Java programozási nyelv 2012-2013/ősz
9. óra

Szervlet-JSP együttműködés

Kérés továbbítás technikái

legradi.gabor@nik.uni-obuda.hu szenasi.sandor@nik.uni-obuda.hu

Szervlet-JSP együttműködés

Témakörök

Osztálykönyvtár elérése

Kérések, objektumok továbbítása

Beanek használata

Osztályok használata JSP oldalakban

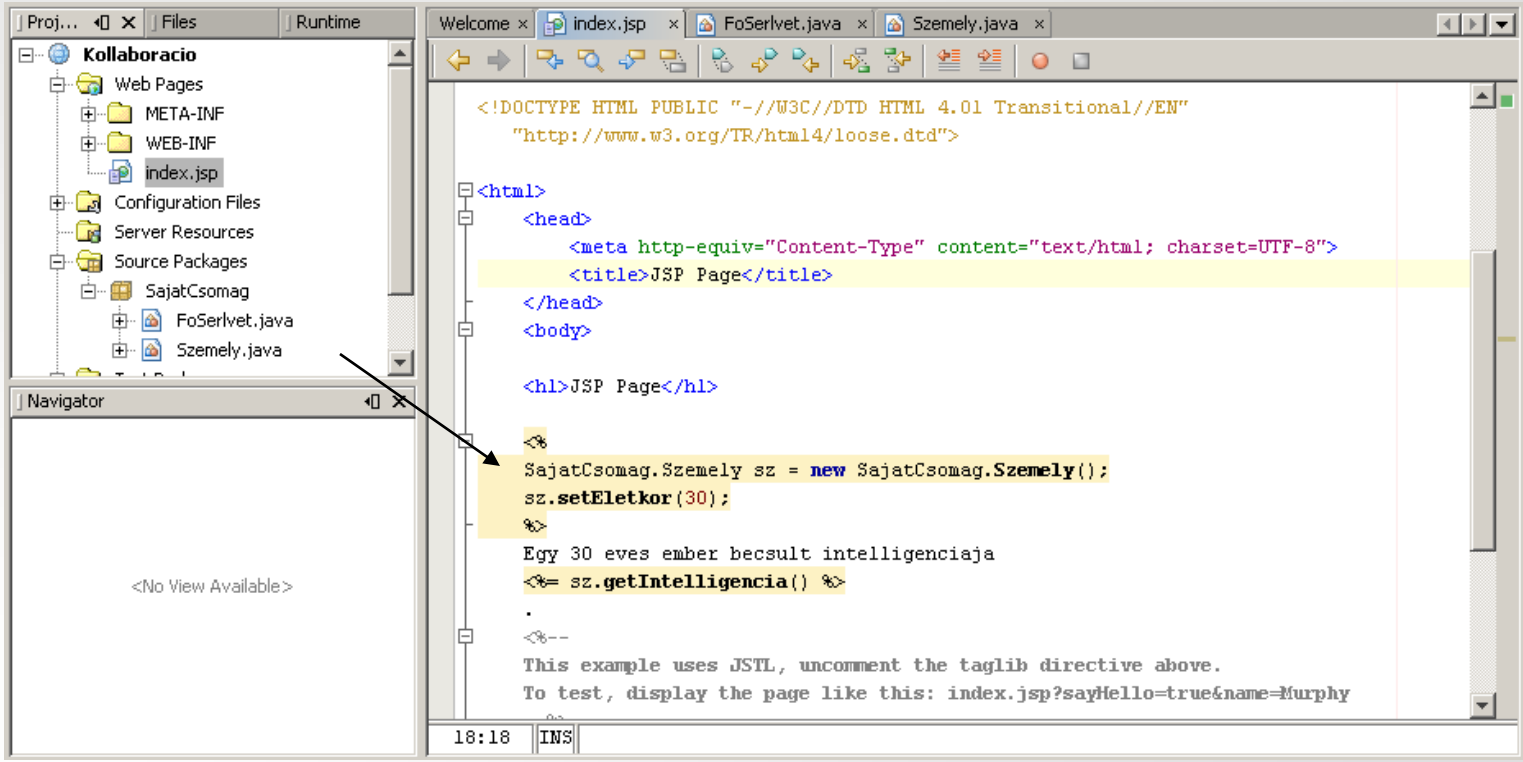
- Java beépített osztálykönyvtára
  - Az osztálykönyvtár bármelyik eleme elérhető, érdemes használni az import direktívát
  - Fontos figyelembe venni, hogy az adott web konténer milyen Java verziót támogat, ez gyakran alacsonyabb, mint a fejlesztőeszköz által használt változat
- A project más osztályai
  - A fejlesztés során az üzleti logikát képviselő osztályokat célszerű a megjelenítési szinttől függetlenül megvalósítani
  - A szervletek fejlesztése során erre eddig is ügyeltünk, célszerű az így elkészített külső osztályokat használni a JSP oldalakban is (fizikailag ugyanazokat az osztályokat)
- Egyéb külső osztályok
  - Gyakran szükség lehet külső osztálykönyvtárak használatára is (pl. adatbázis-kezelő rendszer irányába kapcsolatot nyújtó Java nyelven megírt meghajtó)
  - Ezeknél ügyelni kell, hogy a fordítást követően a JSP futtatásakor is elérhetőek legyenek a konténer számára

Osztályok elhelyezése

- JSP oldalak esetén lehetőség van
  - A konténer számára elérhető osztályokból új példányok létrehozására
  - Ezekkel a példányokkal a már megszokott módon műveletek végzésére, adattagokhoz való hozzáférésre
  - A konténer számára elérhető osztályok statikus mezőinek elérésére, statikus metódusok meghívására
- Osztályok elhelyezésének szabályai
  - A JSP oldalak feldolgozása során a web konténer hasonló módon keresi a szükséges külső osztályokat, mint ahogy a szervleteknél láttuk (hiszen a JSP oldal is egy szervlet)
  - Ennek megfelelően a kontextus hasonló könyvtáraiba érdemes elhelyezni a szükséges osztályokat (`WEB-INF/classes`)
  - Több osztály esetén már célszerű csomagokat használni, ez esetben a megszokott szabályokat itt is be kell tartani, minden osztály a megfelelő alkönyvtárba kerüljön
  - WAR állomány esetén hasonló az eljárás

NetBeans támogatás

- NetBeans esetén az elhelyezés nem jelent különösebb problémát
  - A szükséges osztályok a már megismert módon létrehozhatók
  - Fordításkor a szükséges osztályokat elhelyezi a kontextus megfelelő helyén, hogy a JSP oldalakból is elérhetőek legyenek



Szervlet-JSP együttműködés

Témakörök

Osztálykönyvtár elérése

Kérések, objektumok továbbítása

Beanek használata

Szervletek együttműködése

- Nagyobb fejlesztések során gyakori, hogy egyetlen hívás feldolgozását is több komponenssel szeretnénk megoldani. Ennek előnyei:
  - Jobban áttekinthető kód, az egymástól független komponensek szétválasztásával
  - Az egyes részfeladatokat meg lehet oldani az ahhoz legjobban illeszkedő technológiával
  - Erőforrások elosztása érdekében, a feladat egyes részeit fizikailag más szervereken található szervletek végezhetik el
- Néhány tipikus példa az együttműködésre
  - A beérkező kérést egy szervlet fogadja, megkezdi annak feldolgozását, eközben más szervleteket is felhasználva a kimenet generálásához
  - A kérést fogadó szervlet a kérést egészében átadja egy másik szervletnek, így a kimenet generálásában nem vesz részt
  - A beérkező kérést egy szervlet fogadja, elvégzi az üzleti oldalon szükséges feldolgozást, majd a kimenet generálása érdekében továbbítja a kérést egy JSP oldal irányába

Együttműködés technikai lehetőségei

- Szervletek láncolása (servlet chaining)
  - Még a J2EE előtt megjelent, nem terjedt el
  - Alapja, hogy a web konténeren egy alias alatt szervleteknek egy listáját regisztráljuk, és a beérkező kérés során ezeket a webszerver sorban lefuttatja, majd közös kimenetüket küldi vissza válaszként
  - Nem tárgyaljuk
- Kérés elküldés (request dispatching)
  - Ez a technika lehetővé teszi, hogy egy szervlet a feldolgozása közben kéréseket küldjön más erőforrásokhoz (szervlet, JSP, XHTML oldal stb.)
  - A web konténer a kérés beérkezésekor a szokásos módon jár el, létrehozza, és az aliashoz rendelt szervletnek átadja a kérés és válasz objektumokat
  - Ezt a hívást a *RequestDispatcher* interfészt megvalósító erőforrások támogatják. A szervlet ezeknek a megfelelő metódusain keresztül tudja befolyásolni a kérés feldolgozását

RequestDispatcher interfész

- public void forward(ServletRequest request, ServletResponse response) throws ServletException, IOException
  - Ez a metódus teszi lehetővé a szervlet vagy JSP oldal számára, hogy továbbítsa a kérést egy másik erőforrás irányába
  - Ebben az esetben a kérés teljes feldolgozása (és az ezzel járó felelősség) a hívott erőforrásra hárul
- public void include(ServletRequest request, ServletResponse response) throws ServletException, IOException
  - Ez a metódus lehetővé teszi, hogy a szervlet vagy JSP oldal meghívjon egy másik erőforrást, majd annak kimenetét beillessze az általa generált kimenetbe
  - Ebben az esetben a kérés feldolgozásának csak egy részét végzi a hívott fél, az előtte és utána a kimenetre írt adatok megmaradnak
  - Emiatt a kérés feldolgozásának a felelőssége is az elsőként hívott szervletnél marad, az így meghívottaknak lehetőségük sincs bizonyos paraméterekhez hozzáférni (ContentType stb.)

Erőforrásokhoz való hozzáférés

- A kérés továbbításához az előző két metódust kell meghívni, azonban ehhez előbb hozzá kell férni a szükséges erőforráshoz
- Ehhez a *ServletContext* osztályon kereszül van lehetőségünk (erről már volt szó, egy ilyen típusú referenciával hivatkozunk a szervletet tartalmazó kontextusra)
  - public RequestDispatcher getRequestDispatcher(String path)
  - public RequestDispatcher getNamedDispatcher(String name)
- Lehetőség van a *ServletRequest* osztályon keresztül is hozzáférni ugyanehhez:
  - public getRequestDispatcher(String path)
- Az útvonalat tartalmazó metódusok paraméterként a kívánt erőforrás abszolút (a *ServletRequest* esetén esetleg relatív) útvonalát várják
- A nevet váró metódus pedig paraméterként az erőforráshoz bejegyzett alias-t várja (lásd. `web.xml`)

Példa kérés küldésére

Felhasználó azonosítása, majd ettől függően a kérés különböző irányokba való továbbítása

```
...
Szemely aktfelh = (Szemely)request.getSession().getAttribute("aktfelh");
if (aktfelh == null) {
   RequestDispatcher rd = request.getRequestDispatcher("/login.html");
   rd.forward(request, response);
} else
   if (aktfelh.Belepett()) {
      resp.setContentType("text/html");
      RequestDispatcher rd = request.getRequestDispatcher("/servlet/Fejlec");
      rd.include(request, response);
      rd = request.getRequestDispatcher("/servlet/Feldolgozas");
      rd.include(request, response);
   } else {
      RequestDispatcher rd = getServletContext().getNamedDispatcher("loginsrv");
      rd.forward(request, response);
   }
...
```

Objektumok átadása

- A kérések továbbítása önmagában nem lenne elégséges eszköz egy feladat részfeladatokra bontásához és részenkénti végrehajtásához, további feltétel, hogy az egyes feladatokat végrehajtók számára adatokat tudjunk küldeni, és tőlük adatokat tudjunk fogadni
- Ennek módja függ az adat szükséges „élettartamától”
  - Csak egy kérés feldolgozása során szükséges
  - Egy teljes folyamaton belül életben kell hagyni
  - Kérésektől függetlenül a szervletek működése során szükséges
  - Szerver újraindítása után is megtartsa az értékeket
- A már eddig is megismert módszerek közül számos itt is jól alkalmazható:
  - ServletRequest. setAttribute/getAttribute
  - HttpSession. setAttribute/getAttribute
  - ServletContext .setAttribute/getAttribute
  - osztályok statikus mezői
  - külső erőforrások (adatbázisok stb.)

Szervlet-JSP együttműködés

Témakörök

Osztálykönyvtár elérése

Kérések, objektumok továbbítása

Beanek használata

Beanek használata

- Szervletek és JSP oldalak közös használata során nagymértékben elválasztható egymástól az üzleti logika és a megjelenítéshez tartozó XHTML+scriptlet
- A JSP oldalak szerkesztéséhez azonban összetettebb esetben továbbra is szükség van rövidebb-hosszabb Java kódreszletekre, amik ismét előhoznak néhány problémát:
  - Designerek ehhez nem értenek (nem is akarnak)
  - Szabványos szerkesztőprogramok nem kezelik
- A cél tehát az, hogy lehetőség szerint minimálisra csökkenjen az JSP oldalakban a beágyazott scriptletek száma. Erre a Servlet API egy jó alternatívát kínál, a megjelenítéshez használatos Java kódokat csomagoljuk be beanekbe, és a JSP oldalon már csak ezeket kelljen használni (ez tulajdonságok írását/olvasását jelenti)
- Ezzel elérhető a végső cél: dinamikus oldalfelépítés mellett 0 sor programkód a megjelenítési rétegben

Beanek elérése

- A scriptletekben tetszőleges (akár saját készítésű) osztályokat lehet használni, amennyiben az oldal fordítása során ezek elérhetőek a konténer számára
- A beanek használatával azonban néhány új lehetőség is felmerül, például a kérés kiszolgálása során a beanek tulajdonságait automatikusan fel lehet tölteni a kérésben érkező azonos nevű paraméterek értékeivel
- Lehetőség van a hatáskör magas szintű beállítására is. Egy bean hozzárendelhető:
  - oldalhoz – lap generálása után megszűnik
  - kéréshez – szervletek közt továbbított kérés esetén továbbítódik
  - ügyfélmenethez – session részeként, tehát egymást követő kérések között is megmarad
  - alkalmazáshoz – a teljes webalkalmazás számára hozzáférhető különböző szervletek számára

Bean beágyazása JSP oldalba

- Beágyazás általános alakja:
  <jsp:useBean id="név" scope="hatókör" class="osztálynév" type="típus">
  </jsp:useBean>
- id attribútum
  A bean nevét adja meg, ezen keresztül lehet rá hivatkozni
- scope attribútum
  A már említett hatókör meghatározása, értékei ennek megfelelően: page, request, session, application
- class, type attribútum
  A class a létrehozandó bean osztályának a nevét adja meg, a type pedig azt, hogy milyen típusúként legyen nyilvántartva (a polimorfizmus miatt ez különbözhet)
- beanName attribútum
  A class helyett ez is megadható, ilyenkor a Beans.instantiate() metódussal jön létre a példány

Bean tulajdonságok beállítása

- A legalapvetőbb értékadáson túl a JSP lehetőséget ad a beérkező paraméterek alapján történő értékadásra is
- Értékadás megadott értékkel
  <jsp:setProperty name="id" property="tulajdonság" value="érték" />
- Értékadás egy paraméter alapján (a kérés megadott nevű paraméterének értékét bemásolja a bean tulajdonságába)
  <jsp:setProperty name="id" property="tulajdonság" parameter="paramnév" />
- Értékadás azonos nevű paraméter alapján
  <jsp:setProperty name="id" property="tulajdonság" />
- Értékadás minden paraméternévvel egyező tulajdonságnak
  <jsp:setProperty name="id" property="*" />

Bean tulajdonságok lekérdezése

- A bean egy tulajdonságának a lekérdezése
  <jsp:getProperty name="id" property="tulajdonság" />
  Ennek hatására az *id* azonosítóval létrejött bean *tulajdonság* nevű attribútumának az értéke illesztődik be az JSP oldalba
- Az attribútumhoz tartozó getXXX() metódus természetesen összetett kódot is tartalmazhat (adatbázishozzáférés, számítások stb.)
- Ideális esetben tehát a felhasználónak csak egy ilyen (az XML szabványnak teljesen megfelelő) elemet kell felvennie a tervezett JSP oldalba, így az oldal szabványos XHTML szerkesztőkkel módosítható (és nincs benne Java kód)
- Beanek segítségével egészen magasszintű feladatokat tudunk megoldani meglehetősen egyszerűen. Például:
  - Személy adatait tartalmazó bean. Azonosító és jelszó megadása után az adatbázisból betöltött adatok egyszerűen lekérdezhetők a tulajdonságokon keresztül
  - Adót kiszámító bean. Tulajdonságkon át megadott személyes adatok alapján másik tulajdonságokból kiolvasható a fizetendő adó

TagLib

- Ezen az úton a következő lépés a saját saját tag függvénytár használata. Ezekben saját tag-eket készíthetünk, amiket az XHTML elemekhez hasonlóan használhat a felhasználó
- Ezek lehetnek tartalommal rendelkezők, illetve tartalom nélküliek (mint XHTMLben a ... vagy a
)
- Az XHTML elemeknek megfelelően lehetnek saját attribútumaik is (mint XHTMLben az <input type="xxx">)
- Ezek egymásba is ágyazhatók, így a saját tagfüggvények hasznátával így is kinézhet egy weboldal:

```
<próbálkozás darab="5">
        Próba: <időbélyeg /> <adatkapcsolat szerver="oracle.com" />
</próbálkozás>
```

- Ezzel véglegesen eltűntek a programozási elemek

Gyakorló feladat

Készítsen el egy egyszerű bevásárlólista JSP oldalt:

- A listát egy (sessionbe ágyazott) *BevasarloLista* objektum tárolja, ami az alábbi funkciókkal rendelkezik:
  - Új elem felvétele – a paraméterként megadott szöveget felveszi a lista legutolsó helyére
  - Aktuális lista – visszatérési értéke egy szöveges tömb, ami tartalmazza az eddig felvett elemeket
- Az oldal tetején jelenjen meg a logó, mellette a pontos idő
- Ez alatt egy keretben jelenjen meg egy reklámszöveg, ami 5 előre megadott mondat közül legyen az egyik
- Alatta egy táblázatba kerüljön a bevásárlólista (első oszlop a sorszám, második oszlop a cikk neve)
- A lista alatt egy mezőben legyen lehetőség új tétel megadására, ezt a „Felvesz” gomb lenyomásával lehessen hozzáadni a listához

Gyakorló feladat

Valósítsa meg az alábbi fórum alkalmazást!

- Készítse el az alábbi segédosztályokat:
  - *Felhasználó* – név, emailcím
  - *Hozzászólás* – felhasználó, üzenet szövege
  - *Fórum* – fórum neve, hozzászólások listája
  - *Fórumtároló* – tárolja az összes fórumot
- Készítsen egy web alkalmazást (tetszőleges számú szervlet és JSP oldal felhasználásával), ami elvégzi az alábbi feladatokat
  - induláskor létrehoz egy *Fórumtároló* objektumot néhány fórummal
  - adjon lehetőséget tetszőleges számú felhasználó párhuzamos belépésére
  - belépést követően a képernyőn egy táblázatban jelenjen meg minden eltárolt fórum utolsó 5 bejegyzése
  - minden fórum táblázat alatt egy mezőben legyen lehetőség egy új hozzászólás felvételére
  - legyen lehetőség új fórum létrehozására is

Házi feladat

A, JSP oldal segítségével készítsen egy háttérképletöltő alkalmazást

- Ehhez készítsen egy *HatterKep* osztályt az alábbi mezőkkel, majd néhány ilyen elemmel töltsön fel egy tömböt a szervlet init metódusában (a kis és nagy képeket töltse fel a szerverre kívülröl elérhető állományokként):
  - megnevezés, szélesség, magasság
  - témák (szöveges tömb, pl „kutya”, „ari” stb.)
  - URL a kis és nagy képhez
- A felhasználó a kezdőoldalon tudjon megadni szűrési feltételeket a fenti tulajdonságokra
- Az „Elküldés” gombra kattintva jelenjen meg egy táblázat, benne az összes, a feltételnek megfelelő képpel. Azonosan formázva jelenjen meg a kép minden adata és a kis kép
- A kis képre kattintva töltődjön le a nagy kép (a kis kép legyen egy hivatkozás a nagyra)

B, Készítsen egy felhasználói adatokat kezelő beant

- Ehhez először hozzon létre egy *Felhasznalo* osztályt az alábbi mezőkkel:
  - nev, jelszo, teljesnev, eletkor
- Majd hozzon létre egy *FelhasznaloManager* beant (ami tárol néhány felhasználót, ezek a konstruktorban jönnek létre) az alábbi, kívülről elérhető tulajdonságokkal:
  - nev, jelszo – olvasható, érték adható
  - bejelentkezett, teljesnev, eletkor – olvasható, a név/jelszó pároshoz tartozó felhasználó adatai
- A kezdőoldalon megadott név/jelszó párost a következő JSP oldalon adjuk át egy *FelhasznaloManager* beannek, majd írjuk ki a képernyőre a felhasználó adatait
- Egy „Következő” hivatkozásra kattintva a következő oldalon jelenjen meg a „porno” vagy a „semmi” szó attól függően, hogy a felhasználó elmúlt-e 18 éves (tehát a bean a menethez kapcsolódjon)

Ajánlott irodalom

Az óra anyagához kapcsolódó irodalom

- Nyékyné Gaizler Judit: J2EE útikalauz Java programozóknak; ELTE TTK Hallgatói alapítvány, Budapest
- Jason Hunter: Java szervletek programozása; O'Really-Kossuth, Budapest, 2002
- The J2EE 1.4 Tutorial – Chapter 12: JavaServer Pages Technology

  http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

  http://java.sun.com/j2ee/1.4/docs/tutorial/doc/J2EETutorial.pdf