Alkalmazásfejlesztés - I
ZH

SZTE Szoftverfejlesztés Tanszék

2024. tavaszi félév

1. Általános tudnivalók

A feladat megoldását **Bíró**-ban kell beadni! A feladat kézzel lesz kiértékelve, **de csak az a megoldás ami bíróban hiba nélkül fordul.** Több forduló beadás esetén a **legutolsó beadást értékeljük ki!** A fordításhoz használt rendszer a **Maven**, melyet egy **wrapper**-en keresztül használunk, **ezt elő kell állítani**. A Bíró Linux-on fordít. A feltöltött feladat **nem** tartalmazhat **.class** fájlokat és **nem** lehet benne **target** mappa. A feladatot zip csomagként kell feltölteni, mely tartalmazza a teljes projektet `mvn clean` utáni állapottal.

A ZH során használható anyag a
`https://biro.inf.u-szeged.hu/kozos/alkfejl/alkalmazasfejlesztes_kozos.zip`
linken letölthető, mely egy zip fájl és tartalmazza a

- Oktatási weboldal off-line változatát
- JavaFX13 dokumentációt
- Adatbázis segédletet

Technikai részletek

- bíró fordítási parancs: `./mvnw clean compile`
- feltöltés előtti lépés: `./mvnw clean` vagy az IDE-n belül a teljes projectre kiadott clean.
- zip elkészítés: `zip -r feladat.zip <project-mappa>`

2. Maven segítség

A biro fordításhoz kell a `mvnw` bináris, mely a

```
mvn wrapper:wrapper
```

parancs kiadásával oldható meg.
Ha ez nem található, akkor a projekt nem fordul, így nem értékelhető a megoldás!
Fontos, hogy ha **több modul**ból áll a projekt, akkor a **legkülső pom** fájl mappájában adjuk ki a Maven-es parancsokat!

3. Bíró eredmények

A kiértékelés során 0 vagy 1 pont szerezhető. Az 1 pont jelzi a sikeres feladatbeadást, mely értékelésre kerül. A 0 pont sikertelen beadást jelent, melynek az okai a következők lehetnek:

- A beadott fájl sérült, nem megfelelő zip fájl vagy sérült zip került feltöltésre.
- A beadott zip tartalmaz **target** mappát.
- A beadott zip tartalmaz **.class** (java bináris) fájlt.
- A beadott projekt a fordítás során hibával leállt.

A Maven fordítás eredménye a *mvn.compile* fájlban található.

A ZH során REST végpontokat kell megvalósítani `HttpServlet`ek segítségével.

Adatmodellek

Videojatek:

- cim: string
- ar: int
- fejleszto: string
- besorolas: int (6, 12, 16 vagy 18 karikás)

Core API használat

A `VideojatekController` osztályt kell használni és a `Videojatek` modell osztályt! Az osztályok használatát dokumentáció segíti. A webapi modulban a resources alatti *application.properties* fájlal adhatók meg a konfigurációk. *sqlite* érték esetében a *jdbc:sqlite:* előtag az abszolút útvonalról elhagyható, de nem probléma ha ott van. Egyéb érték nem fogadható el, csakis classpathban megadott osztály fully qualified neve.

Listázás

(13 pont) A végpont leírása:

- **method**: *GET*
- **útvonalak**: */api/list*, */alista*
- **GET-paraméterek**:
  - *fejleszto*: a keresett videojatek fejlesztője, nem kötelező. Az adatbázisban a fejlesztők nevébe nem kell szóközt írni, az adatbázisban is így szerepelnek. Pl. RidleyScott, RockstairGames
  - *besorolas*: a keresett videojatek besorolása.
- **cookie-k kezelése**:
  - Egy *request-number* nevű cookie-t állít be, ami nyomon követi a kérések számát.
- **status**: *200* OK, ha a kérés sikeresen feldolgozásra került.

A servlet a megadott szűrőparaméterek alapján keres a videojatekek között, és a találatokat JSON formátumban adja vissza. Ha a *besorolas* paraméter nem a [6, 12, 16, 18] számok valamelyike, akkor nincs figyelembe véve a szűrés során.

Hozzáadás

(13 pont) A végpont leírása:

- **method**: *POST*
- **útvonal**: */api/add*
- **bemeneti adatformátum**: *application/json*
- **bemenet leírása**: JSON formátumban vár egy *Videojatek* objektumot, ami tartalmazza a videojatek minden releváns adatát.
- **cookie-k kezelése**:
  - A kéréssel érkező összes cookie értékét módosítja az "modified" előtag hozzáadásával, és visszaállítja a felhasználó böngészőjébe.
- **status kódok**:
  - *201* Created, ha a videojatek sikeresen hozzáadásra került.
  - *400* Bad Request, ha a beérkezett JSON formátum hibás vagy nem olvasható.
  - *500* Internal Server Error, ha a videojatek hozzáadása egyéb okból nem sikerült.

A végpont a *Videojatek* objektumok hozzáadásáért felelős. A kérés tartalmát JSON formátumban értelmezi, és ha a JSON szintaktikailag helyes, megpróbálja hozzáadni a videojatekot az adatbázishoz. A művelet sikerességétől függően állítja be a válasz HTTP státuszkódját.

Frissítés

(12 pont) A végpont a *Videojatek* objektumok besorolásának módosításáért felelős. A végpont leírása:

- **method**: *PUT*
- **útvonal**: */api/update*
- **bemenet leírása**: JSON formátumban vár egy *Videojatek* objektumot. Az objektumnak csak a címét és besorolását kell felhasználni!
- **status kódok**:
  - *200* **OK** Ha megtalálta a cím alapján a videojatekot és sikeresen módosította a besorolást.
  - *400* **Bad Request**, ha a beérkezett JSON formátum hibás vagy nem olvasható vagy a besorolás nem a [6, 12, 16, 18] számok valamelyike.
  - *500* **Internal Server Error**, ha a videojatek módosítása egyéb okból nem sikerült.

3.1. Törlés

(12 pont) A végpont a *Videojatek* objektumok besorolásának törléséért felelős. A végpont leírása:

- **method**: *DELETE*
- **útvonal**: */api/delete/**
- **bemenet leírása**: A törlés a *Videojatek* címe alapján történjen!
- **status kódok**:
  - *200* **OK** Ha megtalálta a cím alapján a videojatekot és sikeresen törölte.
  - *400* **Bad Request**, ha nincs megadva videojatekcím.
  - *500* **Internal Server Error**, ha a videojatek törlése egyéb okból nem sikerült.