Java Server Pages

JSP technika alapjai

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

Témakörök

JSP architektúra

Scriptletek elhelyezése a kódban

JSP direktívák

JSP működése

JSP kialakulásának okai

- A szervletek egy jól használható technikát jelentenek a dinamikus weboldalak készítése terén, a gyakorlatban azonban számos probléma merült fel:
  - A szervlet kódja már egy közepes méretű weboldal esetén is áttekinthetetlen lett a sok HTML formázási parancstól
  - A szerveroldali programok készítése és a megjelenő weboldalak küllemének kialakítása általában két külön személyt igényel. Amennyiben a szervletből generáljuk a HTML kódot, ez nagyon nehezen oldható meg
  - A szervlet kódja miatt nincs lehetőség külső HTML szerkesztő programok használatára sem
- Ezekre a megoldást a szerveroldali program és a felhasználó felé küldött HTML oldal kódjának szétválasztása jelenti
- Ideális esetben az előbbit a már megismert szervletek, az utóbbit pedig JSP oldalak segítségével oldják meg. Ez az úgynevezett Model 2 architektúra

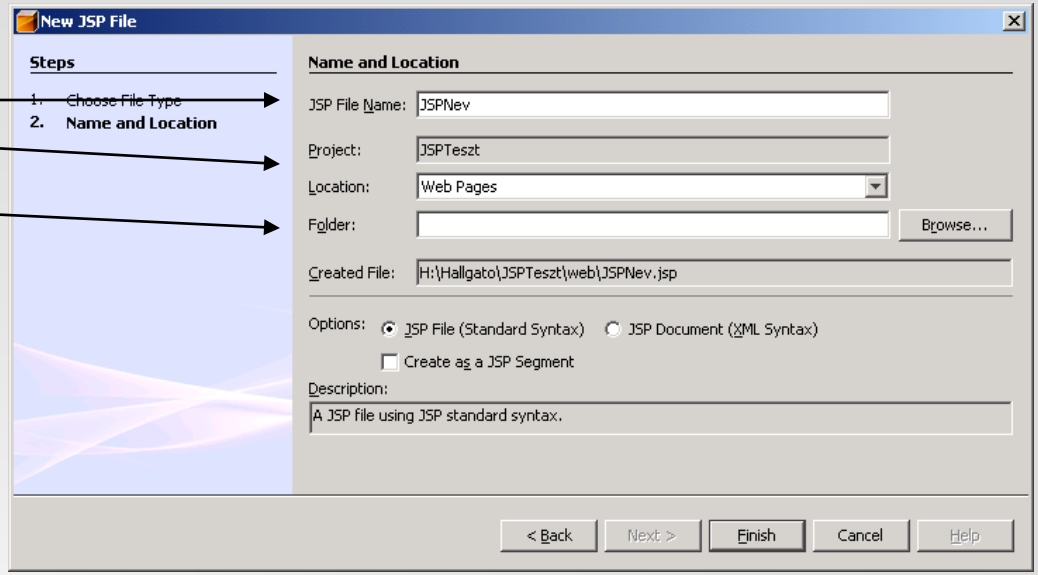
JSP architektúra

- A JSP oldalak felépítése hasonló a gyakorlatban már bevált scriptnyelvek (PHP stb.) működéséhez
- A JSP felfogható egy „kifordított” szervletként:
  - Szervlet – java kód, amely a kimenetre küld XHTML kódokat
  - JSP – egy XHTML oldal, ami a dokumentumba ágyazva tartalmaz Java kódrészleteket is
- A két technika tehát nem választható el egymástól, a programkód – HTML tartalom mennyisége alapján célszerű választani közülük
  Technikailag a különbség elenyésző, a JSP lapok tulajdonképpen ugyanúgy szervletként működnek a konténerben (lásd később részletesebben)
- A project tervezés során eleve célszerű az üzleti logikát szervletekkel megvalósítani (összetett programozói feladat), a felhasználó felé irányuló megjelenítési szintet pedig JSP oldalak segítségével megoldani (designer feladat, minimális programozással)

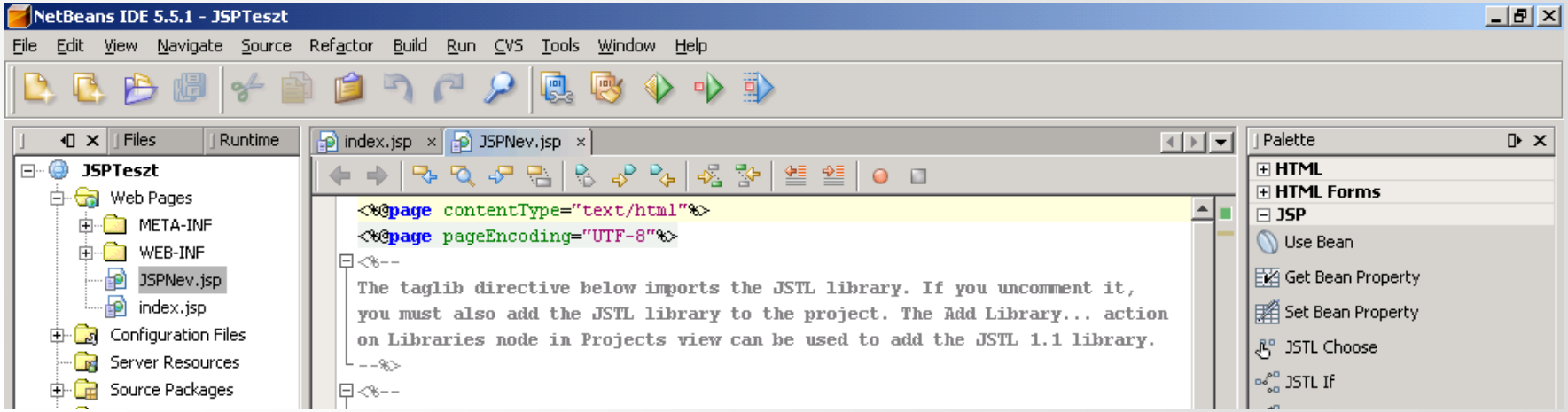
JSP oldal készítése NetBeans alatt

File/New file/Web/JSP

- JSP állomány neve
- Elhelyezkedése a csomagban
- Csomagon belül alkönyvtár



File/New file/Web/JSP



JSP oldal telepítése a szerveren

- Önállóan működő szerver esetén
  - Az elkészített JSP állomány bemásolható a `h:\HallgatoWeb` könyvtárba
  - Ezt követően elérhető az alábbiak szerint: `http://localhost/hweb/JSPNev.jsp`
  - A Tomcat szerver a kiterjesztés alapján tudja, hogy az állományt nem letölteni kell, hanem szervletként futtatni és a kimenetét elküldeni a kliens felé
- NetBeans beépített szerver esetén
  - Külön telepítésre nincs szükség, a JSP állomány eleve a webalkalmazás könyvtárában jön létre
  - Futtatása hasonló az eddig megismertekhez: `http://localhost:8084/JSPTeszt/JSPNev.jsp`
  - Töréspontok elhelyezése, soronkénti futtatás az eddig megismertekhez hasonlóan itt is működik

Témakörök

JSP architektúra

Scriptletek elhelyezése a kódban

JSP direktívák

JSP működése

Scriptletek beágyazása

- Egy JSP lap felfogható egy XHTML oldalként, amely a dinamikus tartalom generálása érdekében tartalmazhat Java kódrészleteket. Ezeket nevezzük scriptleteknek
- A scriptletek <% kezdő és a %> lezáró jelek közé kerülhetnek a JSP oldal tetszőleges pontján
  Az alábbiakat tartalmazhatják:
  - Változó deklaráció
  - Primitív változókkal, objektumokkal végzett műveletek
  - A lokálisan deklarált változókon túl van lehetőség a környezet által biztosított objektumokhoz való hozzáférésre is (következő oldalon részletesebben)
  - Ciklus/elágazás
    Ebben az esetben nem kötelező a megnyitott blokkot a scriptleten belül lezárni, azonban ügyelni kell arra, hogy a teljes JSP lap tekintetében a blokkok megfeleljenek a Java nyelv szabályainak

Scriptletekben előre definiált változók

- *HttpServletRequest* request
  A már megismert, kérési adatokat tartalmazó objektum
- *HttpServletResponse* response
  A már megismert, a válasz adatait tartalmazó objektum
- *JspWriter* out
  Közvetlenül a kimenet írására használható. Működése a *PrintWriter* objektumokhoz hasonló, csak máshogy pufferel
- *HttpSession* session
  A már megismert menetkövető objektum. A kérésből automatikusan lekérdezésre kerül, azonnal használható (beállítható, hogy ne jöjjön létre, hiszen bizonyos alkalmazásoknál ez felesleges)
- *ServletContext* application
  A webalkalmazást reprezentáló objektum, a konténer paramétereihez enged hozzáférést
- *PageContext* pageContext
  A kiszolgáló adataihoz enged hozzáférést

Egyszerű scriptlet példa

Napszaknak megfelelő üdvözlet

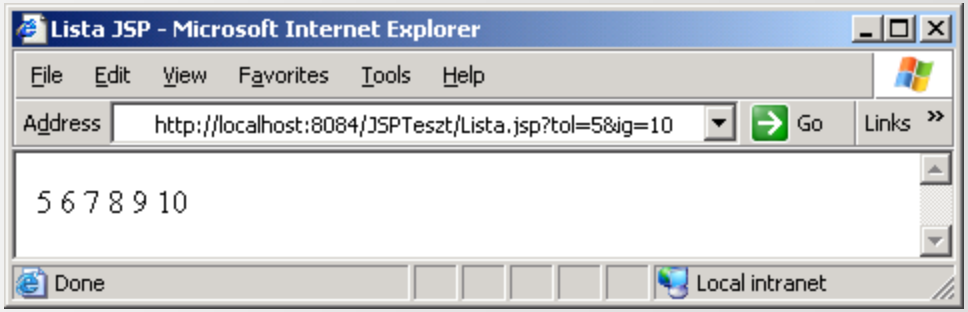
```
<html>
   <head><title>Illedelmes JSP</title></head><body>
        <%
        int ora = java.util.Calendar.getInstance().get(java.util.Calendar.HOUR_OF_DAY);
        if (ora < 7) {%>
           Jó reggelt!
        <%}  else if (ora > 20) {%>
           Jó estét!
        <%}  else {%>
           Jó napot!
        <% } %>
   </body>
</html>
```

Egyszerű scriptlet példa

Listázzuk a „tol” és „ig” paraméterek közötti számokat

```
<html>
   <head><title>Lista JSP</title></head>
   <body>
      <%
         for(int i = Integer.parseInt(request.getParameter("tol"));
             i <= Integer.parseInt(request.getParameter("ig"));
             i++) {
           out.println(i);
         }
      %>
   </body>
</html>
```

Kimenete:



További lehetőségek

- Ha csak egy kifejezés (változó) értékét kell kiírni az XHTML oldalba, használható a <%= kifejezés %> tag is
- Változó deklarációjához használható a <%! ... %>
- Pl. írjuk ki 1-től 10-ig a számok kétszeresét és négyzetét

```
<html>
   <head><title>Lista JSP</title></head>
   <body>
      <table>
         <tr><td>Szám</td><td>Kétszeres</td><td>Négyzet</td></tr>
         <% for(int i = 1; i <= 10; i++) { %>
            <tr>
               <td><%= i %></td> <td><%= i*2 %></td> <td><%= i*i %></td>
            </tr>
         <% } %>
      </table>
   </body>
</html>
```

Témakörök

JSP architektúra

Scriptletek elhelyezése a kódban

JSP direktívák

JSP működése

JSP direktívák

- A direktívák segítségével van lehetőség a JSP oldal (illetve a háttérben működő szervlet) működésének befolyásolására
- Míg a JSP lap a küldendő tartalomról szól, addig a direktívák határozzák meg, hogy milyen tartalomtípust küldjön az oldal, milyen csomagokat importáljon, hogyan pufferelje a kimenetet, miként kezelje a hibákat stb.
- Direktívák általános alakja az alábbi:
  <%@direktíva attribútum=”érték” %>
  A direktíva nevének megadásán túl tehát értéket lehet adni egy attribútumának is
- Példa direktíva használatára:
  <%@page import=”java.util.*” %>

A page direktíva attribútumai

- contentType
  Megadja, hogy milyen legyen a generált lap tartalomtípusa
  <%@page contentType="text/plain" %>
- import
  Megadja, hogy az oldal fordítása során milyen külső csomagokat kell importálni (vesszővel elválasztva többet is meg lehet adni egyszerre)
  <%@page import="java.util.*" %>
- session
  Értéke lehet „true" vagy „false", azt határozza meg, hogy az oldalon belül szükség van-e a session változóra (alapértelmezett értéke igen, ez azonban gyakran felesleges)
  <%@page session="true" %>
- errorPage
  Megadja, hogy melyik lapot kell megjeleníteni, ha a kiszolgálóhoz ér a lap által dobott kivétel
  <%@page errorPage="/hibakezelo.jsp" %>

Egyéb direktívák, tag-ek

- include
  Még a fordítás előtt, mint statikus állományt beszúrja a megadott állományt a JSP oldalba. Beszúrható másik JSP oldal(részlet) is, ilyenkor a fordításkor a két kód együtt fordítódik le
  <%@include file="/fejlec.html" %>
- jsp:include
  Az előzőhöz hasonló a feladata, azonban ez nem a fordítás közben fűzi be a megadott fájlt, hanem a kérés kiszolgálása során. Ilyenkor a szerver „kiértékeli" a megadott erőforrást es az eredményét beszúrja az oldalba (a kérésben érkező paraméterek továbbításra kerülnek, és van lehetőség új paraméterek hozzáadására is)
  <jsp:include page="/szemelyiadatok.jsp" >
  <param name="neptun" value="uccg17" />
  </jsp:include>
- jsp:forward
  A kérés továbbítása egy másik erőforrás irányába
  <jsp:forward page="/sikertelenbelepes.jsp" />

Témakörök

JSP architektúra

Scriptletek elhelyezése a kódban

JSP direktívák

JSP működése

JSP működése a háttérben – JSP oldal

Hogyan működik a valóságban alábbi JSP oldal?

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.Calendar"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                              "http://www.w3.org/TR/html4/loose.dtd">
<html>
   <head><title>Illedelmes JSP</title></head><body>
      <%
      int ora = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
      if (ora < 7) {%>
         Jó reggelt!
      <%}  else if (ora > 20) {%>
         Jó estét!
      <%}  else {%>
         Jó napot!
      <% } %>
   </body>
</html>
```

JSP működése a háttérben – Szervlet

Az oldal első hozzáférésekor előáll és lefordul az alábbi szervlet:

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import java.util.Calendar;

public final class Illedelmes_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent {

  private static java.util.List _jspx_dependants;

  public Object getDependants() {
    return _jspx_dependants;
  }

  public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws java.io.IOException, ServletException {

...
```

JSP működése a háttérben – Szervlet

Szervlet kódjának folytatása (változók beállítása):

```
JspFactory _jspxFactory = null;
PageContext pageContext = null;
HttpSession session = null;
ServletContext application = null;
ServletConfig config = null;
JspWriter out = null;
Object page = this;
JspWriter _jspx_out = null;
PageContext _jspx_page_context = null;
try {
  _jspxFactory = JspFactory.getDefaultFactory();
  response.setContentType("text/html;charset=UTF-8");
  pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
  _jspx_page_context = pageContext;
  application = pageContext.getServletContext();
  config = pageContext.getServletConfig();
  session = pageContext.getSession();
  out = pageContext.getOut();
  _jspx_out = out;
```

. . .

JSP működése a háttérben – Szervlet

Szervlet kódjának folytatása (ismerős rész):

```
out.write("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"\n");
out.write("   \"http://www.w3.org/TR/html4/loose.dtd\">\n");
out.write("<html>\n");
out.write("<head><title>Illedelmes JSP</title></head><body>\n");
int ora = java.util.Calendar.getInstance().get(java.util.Calendar.HOUR_OF_DAY);
if (ora < 7) {
   out.write("\n");
   out.write("Jó reggelt!\n");
}  else
   if (ora > 20) {
      out.write("\n");
      out.write("Jó estét!\n");
   }  else {
      out.write("\n");
      out.write("Jó napot!\n");
   }
out.write("\n");
out.write("</body>\n");
out.write(" </html>\n");
```

. . .

JSP működése a háttérben – Szervlet

Szervlet kódjának folytatása (hibakezelés stb.):

```
    } catch (Throwable t) {
      if (!(t instanceof SkipPageException)){
        out = _jspx_out;
        if (out != null && out.getBufferSize() != 0)
          out.clearBuffer();
        if (_jspx_page_context != null) _jspx_page_context.handlePageException(t);
      }
    } finally {
      if (_jspxFactory != null) _jspxFactory.releasePageContext(_jspx_page_context);
    }
  }
}
```

Az így létrejött szervlet fogja valójában fogadni az összes, a JSP laphoz irányuló kérést.

Amennyiben a JSP lap megváltozott, akkor (beállítástól függően) a webkiszolgáló automatikusan újra elkészíti és lefordítja ezt a szervletet

Gyakorló feladat

Készítse el az alábbi alkalmazást JSP oldalak segítségével:

- A nyitóképernyőn jelenjenek meg az alábbi táblázatok:
  - az egyik 360 sorban tartalmazza fokonként a megfelelő
    - szinusz értéket
    - koszinusz értéket
    - tangens értéket
  - a másik egy 10x10-es szorzótábla
- A generált oldal alján legyen egy form, ahol meg lehet adni egy fokot, illetve két szorzandót. A submit gomb lenyomására jelenjen meg ugyanez az oldal (esetleg ennek egy másolata a többlet kóddal kiegészítve), amely
  - a 360 soros táblázatból csak a szükséges sort,
  - a szorzótáblának mind a 100 elemét, de a választott sort és oszlopot más héttérszínnel

  jeleníti meg. Adjon lehetőséget új számok megadására is!

Házi feladat

JSP oldal segítségével készítse el az alábbi alkalmazást:

Az első oldalon jelenjen meg egy formázott fejlécben a program neve (Szám kitaláló), készítője, a szerver címe és a pontos idő. Még ezen az oldalon legyen lehetőség az alábbi két játék közül választani:

A, a gép által kitalált számot kell eltalálnunk

- A kezdőoldalon erre a linkre kattintva a következő oldalon jelenjen meg két szám, amelyek között a gép választott egy véletlen számot (a minimum 1..100, a maximum 100..200 között legyen, a keresett pedig ezek között)
- Ez alatt legyen egy szövegbeviteli mező, amelyben a felhasználó egy tippet tud megadni. A szám megadása után az „Elküldés” gombra kattintva a következő oldalon jelenjenek meg ugyanezek az adatok, kiegészítve azzal, hogy a keresetthez képest kisebb vagy nagyobb az utoljára megadott
- A program a session objektumba a szükséges paramétereken (min, max, keresett szám, tippek száma stb.) túl folyamatosan tárolja el a felhasználó tippjeit is, egy formázott táblázatban ezeket is jelenítse meg minden lépés után, a legközelebbit más színnel rajzolva. Alatta az összes tipp száma legyen.
- Amennyiben a felhasználó eltalálta a számot, az oldal alján jelenjen meg egy plusz mező, ahol megadhatja a nevét. Az itt található „Elküld” gombra kattintva a következő oldalon jelenjen meg egy oklevél a nevével és a játék adataival

B, ugyanez a játék, csak a gép tippeljen

- A felhasználó megad egy számot, majd a gép tippjeire válaszol, hogy kisebb vagy nagyobb. Oklevél nyomtatása nélkülözhető

Ajánlott irodalom

Az óra anyagához kapcsolódó irodalom

- Nyékyné Gaizler Judit: J2EE útikalauz Java programozóknak; ELTE TTK Hallgatói alapítvány, Budapest

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

  546 – 577. o.
- 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