OpenSource
- Eliminovať riziko známych chýb
- Kontrola:
  - Snyk testy
  - <https://snyk.io/test/>

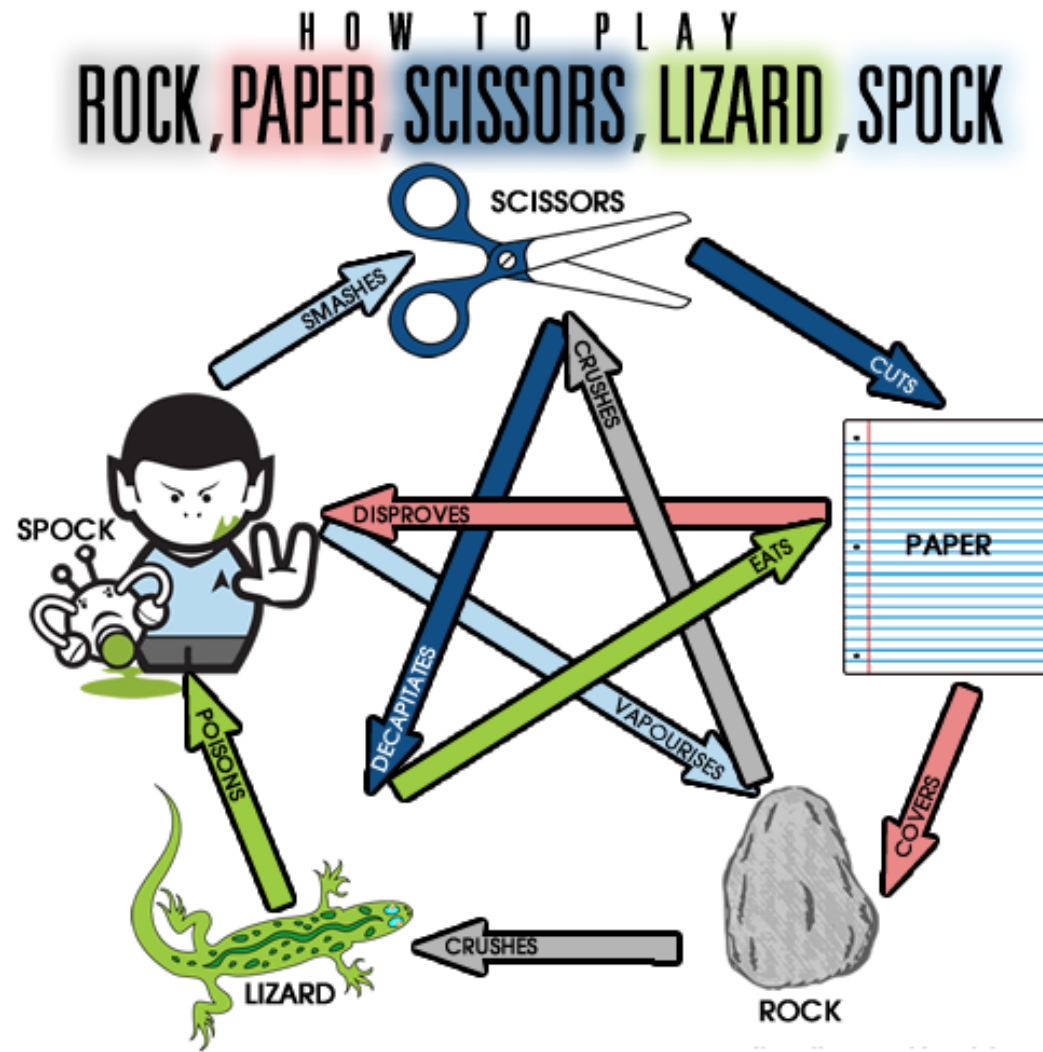


All vulnerable projects [See all projects](#)

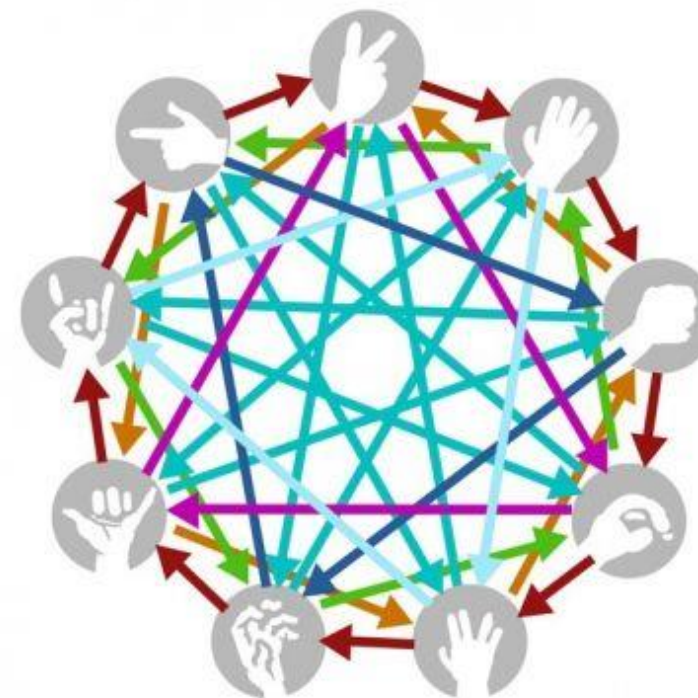
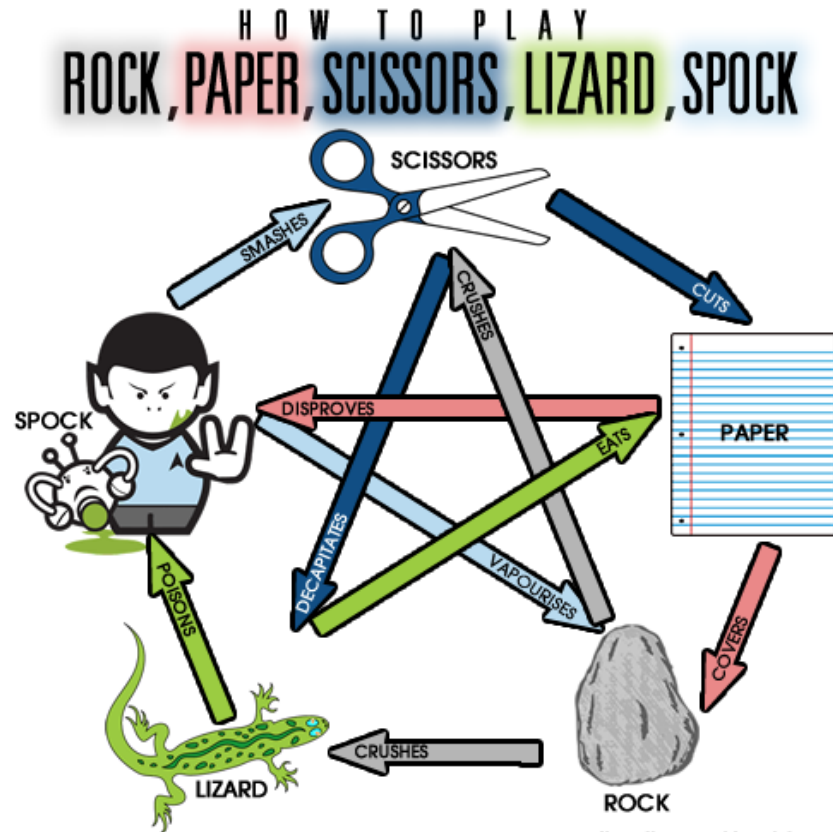
<b>M</b> <a href="#">sjmaple/java-goof:todoist-web-struts/pom.xml</a>
17 H 22 M 2 L Updated a day ago
Dependencies: 47 • Source:  GitHub
<b>M</b> <a href="#">sjmaple/spring.goof:pom.xml</a>
6 H 5 M 0 L Updated a day ago
Dependencies: 61 • Source:  GitHub
<b>M</b> <a href="#">sjmaple/java-goof:todoist-web-common/pom.xml</a>
3 H 4 M 0 L Updated 8 hours ago
Dependencies: 34 • Source:  GitHub



# Testovanie použitých Dependencies



# Testovanie použitých Dependencies



ROCK PAPER SCISSORS  
LIZARD SPOCK  
SPIDER-MAN BATMAN  
WIZARD GLOCK

ROCK PAPER SCISSORS SPOCK LIZARD by Sam Kass and Karen Bryla

Scissors cuts paper.  
Paper covers rock.  
Rock crushes lizard.  
Lizard poisons Spock.  
Spock zaps wizard.  
Wizard stuns Batman.  
Batman scares Spider-Man.  
Spider-Man disarms glock.  
Glock breaks rock.  
Rock interrupts wizard.  
Wizard burns paper.  
Paper disproves Spock.  
Spock befuddles Spider-Man.  
Spider-Man defeats lizard.  
Lizard confuses Batman  
(because he looks like Killer Croc).  
Batman dismantles scissors.  
Scissors cut wizard.  
Wizard transforms lizard.  
Lizard eats paper.  
Paper jams glock.  
Glock kills Batman's mom.  
Batman explodes rock.  
Rock crushes scissors.  
Scissors decapitates lizard.  
Lizard is too small for glock.  
Glock shoots Spock.  
Spock vaporizes rock.  
Rock knocks out Spider-Man.  
Spider-Man rips paper.  
Paper delays Batman.  
Batman hangs Spock.  
Spock smashes scissors.  
Scissors cut Spider-Man.  
Spider-Man annoys wizard.  
Wizard melts glock.  
Glock dents scissors.

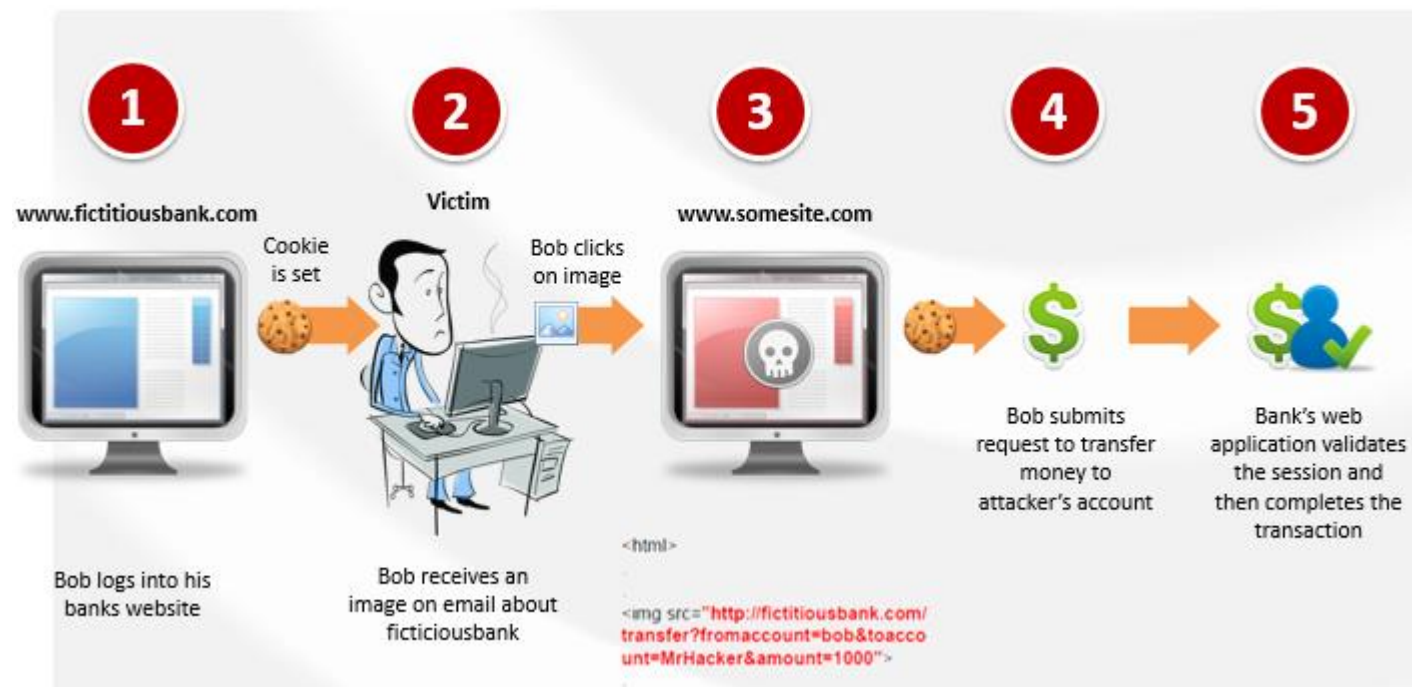


# Povolenie CSRF ochrany

## Cross-site request forgery CSRF

[https://www.owasp.org/index.php/Cross-Site\\_Request\\_Forgery\\_\(CSRF\)](https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF))

- Útok na web aplikáciu
- Obeť je ale používateľ web aplikácie



# Povolenie CSRF ochrany

- výborná podpora CSRF (*Cross-Site Request Forgery*) v Spring
- Automatické pridávanie skrytého poľa
  - Spring MVC tag: `<form:form>`
  - Thymeleaf + `@EnableWebSecurity`
- Pri Angular/React
  - Konfigurácia **CookieCsrfTokenRepository**





# CSRF vs Spring

## Používanie XSRF- TOKEN

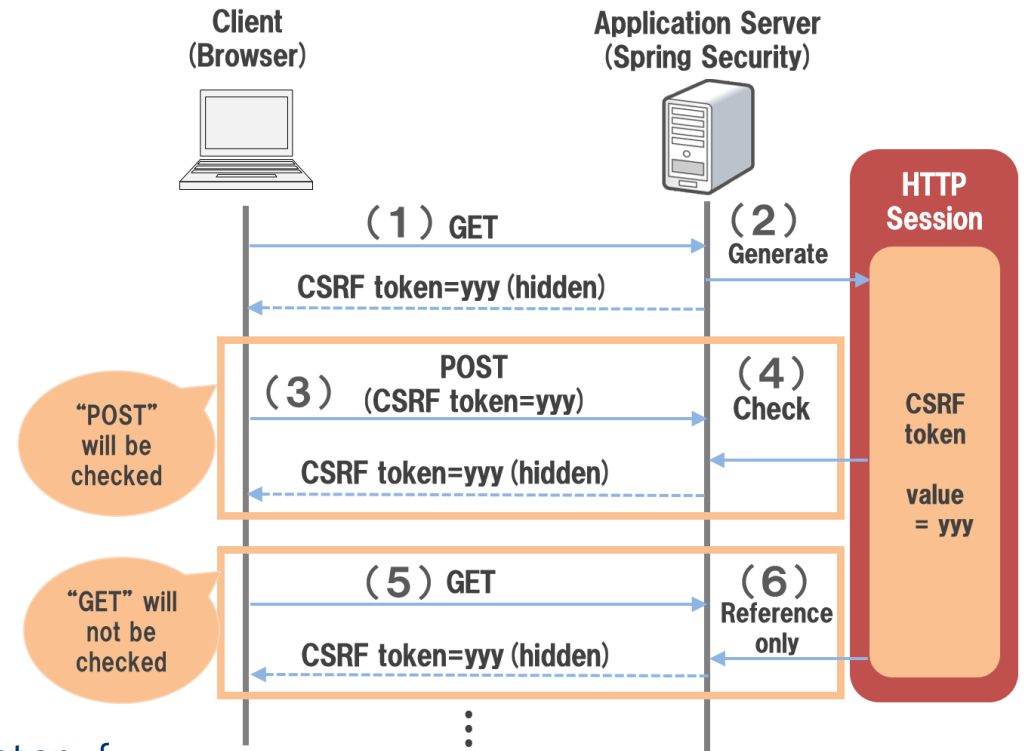
### SameSite=strict

- Spring nepoužíva pre CSRF Cookies
- <https://www.owasp.org/index.php/SameSite>
- Spring Session, WebFlux - OK

```
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .csrf()

            .csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse());
    }
}
```





# Použitie CSP na ochranu pre XSS

## Cross-site scripting (XSS)

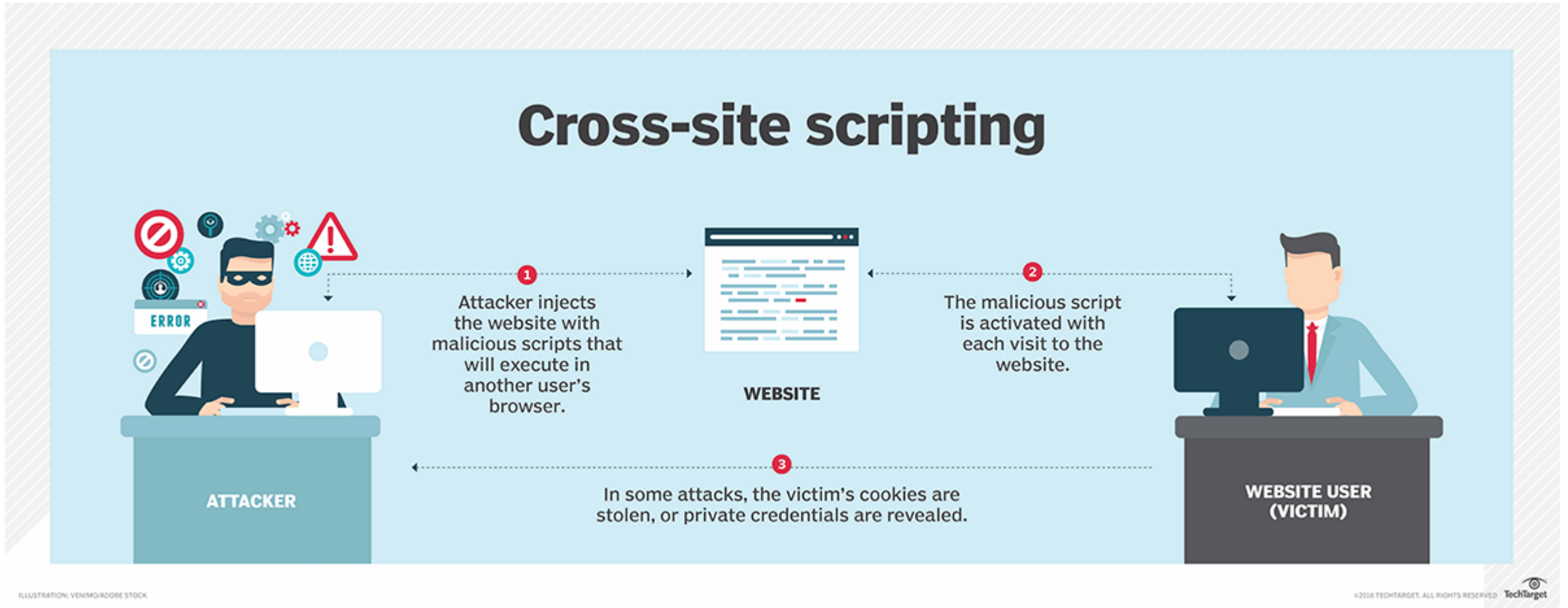
- Využitie bezpečnostných chýb v skriptoch
- Útok vlastným JS, CSS, fontom, obrázkom na
  - vzhľad
  - funkcionality
  - získanie citlivých údajov

## Content Security Policy (CSP)

- Zabránenie načítavaniu nežiadúcich zdrojov
- „whitelisting“ zdrojov
- Pridávanie hlavičky do odpovede servera



# Cross-site scripting (XSS)



# Spring Boot vs CSP

- Nie je pridávaná predvolene v Spring-u
- Povolenie CSP hlavičiek konfiguráciou:

```
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.headers()
            .contentSecurityPolicy("script-src 'self' https://trustedscripts.example.com;
object-src https://trustedplugins.example.com; report-uri /csp-report-endpoint/");
    }
}
```



# Autentifikácia s OpenID Connect

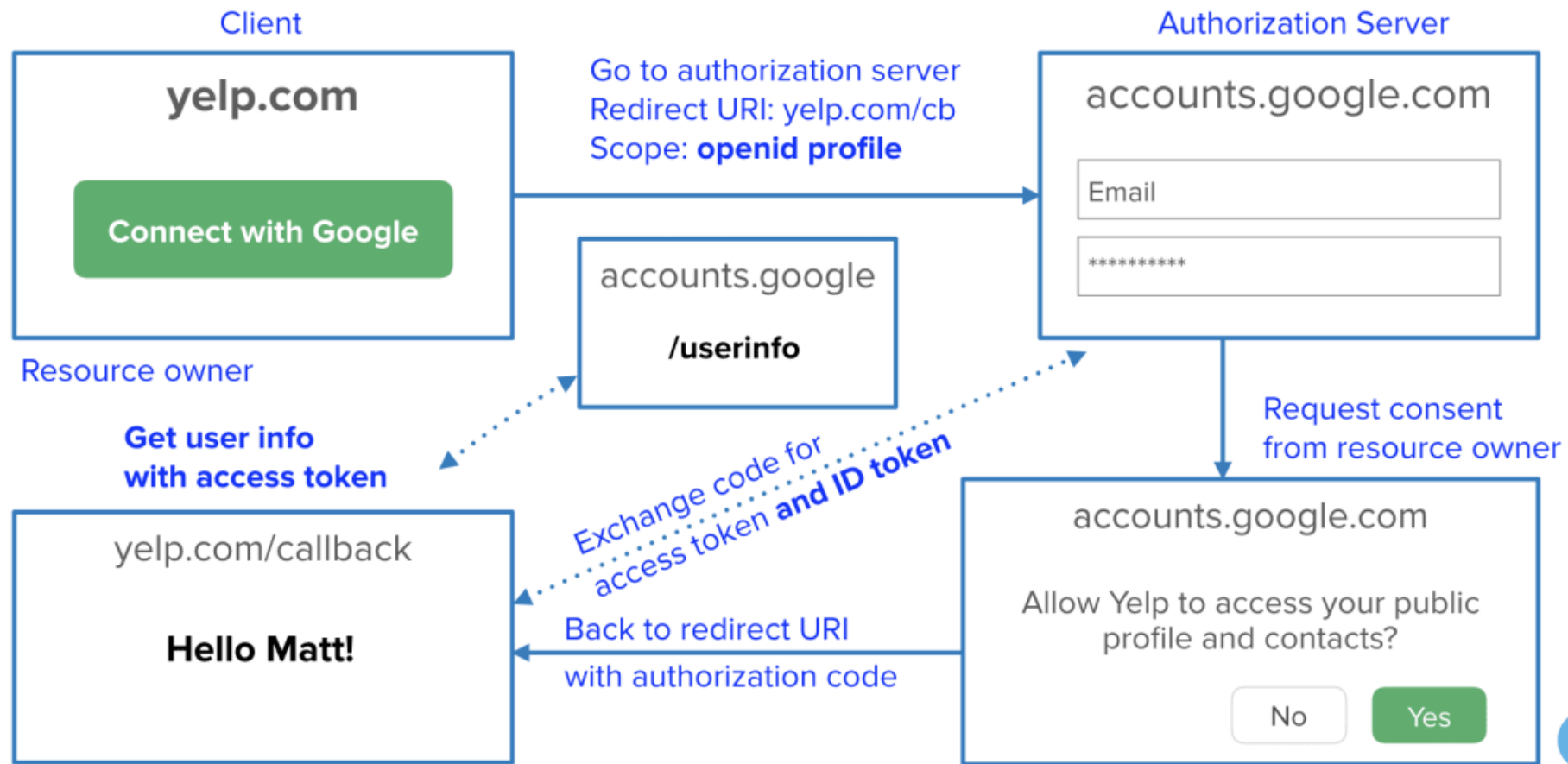
## OpenID Connect (OIDC)

- Overenie používateľa z externých zdrojov
- Pridanie endpointu **/userinfo**
- OAuth 2.0 protokol
  - <https://oauth.net/2/>



# Proces overenia OIDC

## OIDC Authorization Code Flow



# Použitie hashovania hesiel

- Ukladanie ako „obyčajný text“
  - Najhoršia vec pre bezpečnosť aplikácie
- Spring Security
  - Nie je povolené ukladanie „plain text“ hesiel
  - Použitie krypto modulu Spring-u
    - Symetrické šifrovanie
    - Generácia kľúčov
    - Hašovanie hesiel – aj dekodovanie



# Spring krypto modul

- Použitie **PasswordEncoder**

- <https://docs.spring.io/spring-security/site/docs/3.1.x/reference/crypto.html>

```
public interface PasswordEncoder {  
    String encode(String rawPassword);  
    boolean matches(String rawPassword, String encodedPassword);  
}
```

- Spring implementácie:

- **BCryptPasswordEncoder**
  - **Pdkdf2PasswordEncoder**
  - .....



# Používanie posledného Releasu

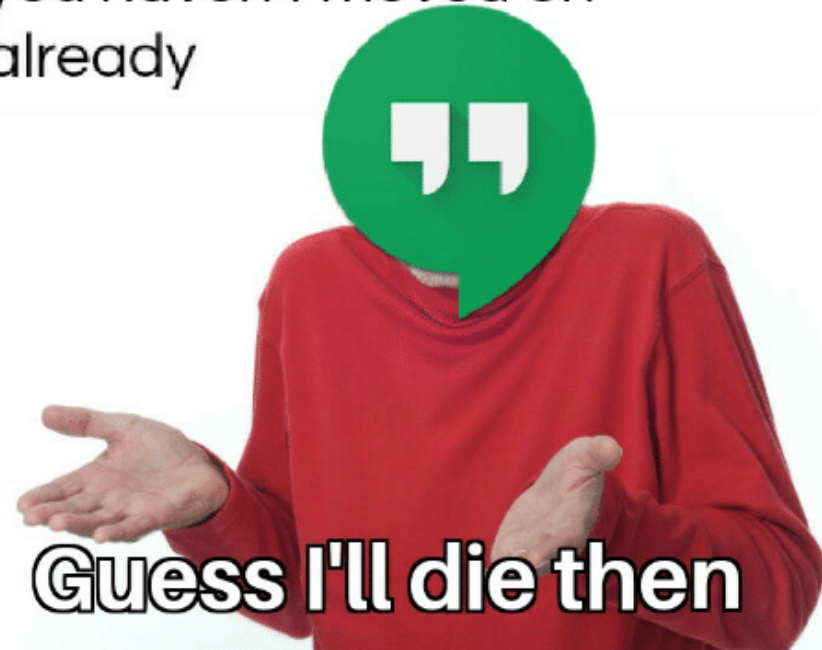
- Viacero dôvodov na používanie posledného vydania (?)
  - Bezpečnosť
  - [start.spring.io](https://start.spring.io)
    - Najnovšie verzie Spring-u
    - Aktuálne Dependencies
- Spätná kompatibilita medzi verziami
- Rušivé aktualizácie
- Ak zistím chybu v bezpečnosti ?
  - 1. aktualizácia
  - 2. záplata chyby „patchom“
  - 3. ignorovanie 😊
- Vývoj aplikácie na chybnom základe – otvorenie app útočníkom





# Používanie posledného Releasu

[Update: Google statement]  
2019 is your last year to use  
Google Hangouts 'classic' if  
you haven't moved on  
already



Sub to "quote" series....



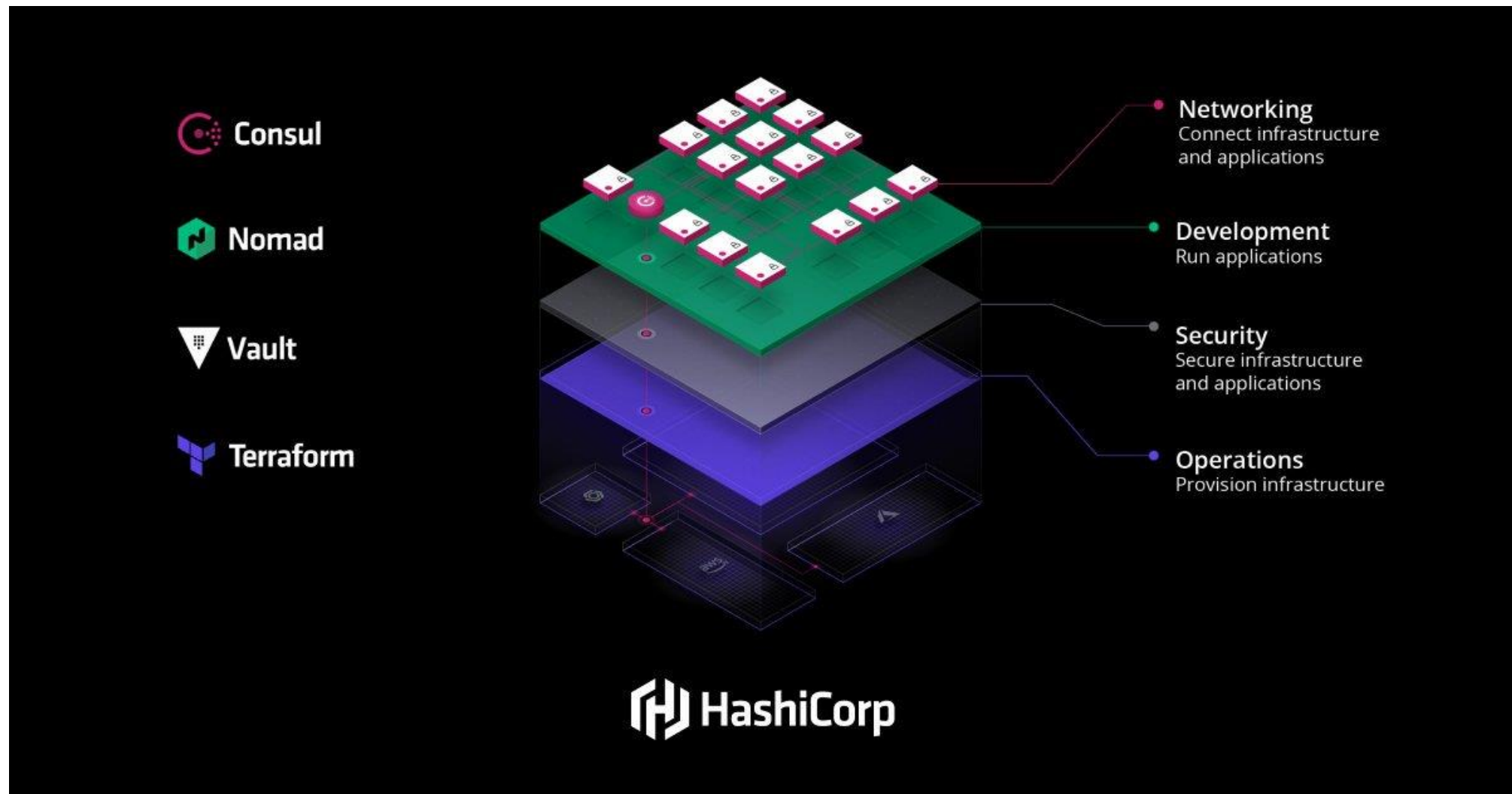
# Bezpečné ukladanie citlivých údajov

- Citlivé údaje:
  - Heslá
  - Prístupové tokeny
  - ...
- Nepoužívať „plain text“
- Nepoučiteľný vývojári – GitHub
- Ukladanie v „trezore“
  - Služby s manažovaným prístupom, povereniami
    - HashiCorp <https://www.vaultproject.io/>
  - Mechanizmus autentifikácie napr. LDAP
  - Použitie Spring Vault
    - <https://projects.spring.io/spring-vault/>
    - Jednoduchý prístup vďaka anotáciám:

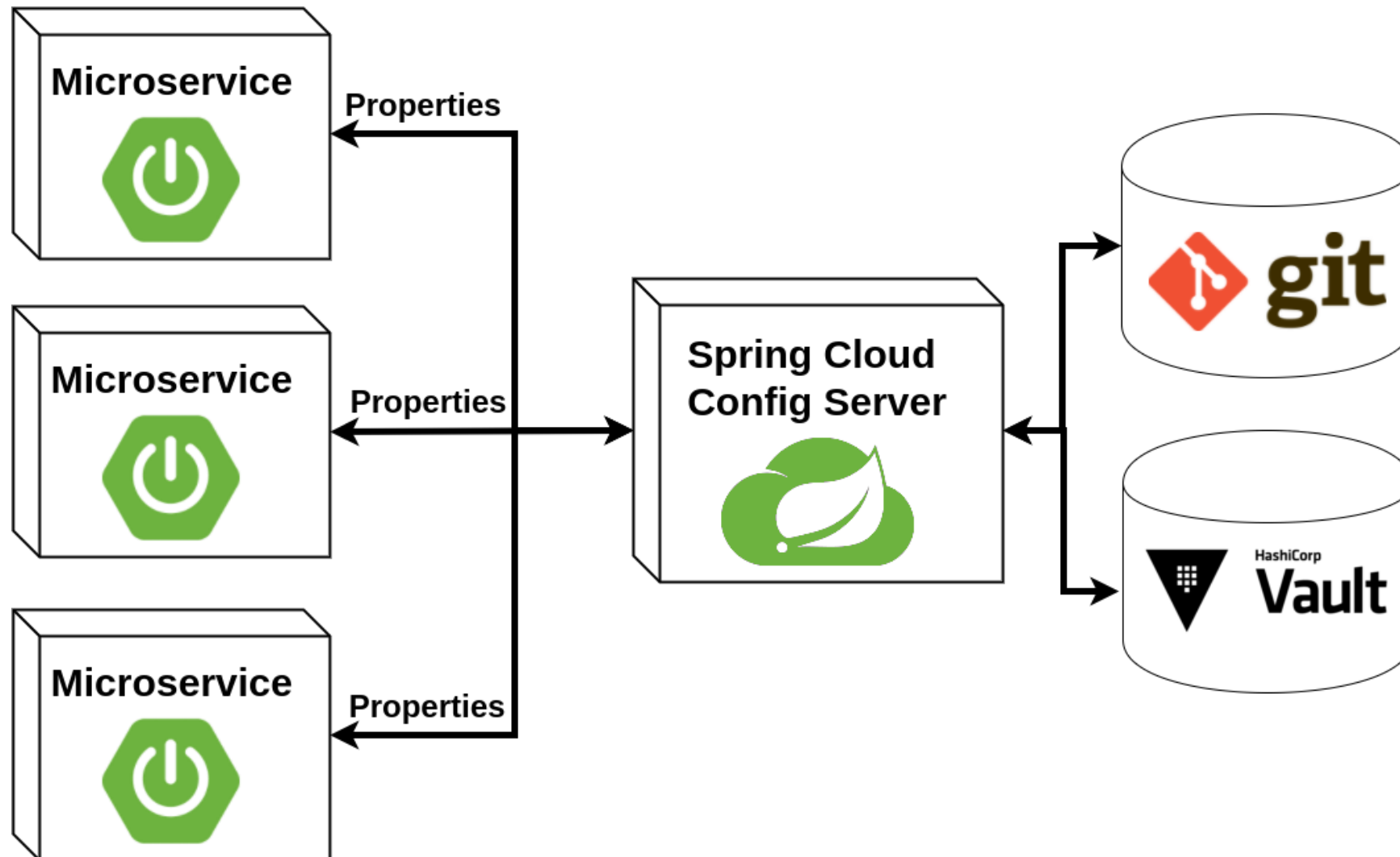
```
@Value("${password}")  
String password;
```



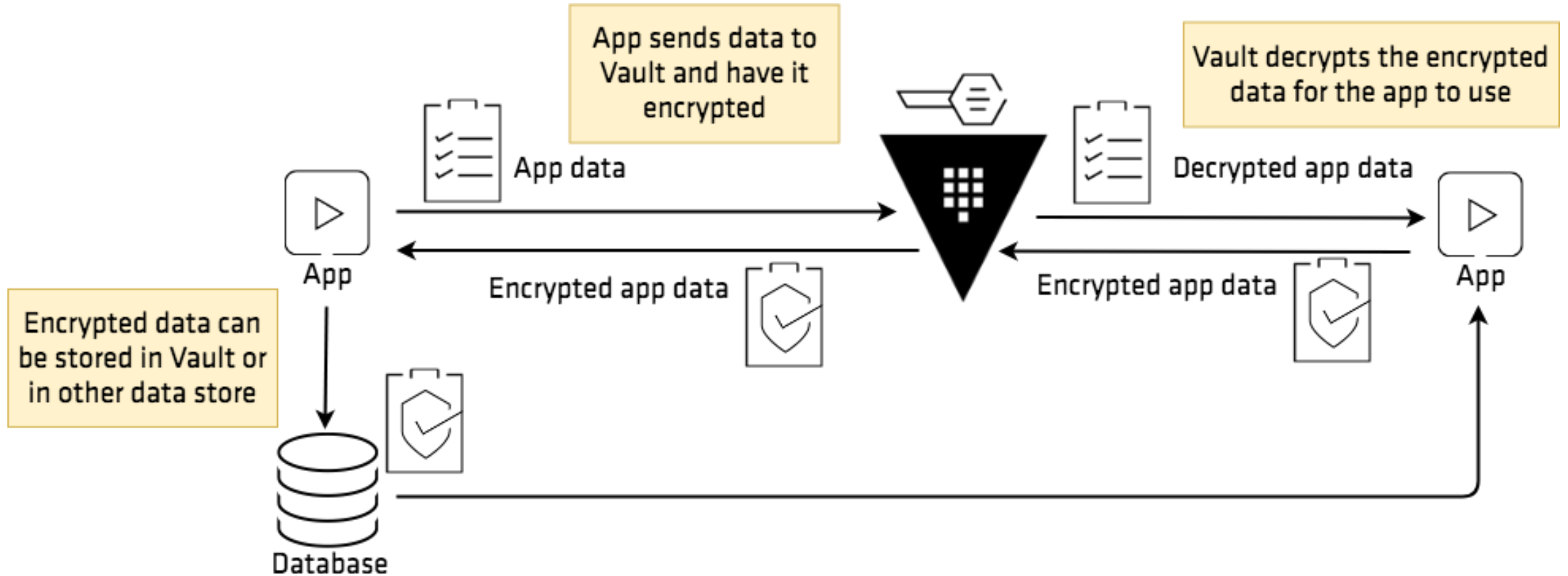
# Spring Vault



# Aplikácia Spring Vault

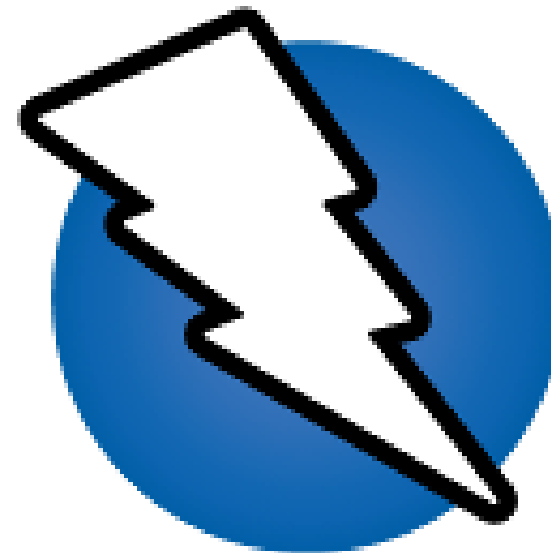


# Spring Vault proces



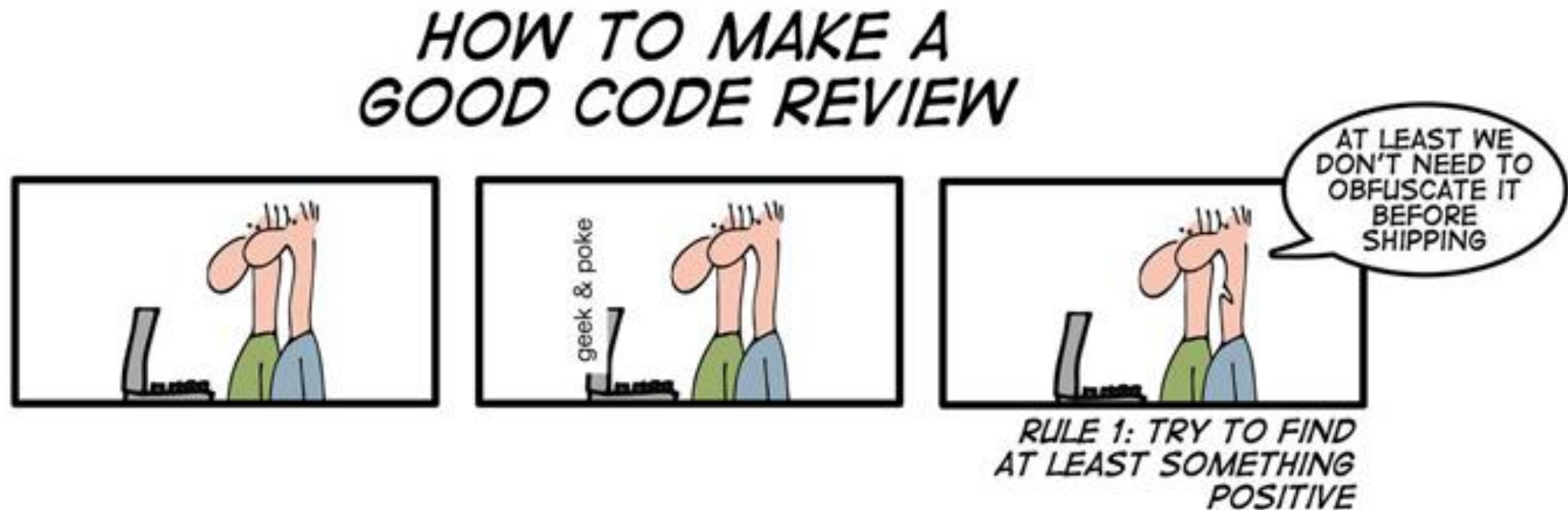
# Penetračné testy aplikácie

- Testovanie, simulácia možných útokov
  - zvonku, zvnútra
- OWASP ZAP
  - Bezpečnostný nástroj na „živé“ aplikácie
  - Open source projekt
  - [https://owasp.org/www-project-developer-guide/draft/verification/tools/zed\\_attack\\_proxy/](https://owasp.org/www-project-developer-guide/draft/verification/tools/zed_attack_proxy/)
  - <https://www.zaproxy.org/>
  - Múd testovania:
    - Spider
      - Mapovanie všetkých odkazov aplikácie
    - Aktívne skenovanie
      - Automatické testovanie vybraných cieľov z listu potencionálnych zraniteľností
  - Vypracovanie reportu

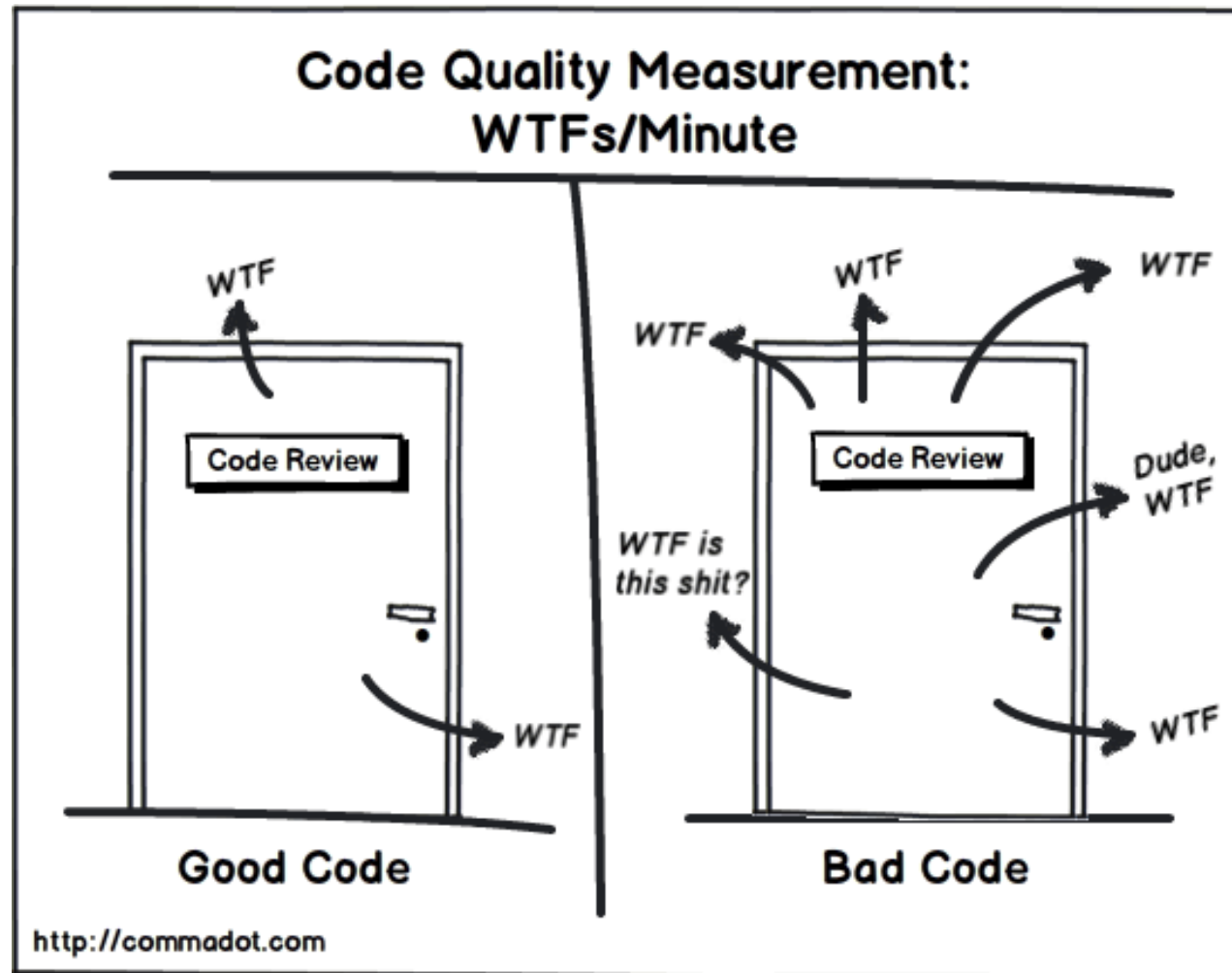


# Bezpečnostný tím a revízia kódu

Dneska už nevyhnutnosť každého vývojárskeho tímu



# Bezpečnostný tím a revízia kódu







**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