Samostatný projekt z Multimediálních systémů (ver. 2020/A)

Pokud vytváříte projekt na systému Windows užívejte v cestách zpětná lomítka (\), u systému na bázi Unixu (Linux nebo i macOS) používejte klasická lomítka (/). **Před odevzdáním však cesty upravte pro použití v systému Windows.** Děkuji.

Odevzdání projektu

Projekt odevzdejte v souboru **.zip**. Jiné formáty na elearning nelze nahrát. Soubor zip bude pojmenován podle **VAŠEHO ID**!

Bude-li odevzdaný soubor pojmenován jinak, je automaticky hodnocen 0b!

Obsah souboru zip bude složka **main**, která se v projektu nachází ve složce **src** (stejně jako domácí úkoly).

Struktura bude následující:

<ID>.zip – main – java – balíček se soubory tříd
– templates – složky se soubory šablon a dalšími potřebnými soubory pro běh šablon

---

Vytvoření projektu

Připravte si pomocí inicializátoru projektu (https://start.spring.io/) projekt s následujícími parametry:

NASTAVENÍ	HODNOTA
PROJECT	Gradle Project
LANGUAGE	Java
SPRING BOOT	2.3.5. (nebo hodnotu, která bude automaticky vybrána)
GROUP	cz.vutbr.feec.utko.bpcmds
ARTIFACT	projekt
NAME	projekt
DESCRIPTION	Samostatny projekt
PACKAGE NAME	cz.vutbr.feec.utko.bpcmds.projekt
PACKAGING	Jar
JAVA	11

V rámci výběru závislostí (Dependencies) vyberte:

- **Spring Web**
- **Thymeleaf**

Vygenerovaný projekt stáhněte a pokračujte v IDE IntelliJ IDEA.

Zadání vypracování projektu

Všechnu logiku webového serveru umísťujte do nové třídy **WebController.java**. Případné metody **handlerů nebo servletů** do tříd jim určeným.

Implementujte kód v jazyce Java, který rozšíří funkcionalitu serveru o podporu poskytování adaptivního streamu. Implementaci omezte pouze pro adaptivní stream typu **MPEG-DASH**.

Vytvořte třídu komponenty s názvem **ProjectResourceComponent**, která bude rozšiřovat třídu **ResourceHttpRequestHandler**. V této třídě vytvořte metodu **getResource()**, která bude rozšiřovat pomocí anotace @Override stejnojmennou metodu z třídy **ResourceHttpRequestHandler. (1b)**

Ve třídě **WebController.java** vytvořte metodu s názvem **streaming**(). Tato metoda bude naslouchat na adresách „**/dash/{file}**“, kde **{file}** značí proměnnou složku adresy, kde bude umístěn dotazovaný

soubor. Celá metoda bude poskytovat data adaptivního streamu typu MPEG-DASH. Celá metoda bude využívat globální proměnnou **DASH_DIRECTORY**, ve které bude definována cesta ke kořenové složce adaptivního streamu typu MPEG-DASH. Metoda bude také reagovat pouze na požadavek typu **GET**. **(1b)**

Dále vytvořte metodu **index()**, naslouchající na adrese „**/index**“ (http://localhost:8080/index), která bude poskytovat stejnojmennou šablonu (**index.html**). V šabloně bude umístěn element s přehrávačem, který bude přehrávat lokální stream uložený v počítači.

V šabloně bude „napevno“ definován zdroj adaptivního streamu:

`http://localhost:8080/dash/manifest.mpd`

V šabloně využijte JS knihovnu Dash.JS, která je pro přehrávání adaptivního streamu MPEG-DASH nezbytná. Pro knihovnu vytvořte samostatnou složku s názvem „**js**“. **(1b)**

Přehrávač bude obsahovat volbu přehrávaného profilu. (viz domácí úkol č. 3) Soubory potřebné pro plnou implementaci přehrávače jsou umístěny v sekci zadání projektu. **(1b)**

Po splnění všech předchozích kroků bude stránka vypadat následovně:

U přehrávače je důležité mít zobrazené profily videa. Nezobrazení profilů audia **není** bráno jako chyba. Přehrávač **nebude** po načtení stránky automaticky spouštět přehrávání. **Pro testování využívejte pouze soubory z elearningu!**

V projektu dále implementujte funkcionalitu, která bude vyhledávat adaptivní streamy typu MPEG-DASH v lokálním úložišti serveru Spring.

Podobně jako u cvičení č. 6 vyhledejte dostupné adaptivní streamy typu MPEG-DASH a vypište je na webovou stránku ve formátu textového seznamu (není třeba vytvářet náhledové obrázky).

Začněte vytvořením třídy **Video**, která bude obsahovat informace o zdroji adaptivního streamu. Třída bude obsahovat objekt **File** s názvem **file**, ve kterém bude uložen samotný hlavní soubor adaptivního streamu. Druhý objekt typu **String** s názvem **file_name**, bude obsahovat název konkrétního streamu. V třídě **budou přítomny tzv. gettery a settery** objektů. Oba přítomné objekty budou **privátní**. **K hodnotám přistupujte skrze „getter“**. Objekt nastavujte pomocí konstruktoru. **(0,5b)**

Dále vytvořte třídu **VideoLibrary**, která bude procházet zadaný adresář a bude hledat soubory adaptivních streamů. V třídě vytvořte metodu **discoverFiles**, která bude (stejně jako v projektu cvičení č. 6) vracet kolekci objektů typu **Path**.
Kolekci s objekty **Path** přetypujte v metodě **scanFiles()** a dále vyfiltrujte potřebné soubory v metodě **getFiles()**. Metoda **getFiles()** bude vracet list typu **ArrayList** s objekty typu **Video**.
Tedy při zavolání metody **getFiles(**<cesta ke složce>**)** mi bude vrácen list, který bude obsahovat **všechny** nalezené adaptivní streamy typu MPEG-DASH. **(2,5b)**

TIP: Pro detekci adaptivního streamu se zaměřte na manifest soubory s příponou „**.mpd**“.

*Při generování seznamu si **dejte pozor** na proměnnou **file_name**. Ukládejte do ní jedinečný identifikátor (například cestu od kořenové složky uvedené v globální proměnné **DASH_DIRECTORY**), tak aby bylo možné při požadavku od klienta jednoznačně určit, o který stream se jedná.*

TIP: Pro získání názvu složky, ve které se soubor nachází, můžete využít:

```
file.getParentFile().getName()
```

Ve třídě **WebController** vytvořte novou metodu s názvem **videoCollection()**, která bude naslouchat na adrese „**/videocollection**“ (http://localhost:8080/videocollection) a bude vracet stejnojmennou šablonu **videocollection.html**.
Na stránce „videocollection“ bude umístěn textový seznam se všemi nalezenými streamy. Prvky seznamu budou klikací odkazy. Při kliknutí na odkaz bude uživatel přesměrován na novou stránku **player**. **(2b)**

Ve třídě **WebController** vytvořte novou metodu s názvem **player()**, která bude naslouchat na adrese „**/player**“ (http://localhost:8080/player) a bude vracet stejnojmennou šablonu **player.html**. Stránka bude odpovídat pouze na metodu GET (REST) a bude přebírat hodnotu adresy streamu, kterou následně vloží **pomocí MVC** (Model View Controller) modelu do šablony s přehrávačem. Šablona s přehrávačem bude opět obsahovat **přehrávač vycházející z knihovny Dash.JS a bude možné v ovládacím panelu přepínat profily přehrávaného streamu**. **(1b)**

Po splnění všech předchozích kroků bude stránka /videocollection vypadat následovně:

Video
DASH-BIG_BUCK_BUNNY\manifest.mpd
DASH-SINTEL\manifest.mpd
DASH-TEARS_OF_STEEL\manifest.mpd

Za předpokladu, že v kořenové složce, kde jsou uložené streamy jsou tři podsložky:

DASH-BIG_BUCK_BUNNY
DASH_SINTEL
DASH_TEARS_OF_STEEL

Každá obsahuje jeden adaptivní stream s názvem manifestu „manifest.mpd“.

Při kliknutí na odkaz se zobrazí stránka s přehrávačem. Viz první obrázek.

Konec zadání

Možné problémy a jejich teoretické řešení

1) Uvědomte si, že budete přistupovat k více streamům skrze jednu stránku. Bude nutné upravit adresy, na kterých metoda **streaming()** naslouchá. Metodu upravte tak, aby byla zachována funkčnost i první části projektu.
2) Nenalezena třída FilenameUtils: implementujte do **build.gradle** následující implementaci:

```
implementation 'org.apache.commons:commons-io:1.3.2'
```

   A v třídě použijte import:

```
import org.apache.commons.io.FilenameUtils;
```

3) Vkládání dat z modelu MVC do proměnné v JavaScriptu:

```
url = 'http://localhost:8080/dash/[[${name}]]';
```

Hodnocení projektu

Projekt bude hodnocen jak po celkové funkční stránce, tak i po dílčích úkolech. Celkově je možné z projektu získat 10b. Body jsou v projektu rozloženy následovně:

1. část

1b – vytvoření komponenty **ProjectResourceComponent** a její funkčnost
1b – metoda **streaming()**, její základní funkčnost definovaná na první straně
1b – metoda **index()** a její funkčnost ověřená výslednou stránkou v prohlížeči
1b – možnost výběru profilu videa streamu

2. část

0,5b – vytvoření třídy **video** pro ukládání informací o streamu
2,5b – vytvoření třídy **videolibrary** s prohledáním adresáře, vyfiltrováním potřebných souborů a následné vrácení **ArrayListu** s objekty **video** obsahující informace o nalezených streamech (pouze MPEG-DASH)
2b – vytvoření metody **videoCollection()** generující webovou stránku z šablony se seznamem nalezených streamů
1b – vytvoření metody **player()** generující webovou stránku z šablony s přehrávačem streamu získaného z odkazu stránky se seznamem nalezených pomocí metody

Celkově 10b