Java technológiák – 3. előadás Szervletek (folytatás)

ANTAL Margit

Sapientia - EMTE

2010

A 3. előadás célja

- Vezérlési komponens tervezése
- HTML űrlap készítése
- Űrlap adatainak küldése
- Vezérlési komponensek megvalósítása
- Szervletek közötti kommunikáció
- Ügyfélinformációk
- ServletContext

Vezérlési komponestípusok

- felhasználói adatok feldolgozása,
- navigációt segítő komponensek,
- adatok előkészítése más komponenseknek.

A webalkalmazás logikai szerkezete

```
/----index.jsp
   |
   |--list_courses.view
   |
   |--add_course.jsp
   |
   |--add_course.do
   |
   |--success.view
   |
   |--error.view
```

A webalkalmazás fizikai szerkezete I

A webalkalmazás fizikai szerkezete II

```
|     |---SuccessServlet.class
|     |
|     |---ErrorServlet.class
|
|---model/
      |---Course.class
```

A WEB-INF mappa

- kötelező módon jelen van minden webalkalmazásban,
- az itt elhelyezett erőforrások direkt módon nem érhetők el.

HTTP status 404

`http://localhost:8080/app/WEB-INF/web.xml`

OK

`http://localhost:8080/app/index.jsp`

Egyszerű HTML űrlap

Űrlap

```
<form action='URL TO CONTROLLER'
          method='GET vagy POST'>
<!-- urlap elemei -->
</form>
```

Példa űrlapra

```
<form action='add_course.do' method='POST'>
Course      : [szovegdoboz]
Description: [szovegdoboz]
Price       : [szovegdoboz]
[Submit nyomogomb]
</form>
```

A `form` elem

- `action` tulajdonság: egy URL a webkonténerben meghívandó komponensre,
- `method` tulajdonság: a HTTP metódus neve: `GET` vagy `POST`.

Új tanfolyam űrlap

```
<form action="add_course.do" method="POST">
  Name: <input type="text" name="name"
                           size="50">
  <br><br>
  Description: <input type="text"
        name="description" size="75">
  <br><br>
  Price: <input type="text" name="price"
                            size="10">
  <br><br>
  <input type="submit" value="Add Course">
</form>
```

Űrlap a böngészőben

Figure: Egyszerű űrlap

HTML űrlapok

Megjegyzések

- Egy weboldal több űrlapot is tartalmazhat..
- Az űrlapokat nem lehet egymásba ágyazni.
- Deklarálhatunk olyan űrlapot is, amely nem tartalmaz sem beviteli mezőt, sem pedig Submit típusú nyomógombot. Az ilyen űrlapok küldését Javascript kóddal kell megoldani.

Szövegdoboz komponens

```
Year: <input type='text' name='name'/>
```

Legördülő lista komponens

```
Season: <select name='season'>
<option value='Spring'>Spring</option>
<option value='Summer'>Summer</option>
<option value='Fall'>Fall</option>
<option value='Winter'>Winter</option>
</select>
```

Nyomógomb komponens

```
<input type='submit' value='Add Course'/>
```

Űrlapadatok küldése

A böngésző felelőssége

- A HTTP kérés létrehozása felhasználva az űrlapból az `action` tagban szereplő URL-t
- Az űrlap mezők (bevitt, kiválasztott értékek) összegyűjtése és csomagolása a HTTP kérésbe

Űrlapadatok HTTP kérésben

Szintaxis

```
mezoNev1=mezoEretek1&mezoNev2=mezoErtek2&...
```

Példa

```
name=J2SE&price=100&
description=algorithms+data+structures
```

Speciális karakterek

- = : mezőnév = mezőérték,
- & : elválasztó karakter név-érték párok között,
- + : szóköz helyettesítő,
- ? : elválasztó karakter GET kérés URL része és az űrlap adatai között.

HTTP GET kérés

FONTOS!

Az űrlap adatokat az URL tartalmazza ⇒ látszani fog a böngésző címsorában.

```
GET /add_course.do?name=J2SE&description=...
HTTP/1.1
Host: localhost:8080
User-Agent: ...
Accept:
text/xml,application/xml, ...
Accept-Language: en-us,...
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1, utf-8;...
Keep-Alive: 300
Connection: keep-alive
```

HTTP POST kérés

```
POST /add_course.do
HTTP/1.1
...
Ugyanaz, mint a HTTP GET eseteben
...
Referer:
http://localhost:8080/add_course.jsp
Content-Type:
application/x-www-form-urlencoded
Content-Length: 57

name=J2SE&description=algorithms+data
+structures&price=100
```

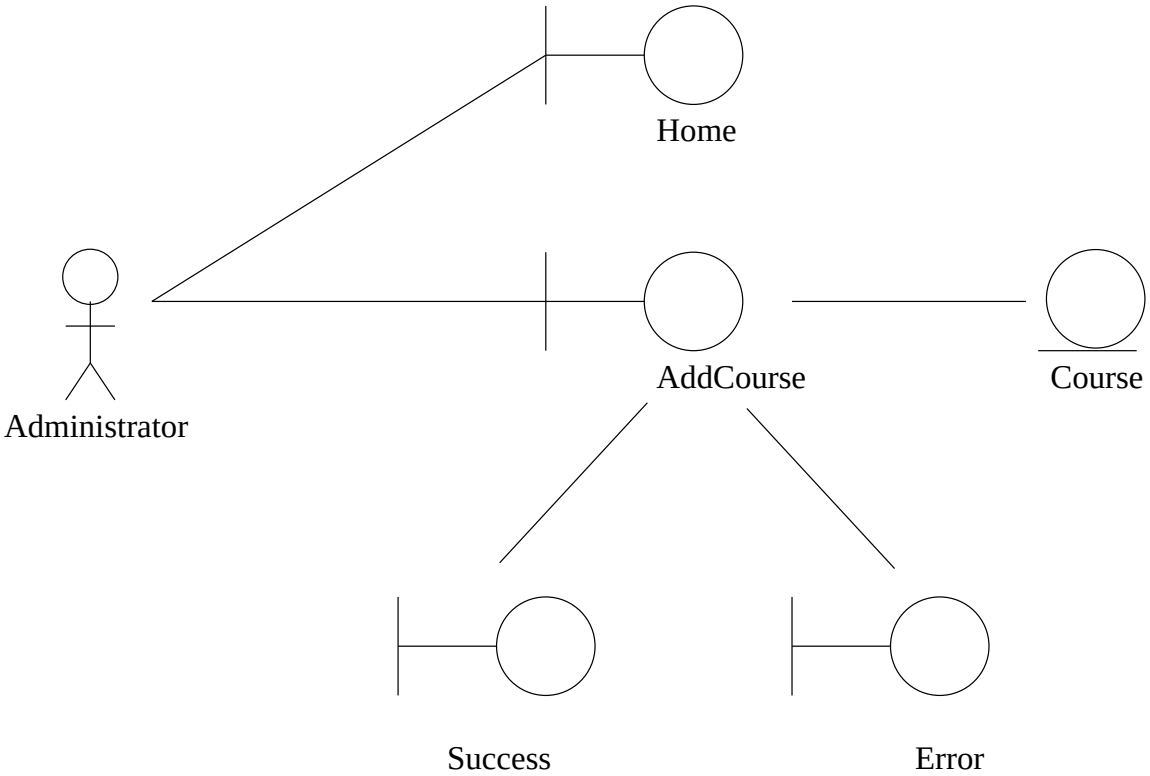
HTTP GET és POST metódusok

- HTTP GET használata:
  - ha a HTTP kérésnek nincs mellékhatása a szerveren,
  - az űrlap kevés adatot tartalmaz,
  - megengedhető a kérés URL-jének lementése (bookmark).
- HTTP POST használata:
  - a HTTP kérés feldolgozása megváltoztatja a szerver állapotát (pl. adatokat tárol egy adatbázisban),
  - az űrlap sok adatot tartalmaz,
  - az űrlap adatait nem jeleníthetjük meg az URL-ben (pl. jelszó).

Vezérlőszervlet megvalósítása

1. űrlap paramétereinek kinyerése,
2. paraméterkonverziók (pl. szöveg →numerikus),
3. paraméterek ellenőrzése,
4. üzleti logika végrehajtása (pl. adatok mentése adatbázisba),
5. vezérlés átadása a következő megjelenítési komponensnek.

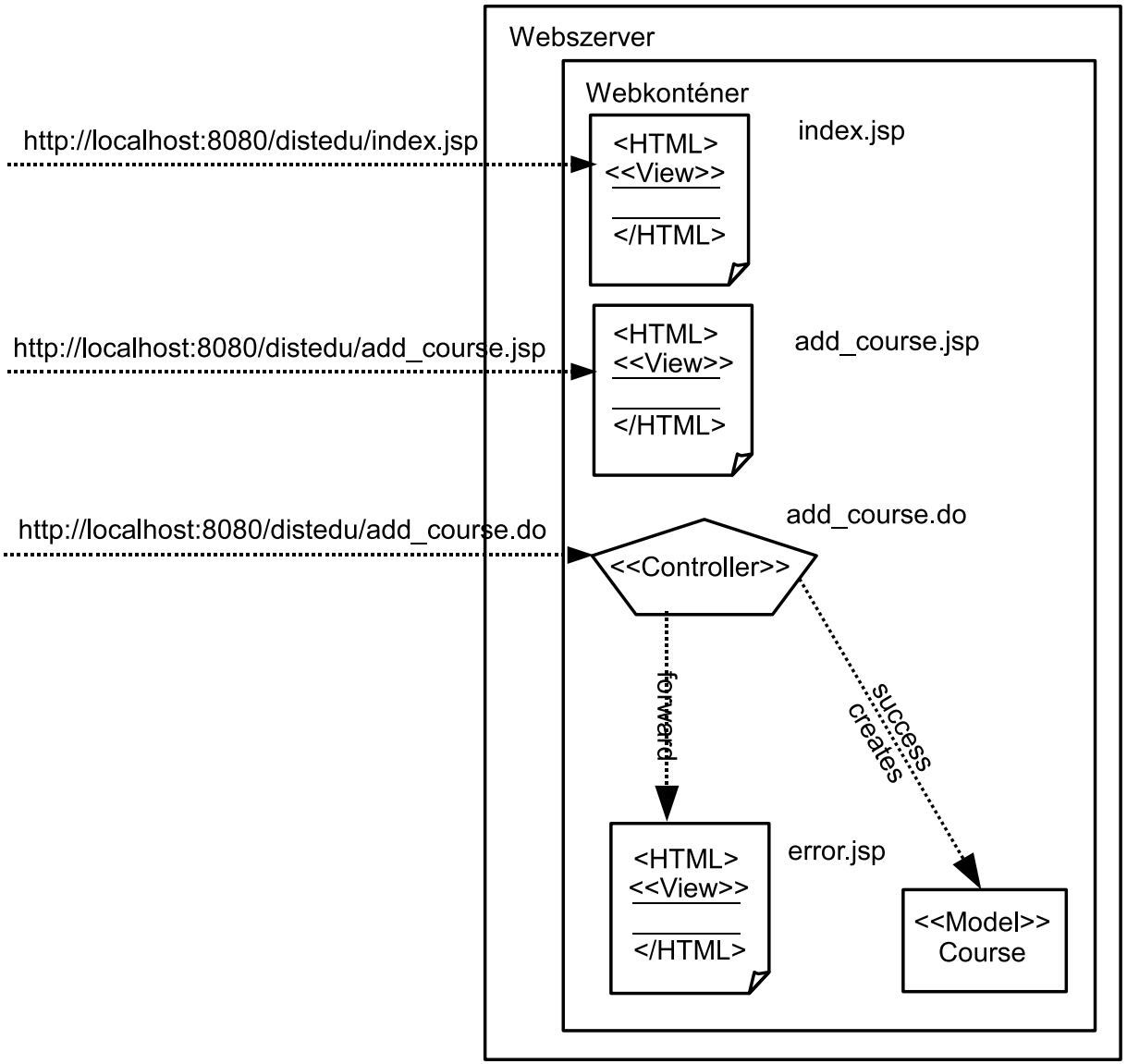
Új tanfolyam felvitele



HTTP kérések sorrendje

1. `http://localhost:8080/distedu/index.jsp`
2. `http://localhost:8080/distedu/add_course.jsp`
3. `http://localhost:8080/distedu/add_course.do`

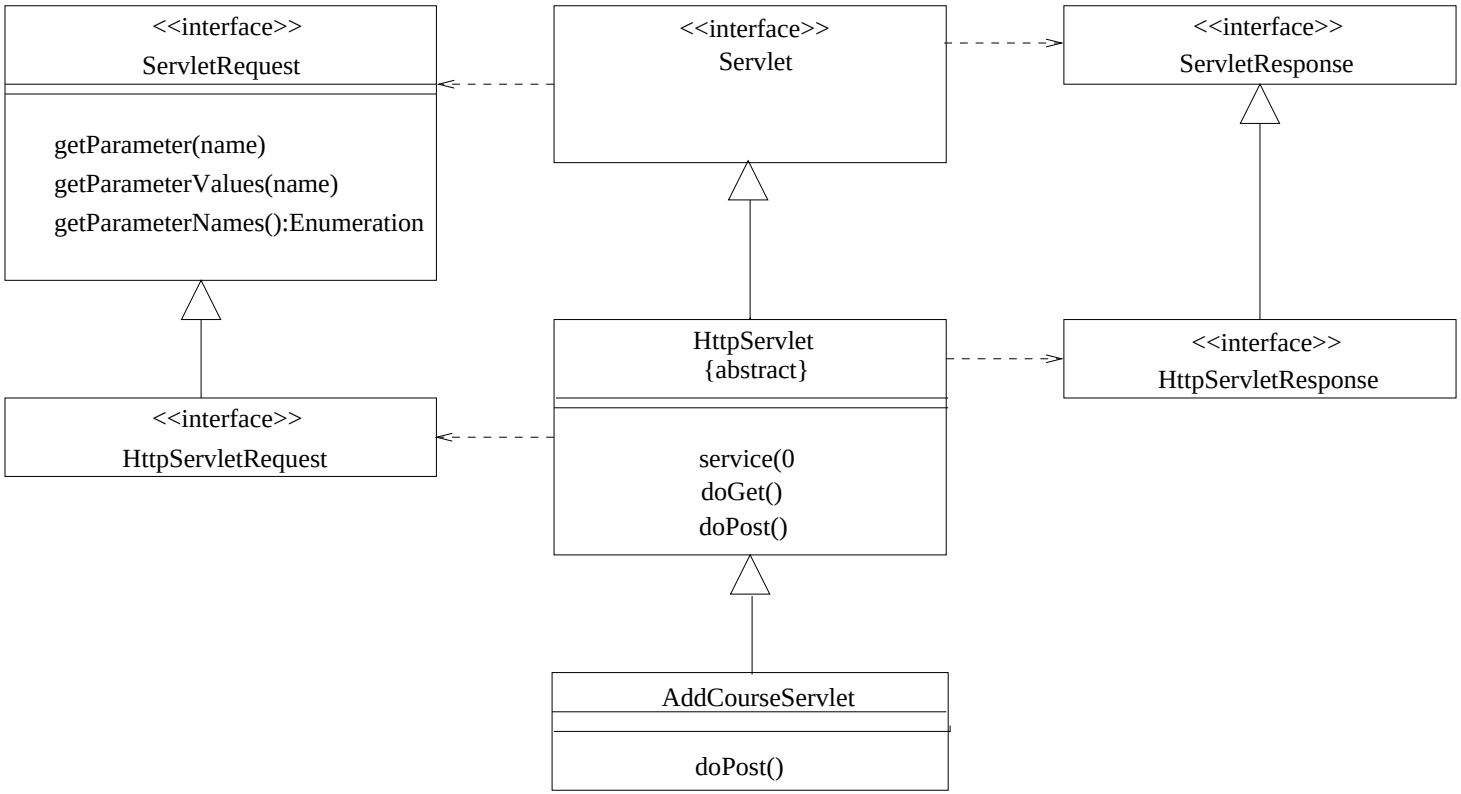
HTTP kérések sorrendje



HTTP kéréseknek megfelelő komponensek

- index.jsp: megjelenítési komponens,
- add_course.jsp: megjelenítési komponens,
- add_course.do : vezérlési komponens, szervlet.

Servlet API paraméterek kinyerése



Paraméterek kinyerése

```
String      getParameter(String name);
Map         getParameterMap();
Enumeration getParameterNames();
String[]    getParameterValues(String name);
```

AddCourseServlet I

```
//Szervlethez szukseges importok
import javax.servlet.http.*;
import javax.servlet.*;

//Mas importok
import java.io.PrintWriter;
import java.io.IOException;
import java.util.*;

//Modell osztalyok
import distedu.model.*;
```

A doPost metodus:

```
List errorMsgs = new LinkedList();
String name=
```

AddCourseServlet II

```
  request.getParameter("name").trim();
String description =
  request.getParameter("description").trim();
String priceStr=
  request.getParameter("price").trim();
double price = 0;
try{
  price = Double.parseDouble(priceStr);
}
catch( NumberFormatException e ){
   errorMsgs.add("Invalid price");
}
if( price < 0 )
  errorMsgs.add("Negative price");

if( !errorMsgs.isEmpty() ){
  request.setAttribute("errorMsgs", errorMsgs);
```

AddCourseServlet III

```
    //Vezerles atadasa
    //a hibaszervletnek
    //ErrorServlet --> error.view
  }
  else{
    Course course=
          new Course(name, description, price);
    request.setAttribute("course", course);
    //Vezerles atadasa
    //a sikeres szervletnek
    //SuccessServlet --> success.view
  }
```

Szervletek közötti kommunikáció

- Ugyanazon webkonténeren belüli szervletek kommunikálhatnak egymással.
- A kommunikáció objektumok segítségével történik.
- Objektum tárolása:
  `request.setAttribute("errorMsgs", errorMsgs)`
- Hozzáférés objektumhoz: `List errorMgs=(List) request.getAttribute("errorMsgs")`

Vezérlés átadása

- `ErrorServlet.java` → `error.view`

AddCourseServlet → ErrorServlet

```
import javax.servlet.RequestDispatcher;
RequestDispatcher view =
request.getDispatcher("error.view");
view.forward(request,response);
```

Új tanfolyam felviteléhez szükséges komponensek

- add_course.jsp : megjelenítési
- add_course.do : verzérlési, `AddCourseServlet.java`
- success.view : megjelenítési, `SuccessServlet.java`
- error.view : megjelenítési, `ErrorServlet.java`

SuccessServlet.java

```
Course course=
 (Course)request.getAttribute("course");
response.setContentType("text/html");
PrintWriter out=response.getWriter();
out.println("<html>");
out.println("<head>");
...
out.println("</head>");
out.println("<body>");
out.println("<p> Successfully added: "+
  course.toString()+</p>");
out.println("</body>");
out.println("</html>");
```

HTTP kérés továbbítása (szerveroldal)

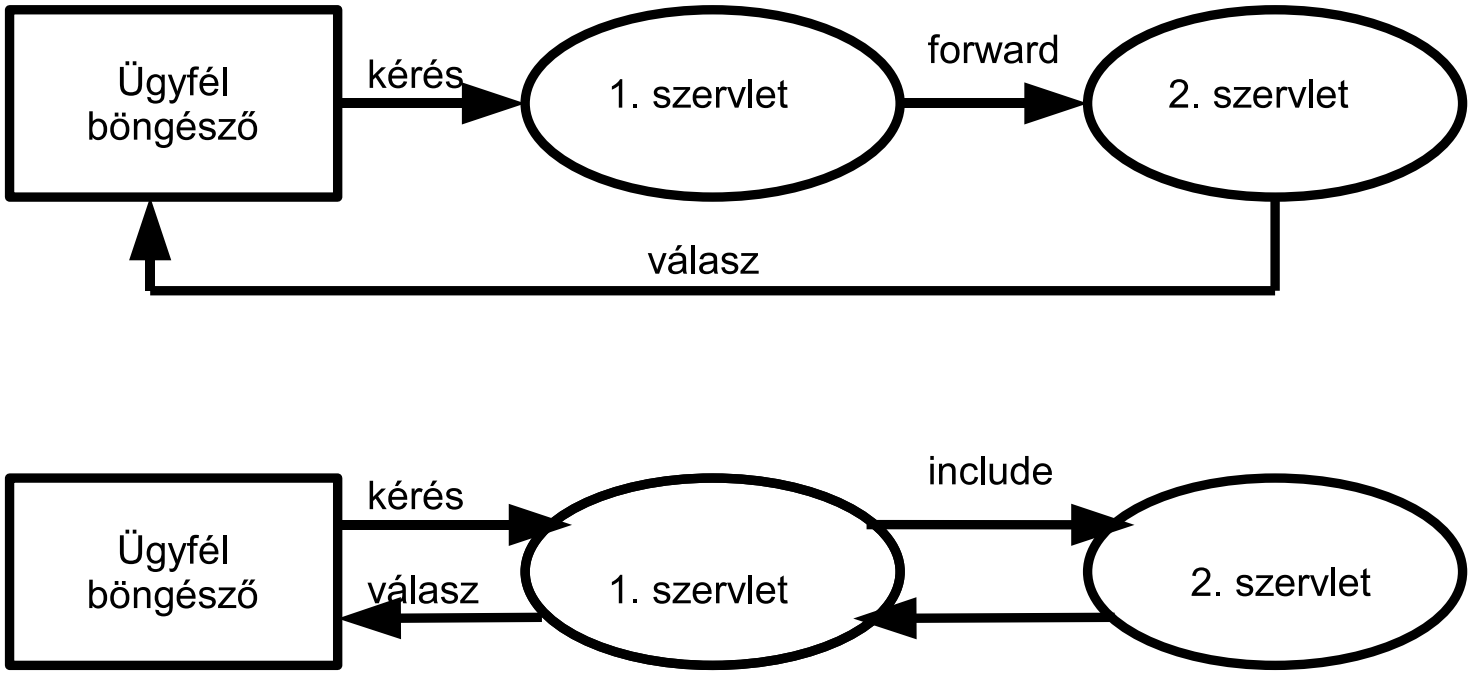


Figure: `forward` és `include`

forward

```
RequestDispatcher comp =
  request.getRequestDispatcher("error.jsp");
```

forward:

```
comp.forward(request,response);
```

1. a kérés továbbítódik a megadott webkomponenshez,
2. a válasz előállítása a fogadó komponens feladata,
3. ha a továbbítás előtt írtunk a válaszpufferbe, ez elvesztődik.

include

```
RequestDispatcher comp =
  request.getRequestDispatcher("error.jsp");
```

include:

```
comp.include(request,response);
```

1. lehetővé teszi más erőforrások tartalmának beillesztését,
2. a beillesztett erőforrás nem változtathatja meg a válasz fejléceket, és nem zárhatja le a választ.

Szervlet inicializáló paraméterek

web.xml

```
<servlet>
    <servlet-name> ...</servlet-name>
    <servlet-class>...</servlet-class>
    <init-param>
        <param-name>... </param-name>
        <param-value>...</param-value>
    </init-param>
    ...
</servlet>
```

web.xml részlet I

```
<servlet>
  <servlet-name>
    ListCoursesServlet
  </servlet-name>
  <servlet-class>
    distedu.view.ListCoursesServlet
  </servlet-class>
  <init-param>
    <param-name>cnames</param-name>
    <param-value>
      C++, J2SE, J2ME, J2EE
    </param-value>
  </init-param>
  <init-param>
    <param-name>cprices</param-name>
    <param-value>
```

web.xml részlet II

```
        100, 100, 150, 200
      </param-value>
    </init-param>
  </servlet>
```

Szervlet paraméterek kinyerése

```
List courses = new LinkedList();
StringTokenizer stk1=new StringTokenizer
  (this.getInitParameter("cnames"),",");
StringTokenizer stk2=new StringTokenizer
  (this.getInitParameter("cprices"),",");
while( stk1.hasMoreTokens()){
  String name = stk1.nextToken();
  double price = Double.parseDouble(
                          stk2.nextToken());
  ...
}
```

HttpRequest

Mit tartalmaz?

- Paraméterek elérését végző metódusokat: `request.getParameter("price")`
- Adatfolyam lekérése (ez kézileg is feldolgozható): `request.getReader()`, `request.getInputStream()`
- Ügyfélinformációk: `request.getHeaderNames()`
- Objektum típusú attribútumok: `request.setAttribute("nameList",nameList)`
- ...

Ügyfélinformációkat megjelenítő szervlet

```
response.setContentType("text/plain");
PrintWriter out = response.getWriter();
out.println("Request's headers");
out.println();
Enumeration names=request.getHeaderNames();
while( names.hasMoreElements()){
  String name =(String)names.nextElement();
  Enumeration values=
                    request.getHeaders(name);
  while( values.hasMoreElements()){
    String value =
      (String) values.nextElement();
    out.println(name+": "+value);
  }
}
```

Ügyfélinformációk

```
host: localhost:9545
user-agent: Mozilla/5.0 (X11; U; Linux i686;
Gecko/20071213 Fedora/2.0.0.10-3.fc8 Firefox.
accept: text/xml,application/xml,application.
text/html;q=0.9,text/plain;q=0.8,image/png,..
accept-language: en-us,en;q=0.5
accept-encoding: gzip,deflate
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.
keep-alive: 300
connection: keep-alive
referer:
  http://localhost:9545/fejleclekerdezo/
cookie:
  JSESSIONID=bab386c1f5e42bd3ae400732df66
```

ServletContext – tulajdonságok

Tulajdonságok

- Minden működésben levő webalkalmazást egy ServletContext objektum reprezentál
- Egy webalkalmazáshoz tartozó összes webkomponensben elérhető
- lekérhető a `szervlet.getServletContext()` metódussal

ServletContext – attribútumok

Attribútumok

```
getAttribute(String name): Object
setAttribute(String name,Object value):void
getAttributeNames(): Enumeration
```

ServletContext – naplózás szervletben

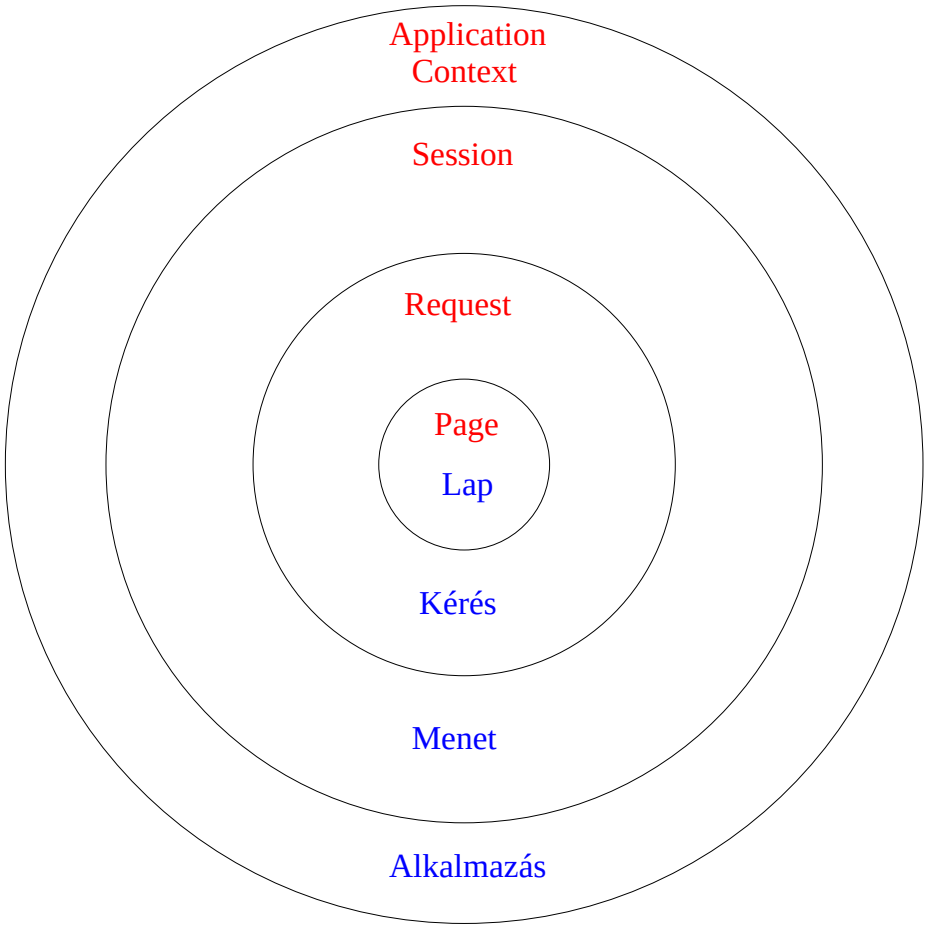
Szintaxis

```
log(String message): void
```

Kliens IP címének naplózása

```
ServletContext ctx =
  this.getServletContext();
ctx.log("Ezt ird be a naplofajlba:"+
  request.getRemoteAddress());
```

Hatókörök (Scope)



Hatókörök (szervletek)

- kérés: `HttpServletRequest`
- menet: `HttpSession`
- alkalmazás: `ServletContext`

Kérés átirányítása

Metódus:

```
public void sendRedirect(String url)
                  throws java.io.IOException;
```

Példák

```
response.sendRedirect(
     "/webalkalmazas/index.jsp");
response.sendRedirect(
     "http://www.sapientia.ro");
```

forward és sendRedirect

- forward
  - szerveroldalon történik - a kliensoldal nem érzékeli
  - csak ugyanazon webalkalmazáson belüli komponensre
  - gyors
- sendRedirect
  - kliensoldalon történik - a böngészőt utasítja új kérésre
  - rugalmas: bármilyen URL-re történhet az átirányítás
  - lassú

Konkurencia szervletekben

- a szervlet egy példányban van jelen a webkonténerben – az egyidejű kéréseket ugyanaz a szervlet szolgálja ki
- minden kérës kiszolgálása külön szálon történik.

```
public class MyServlet
        extends HttpServlet{
  //Konkurenciat kezelni kell
  private DataType data;
  ...
  protected void doPost(...)
     //Nincs konkurencia problema
     DataType var1, var2;
    ...
  }
}
```

Összefoglalás

- Vezérlési komponens tervezése
- HTML űrlap készítése
- Űrlap adatainak küldése
- Vezérlési komponensek megvalósítása
- Szervletek közötti kommunikáció
- Ügyfélinformációk
- ServletContext