5. BPEL 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 olyan BPEL folyamat elkészítése, amely két mozi jegyfoglalási ajánlatai közül a jobbat (olcsóbbat) választja ki.

2 A mozi webszolgáltatások

A két mozi ajánlatait egy-egy webszolgáltatáson keresztül lehet lekérdezni. Mindkét mozi ugyanazt az interfészt valósítja meg (**CinemaReservation.wsdl**).

Az interfészt az alábbi pszeudo-kód írja le:

```
[Uri("http://www.iit.bme.hu/soi/hw/CinemaReservation")]
namespace CinemaReservation
{
    struct Offer
    {
        string OfferId;
        int Price;
    }

    struct Seat
    {
        string Row;
        string Column;
    }

    struct Reservation
    {
        string ReservationId;
        Date Date;
        string Room;
        Seat[] Seats;
    }

    interface ICinemaReservation
    {
        Offer GetOffer(int seatCount);
        Reservation Reserve(string offerId);
    }
}
```

A **GetOffer** függvénynek meg kell adni, hogy hány darab széket szeretnénk lefoglalni (**seatCount**). Erre a szolgáltatás visszaad egy **Offer** típusú ajánlatot, amelyben szerepel egy azonosító (**OfferId**) és az ajánlott ár (**Price**).

Amennyiben az ajánlatot elfogadjuk, akkor a **Reserve** függvényen keresztül be kell adnunk paraméterként a korábban megkapott **OfferId**-t. Ennek eredményeként visszakapjuk a foglalás adatait (a foglalás azonosítóját, az előadás dátumát, a terem nevét és a lefoglalt székek pozícióit).

Az két mozi szolgáltatásai a következő URL-eken érhetők el:

`http://localhost:5001/CinemaReservation`

`http://localhost:5002/CinemaReservation`

3 A BPEL folyamat

A BPEL folyamat egy jegyfoglalási ügynökként viselkedik, vagyis mindkét mozitól kér be ajánlatot, és a jobb ajánlatot foglalja le.

A folyamat a **CinemaAgent.wsdl** interfészt implementálja, amelyet az alábbi pszeudo-kód ír le:

```
[Uri("http://www.iit.bme.hu/soi/hw/CinemaAgent")]
namespace CinemaAgent
{
    interface ICinemaAgent
    {
        CinemaReservation.Reservation Reserve(int seatCount);
    }
}
```

A **Reserve** függvénynek meg kell adnunk, hogy hány darab széket szeretnénk lefoglalni, és eredményként visszakapjuk a két mozi ajánlatai közül a jobbat (olcsóbbat).

A BPEL folyamatot a következő URL-en kell publikálni:

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

ahol a *port* helyén a szerver megfelelő portszáma áll. A **NEPTUN** rész context-path-ként adható meg az SCA SOAP binding tulajdonságai között.

4 Segítség a megoldáshoz

Célszerű megvizsgálni a SwitchYard BPEL példákat **quickstarts** könyvtárban.

A megoldás során sok kifejezést XPATH-ban lehet csak leírni, így az XPATH ismereteknek nézzünk utána, ha szükséges.

A SwitchYard az Apache ODE implementációt használja a BPEL folyamat futtatására, így szükség esetén az Apache ODE példáit is megvizsgálhatjuk.

Ebben a feladatban nagyon fontos az XML névterek pontos használata! Nemcsak az XPATH kifejezésekben, a WSDL-ben és a BPEL-ben, hanem a konfigurációs fájlokban (pl. deploy.xml) is.

A két mozit reprezentáló szolgáltatást célszerű JAX-WS szolgáltatásként megvalósítani. **Ezeket azonban nem kell beadni!** Egy egyszerű konzol alkalmazásból az Endpoint.publish() hívással lehet webszolgáltatást publikálni. Ez azonban csak tesztelési célból ajánlott, éles környezetben mindenképpen egy alkalmazásszerver segítségével publikáljuk a szolgáltatásainkat!

5 Beadandók

Beadandó egyetlen ZIP fájl. Más tömörítési eljárás használata tilos!

Beadni csak a BPEL folyamatot implementáló SwitchYard projektet kell. A projekt csak hivatkozzon a két mozis szolgáltatásra az URL-jükön keresztül. A mozis szolgáltatások implementációja nem lehet benne a SwitchYard projektben. A projekt csak a BPEL folyamatot tartalmazhatja.

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

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

A lefordított class fájlokat nem kötelező beadni.

A BPEL folyamatnak a megadott WSDL és XSD fájlokat kell implementálnia és használnia! Ezen fájlok módosítása 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!