NNPIA LAB 01: Inicializace Spring Boot projektu

Cílem prvního cvičení je inicializace backendové aplikace postavené na frameworku Spring Boot. Studenti si projdou proces založení projektu, konfigurace prostředí a vytvoření prvního REST endpointu. Výstupem je projekt, který
lze dále rozšiřovat stejně jako v produkčních aplikacích.

Předpoklady

- JDK 21 nebo novější
  - Na školních počítačích je nainstalována verze JDK 17. Můžete ale použít funkci IntelliJ IDEA pro stažení a
    nastavení JDK 21 přímo v IDE (File → Project Structure → SDKs → + → JDK).
- IntelliJ IDEA
  - Doporučena je verze **IntelliJ IDEA Ultimate**
    - Na školíních počítačích je nainstalovoána verze 2023.3.3.
    - Na osobních počítačích je doporučováno využít verzi 2025.2.3 nebo novější.
  - Studenti mohou využít **bezplatnou studentskou licenci**
- Verzovací systém Git
- **Webový prohlížeč** ideálně postavený na jádru Chromium
  - Google Chrome, Microsoft Edge, Brave…
- **HTTP klient pro testování API**
  - Postman
  - Insomnia

*V rámci cvičení bude použit **Gradle**, který je součástí projektu vytvořeného pomocí **Spring Initializr** (tento nástroj generuje projekt s vybraným build systémem automaticky).*

1.1. Inicializace Spring Boot projektu

V současné době již není nutné inicializovat backendovou aplikaci ručně vytvářením jednotlivých souborů od nuly. Spring
Initializr, umožňuje vývojářům projít procesem založení projektu krok za krokem a automaticky připravit základní
strukturu aplikace.

1. Spusťte **IntelliJ IDEA** a vytvořte nový prázdný projekt z verzovacího systému pomocí tlačítka `Get from VCS`.

2. Do pole URL vložte odkaz na repozitář, který byl vytvořen při připojování do NNPIA - GitHub Classroom.

💡 *Případně můžete využít vlastní repozitář.*

3. V dalším kroku budete vyzváni k výběru autentizační metody. Zvolte tu, která Vám nejlépe vyhovuje.

💡 *Na školních počítačích je doporučen token.*

4. Vytvořte novou složku s názvem `doc` a zkopírujte do ní tento soubor `LAB01.md`.
5. Otevřete webový prohlížeč a přejděte na nástroj Spring Initializr.
6. Nastavte následující parametry pro vytvoření nového Spring Boot projektu:
    - Project : Gradle - Groovy
    - Language : Java
    - Spring Boot : 4.0.2 nebo novější
    - Group : cz.upce.fei.nnpia
    - Artifact : název projektu (např. `backend`)
    - Packaging : Jar
    - Java : 21
    - Dependencies : Spring Web
7. Vygenerujte projekt a stáhněte výsledný archiv.
8. Rozbalte stažený archiv do kořenového adresáře projektu tak, aby výsledná složka byla pojmenována `backend`.

💡 *Inicializaci Spring Boot aplikace je možné provést i v IntelliJ. Můžete si prozkoumat možnost File → New module →*
Spring Boot.

9. Ve složce `backend` otevřete soubor `build.gradle` pomocí IntelliJ IDEA. Klikněte pravým tlačítkem myši na soubor a
vyberte možnost **Link Gradle Project**.
    - V pravé části IDE se zobrazí nová ikonka **Gradle** (s logem slona).
    - Ve spodní části IDE se zobrazí loadbar průběh importu Gradle projektu.

💡 *Součástí importu projektu je i stahování závislostí (knihoven třetích stran). Ty si je možné prohlédnout v souboru*
`build.gradle` *v sekci* `dependencies`.

❗ Pokud se Vám během importu objeví chybové hlášky s nekompatibilní verzí javy. Například:

```
Could not resolve all dependencies for configuration ':compileClasspath'.
Failed to calculate the value of task ':compileJava' property 'javaCompiler'.
Cannot find a Java installation on your machine ... matching: {languageVersion=21,
vendor=any vendor, implementation=vendor-specific, nativeImageCapable=false}. Toolchain
download repositories have not been configured.
```

Zkontrolujte zdali máte shodující se verzi Javy uvedenout v `build.gradle` a v nastavení IDE (File → Project Structure →
Project → Project SDK).
Pokud ne, přepněte verzi Javy v IDE na požadovanou (v tomto případě JDK 21) a stáhněte si ji pokud ji ještě nemáte
nainstalovanou.
Poté znovu proveďte import pomocí ikonky v pravé části IDE **Gradle → Sync All Gradle Projects**.

💡 *Tato situace typicky nastane na školních počítačích kde je naisntalována Java 17.*

10. Vyčkejte, než IntelliJ IDEA dokončí import Gradle projektu a stáhne všechny závislosti.
11. Struktura projektu by nyní měla vypadat takto:

```
- nnpia-assignments-stXXXXX
    - .idea
    - .git
    - doc
    - backend
        - .gradle
        - .build
        - src
            - main
                - java
                    - cz
                        - upce
                            - fei
                                - nnpia
                                    - BackendApplication.java
                - resources
                    - application.properties
        - build.gradle
        - settings.gradle
```

12. Nyní je načase spustit backend. To je možno udělat několika způsoby:
    - Otevřete třídu `BackendApplication.java`, klikněte pravým tlačítkem myši do editoru a vyberte možnost **Run '
      BackendApplication.main()'**.
    - V pravé části IDE otevřete záložku **Gradle → Tasks → application → bootRun** a spusťte úlohu dvojklikem.
    - Otevřete terminál nebo CMD v kořenovém adresáři projektu a spusťte příkaz:
        - `./gradlew bootRun` (Linux/Mac)
        - `gradlew.bat bootRun` (Windows)

```
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v4.0.2)

...

2026-02-04T19:11:28.509+01:00  INFO 21218 --- [NNPIA assignments backend] [
main] f.n.b.NnpiaAssignmentsBackendApplication : Started
NnpiaAssignmentsBackendApplication in 0.623 seconds (process running for 0.766)
```

💡 *Vyzkoušejte postupně všechny možnosti. V případě prvních dvou si všimněte že se Vám v horní části IntelliJ objeví*
nové run konfigurace.

💡 *Spring aplikace implicitně naslouchá na portu 8080.*

❗ Pokud se Vám při spuštění aplikace zobrazí chybová hláška indikující že port 8080 je již obsazen. Může se jednat o
jiný proces využívající tento port nebo jste aplikaci spustili vícekrát. Pamatujte že Spring aplikace není klasický
algoritmus který skončí po vykonání, ale běží stále dokud ho nevypnete.

```
***************************
APPLICATION FAILED TO START
***************************

Description:

Web server failed to start. Port 8080 was already in use.
```

13. Otevřete webový prohlížeč a ověřte, že aplikace běží na adrese http://localhost:8080. Měli byste vidět následující
    chybovou stránku:

```
Whitelabel Error Page
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Feb 04 19:19:05 CET 2026
There was an unexpected error (type=Not Found, status=404).
```

💡 *Tato chybová stránka je výchozí chování Spring aplikace, pokud není definován žádný endpoint pro zpracování požadavků. Je to očekávané chování. V další kapitole to opravíme.*

1.2. Vytvoření prvního endpointu

Endpointy jsou vstupní branou pro komunikaci klientů s webovou aplikací. Kód každého endpointu je obvykle umístěn v tzv.
kontrolerech. Slouží ke zpracování požadavků, které přicházejí na specifické URL adresy, a k odesílání odpovědí zpět
klientům. Při vytváření endpointů je využit standard který se jmenuje REST (Representational State Transfer), který je
postaven na principech HTTP protokolu.
Ten definuje sadu pravidel/best practice jak mají být endpointy navrhovány.

Spring Boot Cvičení

1. Vytvořte nový balíček `controller` v adresáři `src/main/java/cz/upce/fei/nnpia/backend`.
2. V balíčku `controller` vytvořte novou třídu `HealthController.java`.
3. Třídu označte anotací `@RestController` a `@RequestMapping("/api/v1/health")`.
4. Přidejte do třídy metodu `healthCheck`, která bude zpracovávat GET požadavky na endpoint `/api/v1/health`.
    - Metoda bude vracet `ResponseEntity<String>`.
    - Nebude přijímat žádné parametry.
    - Metodu označte anotací `@GetMapping`.
    - To těla implementujte následující kód: `ResponseEntity.ok("Service is running.");`
5. Restartujte aplikaci a ověřte funkčnost nového endpointu zadáním URL adresy http://localhost:8080/api/v1/health.

1.3. Příprava AI asistenta

Pro řešení dobrovolných úkolů bude primárně využíván **GitHub Copilot**, dostupný zdarma pro studenty. Studenti však
mohou použít **jakýkoliv volně dostupný AI nástroj**.

Vibe coding Dobrovolný úkol

1. Pokud **ještě nemáte GitHub účet**, vytvořte si ho na stránce GitHub – Join.
2. K bezplatnému využívání GitHub Copilot Pro je nutné mít aktivní studentskou licenci (GitHub Education), která je
   zdarma. Nicméně je potřeba splnit následující podmínky:
    - Musíte být studentem s platným školním e-mailem přidaným do GitHub účtu a ideálně i studentskou kartou.
        - V průběhu ověřování studentského statusu ji budete muset nahrát jako důkaz. Stačí i forka z mobilní
          aplikace.
    - Musíte mít aktivovanou dvoufaktorovou autentizaci (2FA) na svém GitHub účtu.
    - Musíte mít vyplněné jméno a příjmení ve svém GitHub profilu shodné s údaji na studentské kartě.

    - Musíte mít vyplněné platné fakturační údaje ve svém GitHub účtu.
        - **Aktivace studentské licence GitHub Copilot nevyžaduje žádné platební údaje. Pokud po Vás bude webová stránka vyžadovat zadání například platební karty, nepostupujte dále!**
3. Ověřte nebo aktivujte **GitHub Student Developer Pack**.
    - Navštivte stránku GitHub Student Developer Pack a klikněte na tlačítko **Get your pack**.
    - Postupujte podle instrukcí pro ověření svého studentského statusu.
4. V **IntelliJ IDEA** přejděte do *Settings/Preferences → Plugins → Marketplace* a vyhledejte a nainstalujte plugin ** GitHub Copilot **
5. Po instalaci pluginu se v IDE **přihlaste ke svému GitHub účtu** a ověřte, že je Copilot aktivní.
    - Při používání editoru by se měly objevovat návrhy kódu generované Copilotem.
    - V dolní části IDE by měla být viditelná ikona **GitHub Copilot**, po kliknutí by se měl zobrazit status `ready`.
    - V pravé části IDE by měla být viditelná ikona **GitHub Copilot Chat**, která poskytuje možnost konverzace s AI asistentem (podobně jako ChatGPT).

1.4. Logování

Vibe coding Dobrovolný úkol

Logování slouží k zaznamenávání důležitých informací o běhu aplikace, jako jsou chybové stavy nebo průběh zpracování požadavků.
V reálném světě aplikace běží v různých prostředích, typicky v neprodukčním prostředí (vývoj, testování) a v produkčním prostředí, kde aplikaci používají koncoví uživatelé.
Produkce je pro zákazníky kritická a nedá se ladit přímo – logy se tak stávají hlavním (a často jediným) zdrojem informací.

1. Otevřete **GitHub Copilot Chat** a pomocí funkce **Ask** zjistěte, jaké **úrovně logování (log levels)** jsou běžně dostupné v aplikacích postavených na Spring Bootu.
    - Zaměřte se na význam jednotlivých úrovní.
    - Zjistěte, jaká úroveň je výchozí pro Spring Boot aplikace.
    - Zjistěte, jaké jsou typické hodnoty pro vývojové, testovací a produkční prostředí.
2. Pomocí funkce **Agent** nechte AI asistenta upravit root level projektu na `DEBUG`.
    - Agent by si měl sám sestavit plán kroků pa následně je provést.

    - Vaším úkolem je validovat navržené změny a potvrdit jejich aplikaci.
    - Restartujte aplikaci a ověřte, že se při startu loguje více informací než předtím.
3. Otevřete soubor `application.properties` a postupně změňte hodnotu nové vlastnosti na: `WARN` `ERROR` a `TRACE`.
    - Při každé změně restartujte aplikaci.
    - Sledujte, jak se mění množství logovaných informací při spuštění aplikace.

💡 *Pro rychlé otevření souboru `application.properties` můžete v IntelliJ využít dvojité stisknutí **Shift** a začít psát název souboru.*

Odevzdání

Commity pushněte do repozitáře na GitHub Classroom. Pokud ještě nejste součástí classroomu, využijte tento odkaz pro
připojení:
NNPIA - GitHub Classroom.

- Po dokončení všech úkolů vytvořte **commit** se všemi provedenými změnami a **pushněte jej do vzdáleného repozitáře**.
- Název commitu musí **začínat označením LAB01** a obsahovat **stručný popis změn**.

```
LAB01 – Inicializace Spring Boot projektu

Vytvoření health endpointu.
```

Užitečné odkazy a zdroje

- Understanding logging in Spring Boot