Java programozási nyelv 2012-2013/ősz

6. óra

Interaktív weboldalak készítése

XHTML form

Adatok feldolgozása szervletekkel

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

Témakörök

XHTML Form elemek

Szervlet kérés paraméterei

Form adatok feldolgozása

NetBeans támogatás

Űrlapok kezelése

- Az XHTML oldalakban van lehetőség a felhasználó által kitöltendő űrlapok megjelenítésére. Ezekben a hagyományos programozási nyelvekben megismertekhez hasonló elemek jelenhetnek meg (szövegmező, lenyíló stb.), feldolgozásuk azonban nem a már megismert eseményorientált módon történik
- Az űrlapok adatai elérhetők kliensoldali technikákkal (JavaScript, Appletek), vagy visszaküldhetők a webszerverhez feldolgozás céljából
- A Java megoldása erre a szervleteken alapszik. Az űrlap adatainak elküldésekor a böngésző a HTTP prokoll segítségével az eddig megismert módon kapcsolódik a szerverhez, az elérni kívánt erőforrás azonosítóján túl azonban elküldi az űrlap tartalmát is
- A szervlet az így kapott adatok feldolgozása után az eddig megismert módon küldhet vissza valamilyen (általában a küldött adatoktól függő) választ

XHTML form elem

- Egy űrlapot az alábbi módon definiálhatunk:

```
<form method="post" action="http://szerver/wapp/szervlet" >
</form>
```

- Action tulajdonság

  Az erőforrás címe, ahová a böngésző az űrlap adatait elküldi majd (esetünkben egy szervlet). Ez nem feltétlenül azonos azzal a kiszolgálóval, ahonnan ez az oldal letöltődött
- Method tulajdonság lehetséges értékei:
  - GET – a böngésző a HTTP GET parancsot fogja elküldeni a megadott címre. A form adatai ilyenkor az URL részeként adódnak át, emiatt ez csak kevés adat esetén ajánlatos
  - POST – a böngésző a HTTP POST parancsot fogja elküldeni a megadott címre. A form adatai ilyenkor a kérés fejlécében adódnak át, elvileg korlátlan mennyiségű adat elküldhető

Form elemek - szövegbeviteli mező

- Használatára példa

```
<input type= "text" name= "mezonev" />
```

- Elérhető attribútumok
  - name – a vezérlő neve
  - size – mező szélessége
  - maxlength – beírható szöveg maximális hossza
  - value – kezdeti érték
  - readonly – írásvédett mező
  - disabled – nem engedélyezett mező
- type értéke meghatározhat más típusokat
  - text – egyszerű szövegbeviteli mező
  - password – jelszó megadás (beírt szöveg nem látszik)
  - file – megjelenik mellette egy böngészés gomb
  - hidden – rejtett mező (az egész mező láthatatlan)
- Továbbított adat: *mezőnév=tartalom*

Form elemek – többsoros szövegmező

- Használatára példa

```
<textarea name="soksor" cols="30" rows="10" wrap="virtual">
        Alapértelmezett szöveg
</textarea>
```

- Elérhető attribútumok
  - name – a vezérlő neve
  - cols – mező szélessége
  - rows – sorok száma
  - wrap – tördelés módja
- Továbbított adat: *mezőnév=tartalom*

  A továbbított tartalom pontosan megfelel annak, amit a felhasználó beírt, tehát az összes szóköz, soremelés elküldésre kerül

Form elemek - checkbox

- Használatára példa

```
<input type="checkbox" name="mezonev"/>
```

- Elérhető attribútumok
  - name – a vezérlő neve
  - value – értéke (ha nincs megadva, akkor ”on”)
  - checked – alapértelmezett érték
- A más programozási nyelvekben megszokott módon az egymás mellé helyezett checboxok értékei nincsenek egymásra hatással
- Továbbított adat:
  Ha be van jelölve: *mezőnév=érték*
  Ha nincs bejelölve: *nincs*

Form elemek - radiobutton

- Használatára példa

```
<input type="radio" name="kedvenc" value="kuty" />Kutya
<input type="radio" name="kedvenc" value="macs" />Macska
```

- Elérhető attribútumok
  - name – a vezérlő neve
  - value – értéke (ha nincs megadva, akkor ”on”)
  - checked – alapértelmezett érték
- A komponens csak a „gombot” rajzolja ki, a szöveget mellé már egyszerű szövegként kell kiírni
- A form tartalmazhat több azonos nevű radiobutton komponenst, ilyenkor ezek közül egyszerre csak egy lehet kiválasztott
- Továbbított adat:
  Ha valamelyik be van jelölve: *mezőnév=érték*
  Ha egyik sincs bejelölve: *nincs*

Form elemek - lista

- Használatára példa

```
<select name="allatok" multiple="multiple">
        <option value="kuty">Kutya</option>
        <option value="macs">Macska</option>
</select>
```

- Elérhető attribútumok
  - name – a vezérlő neve
  - multiple – több elem is kiválasztható
  - rows – megjelenő sorok száma
- Az értékeket <option> elemek között kell megadni:
  - value – az elküldendő érték
  - selected – alapértelmezetten kiválasztott legyen-e
- Továbbított adat:
  Ha egy elem be lett jelölve: *mezőnév=érték*
  Több elem esetén: *mezőnév=érték,mezőnév=érték*
  Ha egyik sincs bejelölve: *nincs*

Form elemek - gombok

Adatok elküldése a szervernek

```
<input type="submit" value="Adatok elküldése" />
```

- Elérhető attribútumok
  - name – a vezérlő neve
  - value – a gombon megjelenő felirat

Mezők törlése (alapértelmezett érték betöltése)

```
<input type="reset" value="Adatok törlése" />
```

- Elérhető attribútumok
  - name – a vezérlő neve
  - value – a gombon megjelenő felirat

Témakörök

XHTML Form elemek

Szervlet kérés paraméterei

Form adatok feldolgozása

NetBeans támogatás

HTTPServlethez kapcsolódó osztályok

- A *HTTPServlet* kérést kiszolgáló metódusai minden egyes kérés esetén az alábbi típusú objektumokat kapják paraméterül:
  - *HTTPServletRequest*
    A bejövő kérés adatait tartalmazza, a szervlet ezen keresztül kap információkat
  - *HTTPServletResponse*
    A kimenő válasz adatait tartalmazza, a szervlet ezen az objektumon keresztül tudja visszaküldeni a választ és annak paramétereit a kliens számára
- A szervlet működése közben folyamatosan elérhető
  - *ServletContext*
    A szervletet tartalmazó kontextus adatai. Mivel ez menet közben nem változhat, ezért nem a kérés paramétere, hanem az inicializáláskor állítódik be.
    A szervlet működése során azonban bármikor lekérdezhető a getServletContext() metódus segítségével

HTTPServletRequest metódusai

- Kérés módjának lekérdezése
  - public String getProtocoll()
    Visszatérési értéke a kérés által használt protokoll neve és verziószáma (egy szervlet ugyanis nem feltétetlenül csak a HTTP protokollon keresztül érhető el)
  - public String getMethod()
    Visszatérési értéke a kérésben adott parancs (GET, POST stb.)
- Kapcsolat adatainak lekérdezése
  - public String getRemoteHost()
    A kérést indító gép IP címe
  - public int getRemotePort()
    A kéréshez tartozó kapcsolat távoli portszáma
  - public String getLocalName()
    A kérést kiszolgáló szerver IP címe. Lényeges lehet, ha ugyanaz a szerver több különböző néven is elérhető a kliensek számára
  - public int getLocalPort()
    A kéréshez tartozó kapcsolat helyi portszáma

HTTPServletRequest metódusai

- Form által küldött paraméterek lekérdezése
  - public String getQueryString()
    Az URL részeként küldött paraméterstring (tulajdonképpen a teljes URL ? utáni részét adja vissza)
- Hozzáférés egyetlen paraméterhez
  - public String getParameter(String name)
    Visszaadja a megadott nevű paraméter értékét. Csak akkor használható, ha biztosak vagyunk abban, hogy a beérkező érték egyedi lesz (pl. egy szövegmező tartalma)
  - public String[] getParameterValues(String name)
    Visszaadja a megadott nevű paraméterhez tartozó értékeket (pl. listána esetén ha több elem is kiválasztható, akkor ugyanazon a néven megkapjuk az összes kiválasztott elem értékét, ez a metódus ezeket egy tömbben adja vissza)
  - public Enumeration getParameterNames()

    A felsorolás tartalmazza a böngésző által elküldött összes paraméter nevét

HTTPServletRequest metódusai

- Küldött adatokhoz való hozzáférés
  - public String getCharacterEncoding()
    A kliens által a törzsadatok küldésekor használt karakter kódolási mód
  - public int getContentLength()
    Küldött törzsadat hossza
  - public String getContentType()
    A kliens által küldött adat MIME típusa
  - public ServletInputStream getInputStream()
    Csatorna a küldött adatok kiolvasásához. A csatornáknál megismert módon ezen a streamen keresztül közvetlenül, vagy szűrők segítségével tudunk hozzáférni a küldött adatokhoz
  - public BufferedReader getReader() throws java.io.IOException
    Visszatérési értéke egy *BufferedReader* objektum, ami hozzá lett kapcsolva az előző adatfolyamhoz. Az előző metódus által visszaadott objektummal együtt nem használható

HTTPServletRequest metódusai

- HTTP fejlécmező kiolvasása
  - public String getHeader(String name)
    Visszaadja a megadott nevű fejlécmező értékét (vagy ha nincs ilyen a kérésben, akkor null-t)
  - public long getIntHeader(String name)
    Egész szám típusú fejlécmező értékének kiolvasása
  - public long getDateHeader(String name)
    Dátum típusú fejlécmező értékének kiolvasása
  - public Enumeration getHeaders(String name)
    Több értéket is tartalmazó fejlécmezőknél az értékeket tartalmazó tömb lekérdezése (pl. „Accept-Language”)
  - public Enumeration getHeaderNames()
    Visszaadja a kliens által a kérésben küldött összes fejlécmező nevét
- Kérésben szereplő URL adataihoz való hozzáférés
  - public StringBuffer getRequestURL()
  - public String getRequestURI()
  - public String getServletPath()

HTTPServletResponse metódusai

- Válasz típusának beállítása
  - public void setContentType(String type)
    Paramétere egy MIME típus (pl. ”text/html”)
  - public void setContentLength(int len)
    Válasz hosszának meghatározása. Nem kötelező, de nagyobb mennyiségű adat esetén érdemes beállítani, hogy a felhasználó folyamatosan lássa, hogy a letöltés hol tart
  - public void setCharacterEncoding(String charset)
    A válasz küldése során használt karakter kódolási mód meghatározása
- Válasz elküldési lehetőségei
  - public ServletOutputStream getOutputStream() throws IOException

    Byte alapú adatok visszaküldésére szolgáló csatorna (pl. ha a válasz nem egy HTML oldal, hanem egy kép)
  - public PrintWriter getWriter() throws IOException
    Szöveges adat visszaküldésére szolgáló csatorna (a metódus visszatérési értéke tulajdonképpen egy szűrő a fenti streamen)

HTTPServletResponse metódusai

- Válasz státuszkód beállítása
  - public void setStatus(int sc)

    Segítségként számos konstans érhető el, pl:
    - public static final int SC_OK
    - public static final int SC_NO_CONTENT
    - public static final int SC_NOT_FOUND
    - public static final int SC_INTERNAL_SERVER_ERROR
    - public static final int SC_NOT_IMPLEMENTED
    - public static final int SC_SERVICE_UNAVAILABLE
    - public static final int SC_UNAUTHORIZED
    - public static final int SC_MOVED_TEMPORARILY
    - public static final int SC_MOVED_PERMANENTLY
- Átirányítási kérelem küldése
  - public void sendRedirect(String location) throws IOException

    A megadott címre irányítja át a böngészőt (valójában elküld egy „átirányítás” státuszkódot, és egy „Location” fejlécmezőt, ahol megadja az új URL-t

HTTPServletResponse metódusai

- Fejlécmezők beállítása
  - public void addHeader(String name, String value)
    Új szöveges típusú válasz fejlécmező felvétele
  - public void addIntHeader(String name, int value)
    Új egész szám típusú fejlécmező felvétele
  - public void addDateHeader(String name, long value)
    Új dátum típusú fejlécmező felvétele

Témakörök

XHTML Form elemek

Szervlet kérés paraméterei

Form adatok feldolgozása

NetBeans támogatás

Példa alkalmazás

Készítsünk webalapú horoszkópkészítő alkalmazást, amelyik a kezdőoldalon bekéri a felhasználótól az alábbi adatokat:

- Név
- Születési dátum
- Mire vágyik
- Néhány ismerősének a neve
- Optimista/pesszimista horoszkópot kér

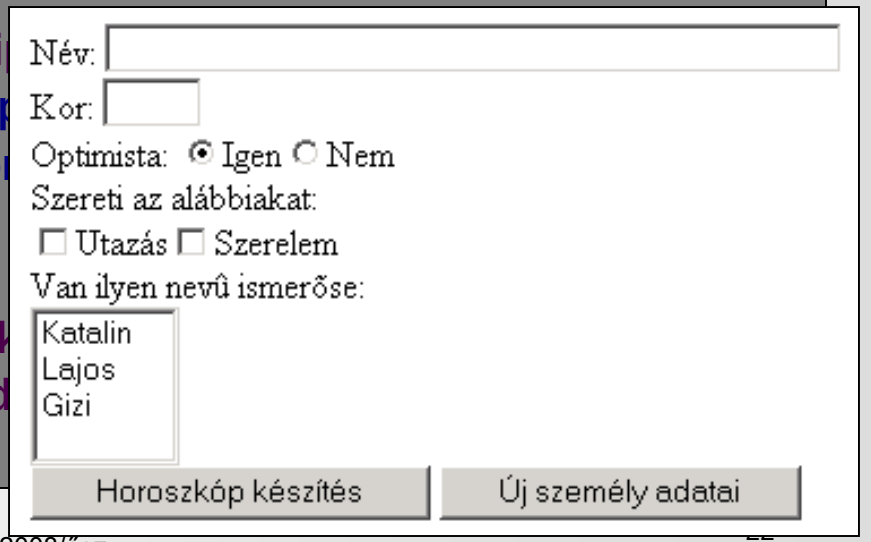
A képernyőn nyomógombokkal legyen lehetőség az adatok elküldésére, illetve az űrlap adatainak törlésére.

Az elküldés gombot lenyomva a fenti paraméterek alapján készítsünk el egy személyre szabott horoszkópot, ami formázott HTML dokumentumként a következő oldalon jelenjen meg!

Következő

Adatok bekérését végző űrlap

```
<form action="Kuruzslo" method="post">
      Név: <input type="text" name="nev" size="50"/><br/>
      Kor: <input type="text" name="kor" size="3"/><br/>
      Optimista:
      <input type="radio" name="optimista" value="igen" checked="checked"/>Igen
      <input type="radio" name="optimista" value="nem"/>Nem<br/>
      Szereti az alábbiakat:<br/>
      <input type="checkbox" name="utazas"/>Utazás
      <input type="checkbox" name="szerelem"/>Szerelem<br/>
      Van ilyen nevű ismerőse:<br/>
      <select name="ismeros" multiple="multi
            <option value="Katalin">Katalin</op
            <option value="Lajos">Lajos</optio
            <option value="Gizi">Gizi</option>
      </select><br/>
      <input type="submit" value="Horoszkóp k
      <input type="reset" value="Új személy ad
</form>
```



Kuruzslo szervlet

```
public class Kuruzslo extends HttpServlet {
   private String veletlenNev(String nevek[]) {
      if (nevek == null || nevek.length == 0) return "önmaga";
      else return nevek[new Random().nextInt(nevek.length)];
   }

   protected void doPost(HttpServletRequest request,
                         HttpServletResponse response) throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      boolean optimista = request.getParameter("optimista").equals("igen");
      String[] ismeros = request.getParameterValues("ismeros");
      out.println("<html><head><title>Tudományos horoszkóp</title></head><body>");
      out.println("<h2>Kedves "+ request.getParameter("nev") +"!</h2>");
      if (request.getParameter("utazas") != null) {
         if (optimista) out.println("Utazás vár önre "+ veletlenNev(ismeros) +" társaságában.");
         else out.println("Utazásra a közeljövőben nem lesz lehetősége.");
      }
   ...
```

Kuruzslo szervlet

Szervlet folytatása

```
    if (request.getParameter("szerelem") != null) {
      if (ismeros == null) out.println("Legyen nyitottabb a világ irányába és meglátja, hamarosan");
        else out.println("Talán meglepő, de hamarosan "+ veletlenNev(ismeros)+ " személyében ");
      out.println(" rátalál élete párjára.");
      if (!optimista) out.println("A kapcsolatnak azonban hamarosan végeszakad.");
        else {
          try {
            if (Integer.parseInt(request.getParameter("kor")) > 50)
              out.println("Gyermekük azonban ebből a kapcsolatból nem lesz.");
            else out.println("Kapcsolatukat akár gyermekáldás is megkoronázhatja.");
          } catch (Exception e) {};
    }
    out.print("</body></html>");
    out.close();
}
}
```

Témakörök

XHTML Form elemek

Szervlet kérés paraméterei

Form adatok feldolgozása

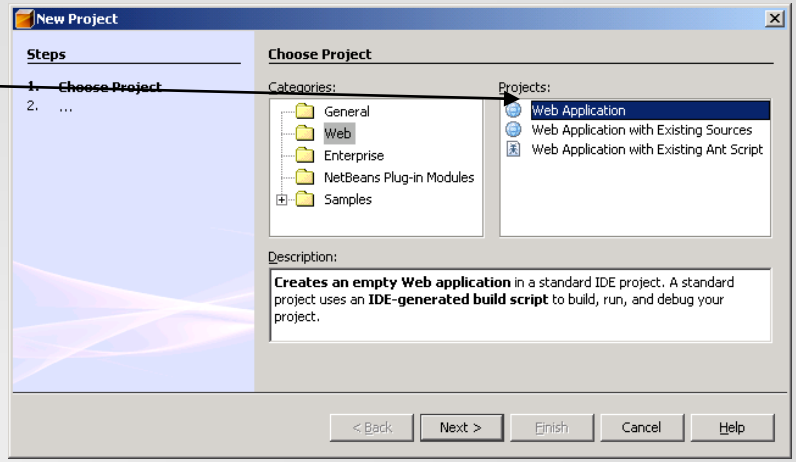
NetBeans támogatás

Netbeans beépített Tomcat szerver

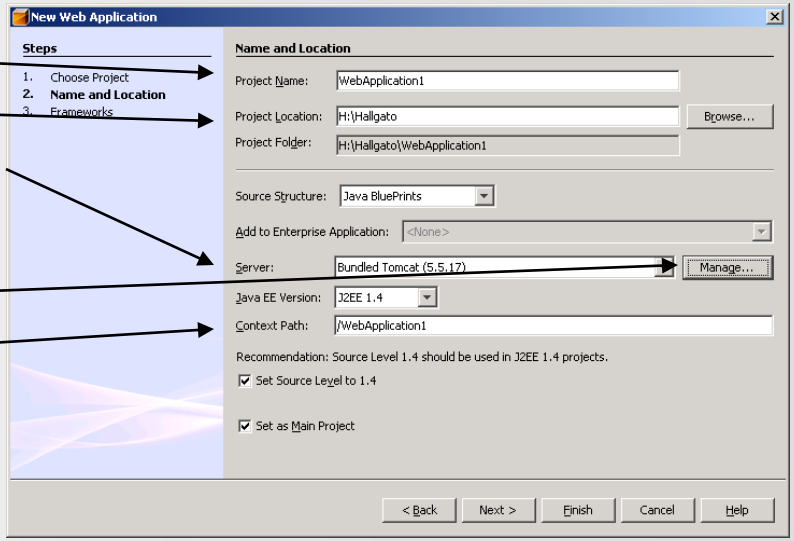
- A Netbeans 5.5.1 letölthető és telepíthető egy beépített Tomcat 5.5.17 webkonténerrel is
- A szerver konfigurációs állományai a `c:\Document and settings\Java\.netbeans\5.5.1\apache-tomcat-5.5.17_base` könyvtárban érhetők el.
  A virtuális gépeken a szerver a 8084 porton érhető el
- Egy web project készítése során a fejlesztői környezet automatikusan készít egy ehhez tartozó context bejegyzést alapértelmezett beállításokkal
- Web alkalmazás fejlesztése során a project könyvtár web alkönyvtárában találhatóak a már megismert állományok
- A szerver elindítását/leállítását a fejlesztői környezet automatikusan elvégzi a program indításakor. Nyomkövetés esetén szintén nincs szükség kézi beavatkozásra. Az eddig megismert módon lehetőség van töréspontok elhelyezésére is

Új web project készítése

- File/New Project menüpontot kiválasztva
  - Web kategória
  - Web application project típus

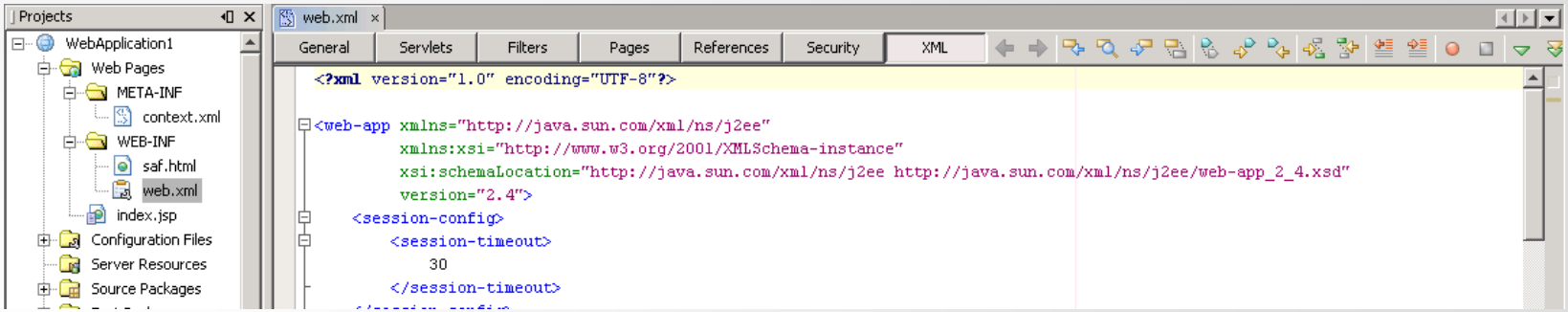


- Számunkra érdekes további beállítások
  - Project neve
  - Project főkönyvtára
  - Szerver kiválasztása az automatikus telepítéshez
  - Beépített szerver beállításai
  - Szerveren létrehozandó kontextus megnevezése



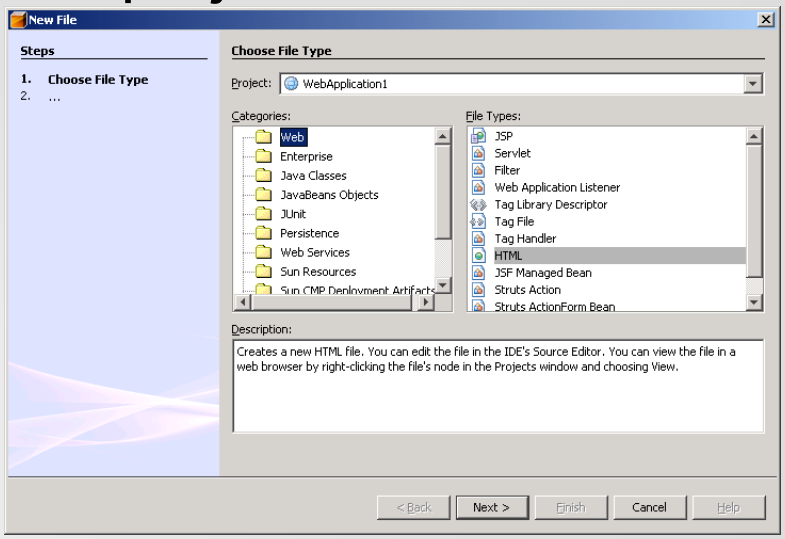
Konfigurációs állományok kezelése

- Új project készítése során a NetBeans létrehozza az új kontextushoz szükséges könyvtárszerkezetet és konfigurációs állományokat
- Ezek elérhetők a Projects ablakban található fa szerkezeten keresztül is:
  `META-INF/context.xml`
  `WEB-INF/web.xml`
- A `web.xml` állomány kezelhető egyszerű XML tartalomként is, vagy az alábbi füleken keresztül:

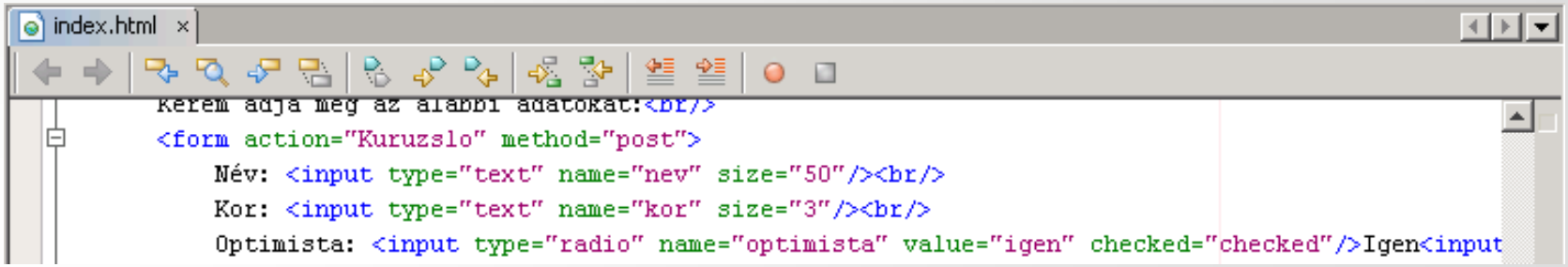


HTML szerkesztés

- HTML állományokat csatolhatunk a projecthez
  Az így létrehozott állományok a kontextus gyökérkönyvtárába kerülnek, ezért egyszerűen tudunk rájuk hivatkozni

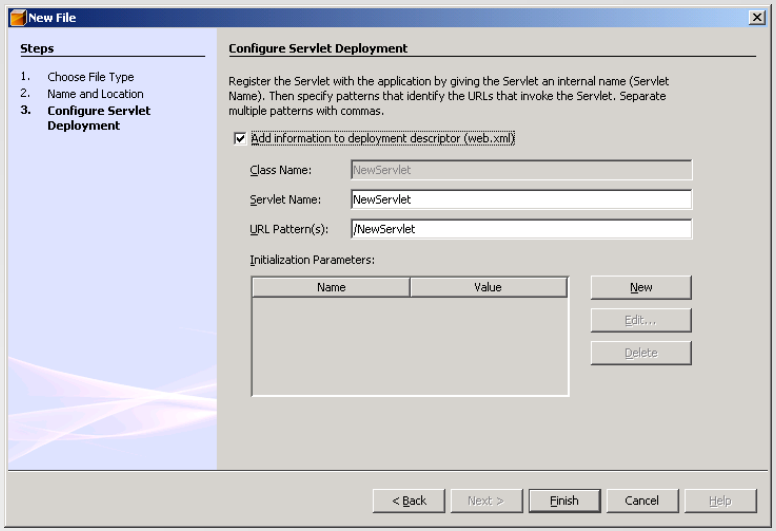


- Beépített HTML szerkesztő
  A megnyitott HTML fájlok szerkesztését természetesen a fejlesztői környezet támogatja



Új szervlet létrehozása

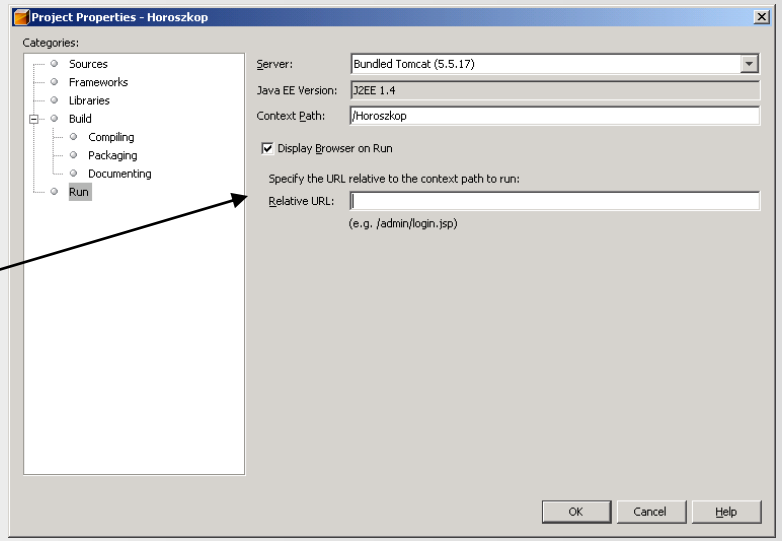
- Új szervlet létrehozása
  További beállítási lehetőség, hogy igényeljük-e az automatikus telepítést, és ha igen, akkor milyen néven kerüljön a kontextusba



- Szervlet sablon
  A létrehozott alap szervlet megvalósítja a doGet(...) és a doPost(...) metódust, (a napi gyakorlatban gyakran használt módon) mindkettő meghív egy processRequest(...) metódust. Így a beérkező kérés tényleges típusától függetlenül csak ezt kell megvalósítani.
  A generált kód ezen túl a getServletInfo() metódust is megvalósítja

Web alkalmazás futtatás

- Az eddigiekben megismert módon a futtatás a „Run/Run main project” menüpont segítségével történik
- Ez automatikusan lefordítja az osztályokat, telepíti őket a kontextusba, majd betölti a kezdőoldalt a böngészőben
- Nyomkövetés szintén a megismert módon, a „Run/Debug maing project” menüpont segítségével érhető el. Töréspontok elhelyezése a konzol alkalmazásokhoz hasonlóan történik. Ha egy kérés kiszolgálása ilyen sorra fut, a program futása megszakad, és a NetBeans nyomkövetése indul el
- A kezdőoldal módosítható a Project tulajdonságai között a „Run” pontban



Gyakorló feladat

Készítse el a matematikai táblázatokat generáló szervlet új változatát:

A,

- A kezdő statikus XHTML oldalon található első űrlapon legyen lehetőség megadni:
  - szövegbeviteli mezőben egy számot
  - rádiogombokkal a mértékegységet (fok vagy radián)
  - listából kiválasztva a szögfüggvényt (sin, cos, tan)
- Az adatok elküldését követő oldalon jelenjen meg a keresett érték

B,

- A kezdőoldalon található második űrlapon egy, a számokat 1-10-ig tartalmazó listából lehessen kiválasztani tetszőleges számokat (legalább egyet kötelező)
- Az adatok elküldését követő oldalon jelenjen meg az egyszer már megvalósított 10x10-es szorzótábla, viszont azok a mezők, amelyek sora vagy oszlopa ki lett választva, jelenjenek meg piros színnel

Házi feladat

A,

Készítsen egy valutaváltó webes alkalmazást

- Készítsen egy Valuta osztályt az alábbi mezőkkel, majd 4-5 ilyen elemmel töltsön fel egy tömböt:
  - rövidítés (EUR)
  - HUF-hoz viszonyított árfolyam (250)
- A dinamikusan generált kezdőoldalon a tömb alapján két rádiógombcsoportban jelenjenek meg a valuta nevek
- A felhasználó megad egy összeget, és hogy miről mire szeretne átváltani
- Egy checkbox segítségével lehessen beállítani, hogy felszámolunk-e kezelési költséget (ha igen, akkor ez mindig fixen 1000HUF)
- Az adatok elküldése után jelenjen meg a fizetendő összeg

B,

Készítsen egy kábeltv szolgáltató számára webalapú csatornainformációs szolgáltatást!

- Készítsen egy Csatorna osztályt az alábbi mezőkkel:
  - csatorna neve
  - csatorna nyelve
  - korhatár
- Készítsen egy CsomagTarolo osztályt, amelyik tetszőleges formában tárolja az éppen elérhető csatornákat, illetve azt, hogy az aktuális csomagok (induló, családi, extra) ezek közül melyeket tartalmazzák!
- Készítsen webes alkalmazást, ahol a felhasználó megadja a nevét, életkorát, illetve milyen nyelveket beszél. Az adatok elküldése után a következő oldalon jelenjen meg, hogy az egyes csomagok választása esetén hány számára érdekes csatornát talál.

Ajánlott irodalom

Az óra anyagához kapcsolódó irodalom

- Nyékyné Gaizler Judit: Java 2 útikalauz programozóknak 1.3 II.; ELTE TTK Hallgatói alapítvány, Budapest

  469 – 480. o.
- Jason Hunter: Java szervletek programozása; O'Really-Kossuth, Budapest, 2002

  87 – 179. o.
- The J2EE 1.4 Tutorial – Chapter 11: Java Servlet 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