4. SCA házi feladat

Simon Balázs (sbalazs@iit.bme.hu), BME IIT, 2015.

A továbbiakban a NEPTUN szó helyére a saját Neptun-kódot kell behelyettesíteni, csupa nagybetűkkel.

1 A feladat leírása

A feladat egy mozi jegyfoglaló rendszerének elkészítése többfajta protokoll és adatformátum támogatásával. Az alkalmazást egy SwitchYard projektben kell megvalósítani. Az összes szolgáltatásnak ebben az egy projektben kell lennie.

2 A SOAP webszolgáltatás

A szolgáltatás WSDL interfésze és működése pontosan megegyezik az első házi feladatban szereplő webszolgáltatás interfészével és működésével. Ugyanezt a szolgáltatást kell SCA-ból WSDL interfésszel és SOAP bindinggal publikálni. A publikált szolgáltatás URL-je a következő legyen:

`http://localhost:`*`port`*`/NEPTUN/CinemaService`

ahol a *port* helyén a szerver megfelelő portszáma áll. Az URL lokális részében a **NEPTUN** context-path-ként állítható be a SOAP binding tulajdonságai között. A **CinemaService** a WSDL **service** tagjének **name** attribútumából származik, így azt kell megfelelően elnevezni.

3 A REST szolgáltatás

A REST szolgáltatás segítségével székeket lehet foglalni vagy vásárolni. Ez a REST szolgáltatás az előző pontban elkészített SOAP szolgáltatás komponensére épül, de nem szükséges a SOAP protokoll használata, lehet memóriabeli Java hívásokkal is továbbítani a kéréseket. A REST szolgáltatás báziscíme a következő legyen:

`http://localhost:`*`port`*`/NEPTUN/resources/Cinema`

ahol a *port* helyén a szerver megfelelő portszáma áll. A **NEPTUN/resources** rész context-path-ként adható meg a REST binding tulajdonságai között.

Az alábbi táblázat írja le, hogy a szolgáltatásnak milyen műveleteket kell támogatnia. A táblázatban szereplő URL-ek a fenti báziscímtől relatív értendők.

HTTP metódus	Relatív URL	Paraméterek	Tevékenység
POST	**/seats**	HTTP body-ban székfoglalás vagy székvásárlás struktúra	lefoglalja vagy megvásárolja a paraméterként kapott székeket

A szolgáltatásnak XML és JSON formátumú adatokat is kell tudnia fogadni és küldeni. Az alábbi példák megadják az egyes kérések és válaszok pontos formátumát. A megadott székeket zárolni kell, majd azonnal le is kell foglalni vagy meg kell vásárolni. Eredményként a zárolás során visszakapott azonosítót kell visszaadni. Ha a SOAP webszolgáltatás kivételt dob, akkor a REST szolgáltatás HTTP 400 (Bad Request) hibával térjen vissza.

Példa 3 db szék foglalására a G6-os széktől kezdve:

POST	**http://localhost:8080/NEPTUN/resources/Cinema/seats**

Request HTTP body:	{
	"type": "reserve",
	"row": "G",
	"column": "6",
	"count": 3
	}

Response HTTP body:	{"lockid":"rti45am2"}

Példa 3 db szék foglalására a G6-os széktől kezdve:

POST	**http://localhost:8080/NEPTUN/resources/Cinema/seats**
Request HTTP body:	<seats> <type>reserve</type> <row>G</row> <column>6</column> <count>3</count> </seats>
Response HTTP body:	<result> <lockid>rti45am2</lockid> </result>

Példa 5 db szék vásárlására a J4-es széktől kezdve:

POST	**http://localhost:8080/NEPTUN/resources/Cinema/seats**

Request HTTP body:	{
	"type": "buy",
	"row": "J",
	"column": "4",
	"count": 5
	}

Response HTTP body:	{"lockid":"xz4ha2t"}

Példa 5 db szék vásárlására a J4-es széktől kezdve:

POST	**http://localhost:8080/NEPTUN/resources/Cinema/seats**
Request HTTP body:	<seats> <type>buy</type> <row>J</row> <column>4</column> <count>5</count> </seats>
Response HTTP body:	<result> <lockid>xz4ha2t</lockid> </result>

4 A JMS szolgáltatás

A **NEPTUN** nevű JMS üzenetsorba az előző pontban definiált JSON üzeneteket lehet küldeni. A JMS szolgáltatásnak ezeket a JSON üzeneteket kell feldolgoznia az előző pontban definiált módon. A zárolás azonosítóját azonban nem kell visszaküldeni sehova. Bármely hiba esetén el kell dobni az üzenetet.

5 A fájl szolgáltatás

A WildFly szerver **bin\temp** könyvtárban egy **NEPTUN_seats.txt** nevű fájlt kell feldolgozni:

```
/home/cloud/soi/wildfly-8.1.0.Final/bin/temp/NEPTUN_seats.txt
```

A fálj létezését 500 milliszekundumonként kell ellenőrizni, és amennyiben a fájl létezik, akkor a tartalmát fel kell dolgozni és törölni kell a fájlt.

A fájl soronként értelmezendő. A sorok tabokkal szeparált értékeket tartalmaznak. Az első oszlop a művelet típusa (**reserve** vagy **buy**), a második oszlop a moziterem sorának azonosítója, a harmadik oszlop a moziterem oszlopszáma, a negyedik oszlop pedig a székek darabszáma. A sor által reprezentált műveletet JSON formátumban tovább kell küldeni a **NEPTUN** nevű JMS üzenetsorba. Tehát ahány sor van a fáljban, annyi JMS üzenetet kell küldeni.

Például:

```
reserve    G    6    3
buy        J    4    5
```

Amennyiben ez a fájl tartalma, akkor a következő két JMS üzenetet kell küldeni a **NEPTUN** nevű JMS üzenetsorba:

```
{
    "type": "reserve",
    "row": "G",
    "column": "6",
    "count": 3
}
```

illetve:

```
{
    "type": "buy",
    "row": "J",
    "column": "4",
    "count": 5
}
```

6 Kimeneti fájl

A kimeneti fájlban mindig a moziterem aktuális állapotának kell látszania, függetlenül attól, hogy melyik szolgáltatáson keresztül történt módosítás. A fájl neve **NEPTUN_room.txt** legyen. A fájlt a WildFly szerver **bin\temp** könyvtárban kell elhelyezni:

```
/home/cloud/soi/wildfly-8.1.0.Final/bin/temp/NEPTUN_room.txt
```

A fájlban annyi sor szerepeljen, ahány sorból áll a terem, és minden sorban annyi karakter szerepeljen, ahány oszlopból áll a terem. Például, ha a terem 10 sorból és 20 oszlopból áll, és a B1-es széktől kezdve 2 szék zárolt, a G6-os széktől kezdve 3 szék foglalt, a J4-es széktől kezdve 5 szék el van adva, akkor a fájl tartalma a következő legyen:

```
FFFFFFFFFFFFFFFFFFF
LLFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFF
FFFFFRRRFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFFFFF
FFFSSSSSFFFFFFFFFFF
```

Az egyes betűk jelentése a következő:

- F: szabad (free)
- L: zárolt (locked)
- R: foglalt (reserved)
- S: eladva (sold)

7 Segítség a megoldáshoz

A feladat megoldása nem egyszerű, sok utánajárást igényelhet. A SwitchYard nagyon sok példát tartalmaz a WildFly szerver **quickstarts** könyvtárban. Érdemes ezeket megnézni.

A WSDL-ből a SwitchYard segítségével kell a Java interfészt generálni, mert a **wsimport** által generált interfészben a függvényeknek több paramétere is lehet, míg a SwitchYard függvényenként legfeljebb egy paramétert enged meg.

REST szolgáltatásként egy szokásos JAX-RS szolgáltatás készíthető, és azt lehet a SwitchYard segítségével publikálni.

A Java és XML közötti leképzést egy Java transzformer segítségével célszerű végezni. A JAXB transzformer a SOAP fault-okat nem jól kezeli.

Nem minden szolgáltatást célszerű Java Bean-ként implementálni. Néha célravezetőbb lehet egy Java Camel implementáció (például a kimeneti fájl készítésénél). A **camel-service** példaprogram arra mutat példát, hogy hogyan lehet Java Camel implementációt készíteni.

Egy egyszerű komponens csak egyetlen interfészt publikálhat service-ként, azonban a csomagoló kompozit komponens tetszőlegesen sok interfészt publikálhat service-ként.

A megoldás során mindent a SwitchYard binding-ok segítségével kell megoldani. A SwitchYard működésének más módon történő megkerülése tilos (pl. a JMS illetve a ki- és bementi fájlok kezelése során).

A belső állapotot egy valódi alkalmazás során célszerű adatbázisban tárolni. A házi feladatban elegendő statikus változóban tárolni az adatokat.

8 Beadandók

Beadandó egyetlen ZIP fájl. Más tömörítési eljárás használata tilos!

A ZIP fájl gyökerében egyetlen könyvtárnak kell lennie, amely a SwitchYard alkalmazás:

- **Sca_NEPTUN:** a SwitchYard alkalmazás teljes egészében

A lefordított class fájlokat nem kötelező beadni.

Fontos: a projektnek hiba nélkül fordulnia kell, ha az **Sca_NEPTUN** könyvtárban a következő parancsot kiadjuk:

```
mvn clean package
```

A szolgáltatásnak a fenti példákban szereplő XML és JSON formátumokat kell támogatnia. Más XML vagy JSON formátum/elnevezés használata tilos!

A megoldásnak a telepítési leírásban meghatározott környezetben kell fordulnia és futnia, más JAR illetve 3rd party könyvtár nem használható! Csak az alapértelmezetten generált SwitchYard POM-ban szereplő függőségek használhatók, más függőségek felvétele a Maven POM fájlba tilos!

Fontos: a dokumentumban szereplő elnevezéseket és kikötéseket pontosan be kell tartani!

Még egyszer kiemelve: a NEPTUN szó helyére mindig a saját Neptun-kódot kell behelyettesíteni, csupa nagybetűkkel!