L2: Java webalkalmazások

- A laborfeladatok továbbra is használják az első feladatban előírt projekttematikát (ld. `ubb-idde-lab1-build.pdf`).
- Továbbra is feltétel, hogy ne legyenek *statikus kódelemző jelzések*, illetve hogy helyesen fusson le a `check` taszk (ld. korábbi laborfeladatok leírása).
- Az alábbi leírások általánosan érvényesek mindenkire, de mindenki alkalmazza őket azokra az entitásokra/funkcionalitásokra, melyek már léteznek a korábbi laborról. Ha valahol az általános leírás nem tisztán alkalmazható a meglévő projektre, kérdezzétek a laboránsokat tisztázásért.

Web modul

- Készítsünk egy harmadik alprojektet a Gradle multimodul projektünkben: `<diakid>-web`.
  - A projekt függjön a backend projekttől.
  - A példák alapján ez a modul használja a `war` plugint, kimeneti artifactje egy konvencióknak megfelelő WAR állomány, melyet kitelepíthetünk egy Java webkonténerbe. Opcionálisan konfigurálhatjuk a példaprogramokban definiált `deploy`/`undeploy` taskokat is ebben a modulban.
- A modul tartalmazzon egy `Servlet` technológián alapuló web frontendet az első laborfeladaton elkészített alkalmazásnak. Implementáljuk *legalább* az alábbiakat.

Servlet 1: API JSON-nal

- Minden metódus használjon JSON-t, mint a kérés és válasz body-k formátuma. Használjuk egy ehhez megfelelő külső könyvtárat, amelyet a megfelelő gradle állományban behúzunk függőségként.
- A `Servlet` neve és elérési útvonala reflektálja az entitás nevét, pl. `GET /menus`, `GET /cars`, stb.
- `GET` hívásra visszatéríti az összes tárolt entitást, tömb formájában.
- Ha adott egy `id` nevű *request paraméter* a fenti `GET` híváskor, akkor csak az adott paraméterrel rendelkező példányt térítsük vissza. Ha ilyen példány nem létezik, `404 Not Found` státuszkódot térítsünk vissza.
- Ugyanezen servletre intézett `POST` kéréssel tudjunk beszúrni egy entitást a memóriában tárolt adatbázisunkba. Ha a body-ból hiányzik bármely kötelező tulajdonság, vagy nem megfelelő típusú, térítsünk vissza `400 Bad Request` státuszkódot megfelelő üzenettel.

- `DELETE` esetén töröljük az `id` request paraméterű entitást.
- `PUT` esetén módosítsuk az `id` request paraméterű entitást a body-ban megadott értékre. Nem talált entitás esetén küldjünk `404 Not Found`-ot.
- Teszteljük Postmannel az összes CRUD metódust.

Servlet 2: Sablonmotor

- Készítsünk egy második servletet, amely `GET` hívások esetén jelenítsen meg egy **sablonmotorral felépített HTML oldalt**. Ez jelenítse meg olvashatóan formázva az entitásokat (használjunk egy statikus CSS állományt a formázáshoz). Használhatunk bármilyen sablonmotrot–Handlebars, Thymeleaf, FreeMarker, JSP.

Filter

- Implementáljunk egy minimális hitelesítési rendszert (elegendő egyetlen hard-code-olt felhasználó-jelszó párost használni). Az egyetlen helyes felhasználónév és jelszót állítsuk be a login servlet tulajdonságaként (lehet konstans vagy Servlet init property).
- Készítsünk egy szűrőt (`Filter`), mely átirányít egy bejelentkezési oldalra, ha még nem vagyunk bejelentkezve. Alkalmazzuk a szűrőt a 2-es Servletünk útvonalára.
- A bejelentkezési oldal kérjen be egy felhasználónév és jelszó párost. Helyes értékek esetén tekintsük a felhasználót bejelentkezettnek (tároljuk *session*ben ezt az információt).
- A sablonnal generált entitáslistázó oldal tartalmazzon egy kijelentkezési gombot, melynek kattintása törli a sessiont (ld. szerveroldali session `invalidate` metódus).

Web modul tesztelése

- A `gradle war` parancs előállít egy WAR erőforrást, melyet bemásolhatunk a Tomcat mappánk `webapps` alkönyvtárába. Egy futó Tomcat példány a 8080-as porton figyel, s automatikusan követi az új WAR állományok megjelenését.
- Mint említett fennebb, a konzolból való könnyű tesztelés érdekében konfigurálhatunk `deploy` és `undeploy` taszkokat, mint a `02-servletjsp/war-static` példában.
- IntelliJ-ből való könnyű tesztelésért telepíthetünk egy Tomcat plugint.
  - Aki rendelkezik IDEA Ultimate Edition licensszel, eléri a hivatalos Apache Tomcat plugint: `https://www.jetbrains.com/help/idea/run-debug-configuration-tomcat-server.html`
  - Egy jó ingyenes alternatíva a „*SmartTomcat*” plugin: `https://plugins.jetbrains.com/plugin/9492-smart-tomcat/`

Feltöltés

- A feladatot töltsük fel a saját git tárolónkra, egy **dedikált ágra**. Az ág neve tartalmazza a laborfeladat sorszámát, pl. `lab2`. További információk a `ubb-idde-lab0-setup.pdf` állományban.
- Hozzunk létre egy **merge request**et, amely tartalmazza a laborfeladat számát, állítsuk a merge requestet a jelen feladathoz tartozó **csoportos milestone-ra**, majd linkjét **adjuk le Canvasen**. Végső leadási időpontnak tekintjük az utolsó commit, az utolsó push, és a Canvasre való linkfeltöltés közül a **legkésőbbi** mozzanatot.