Szervletek

ANTAL Margit

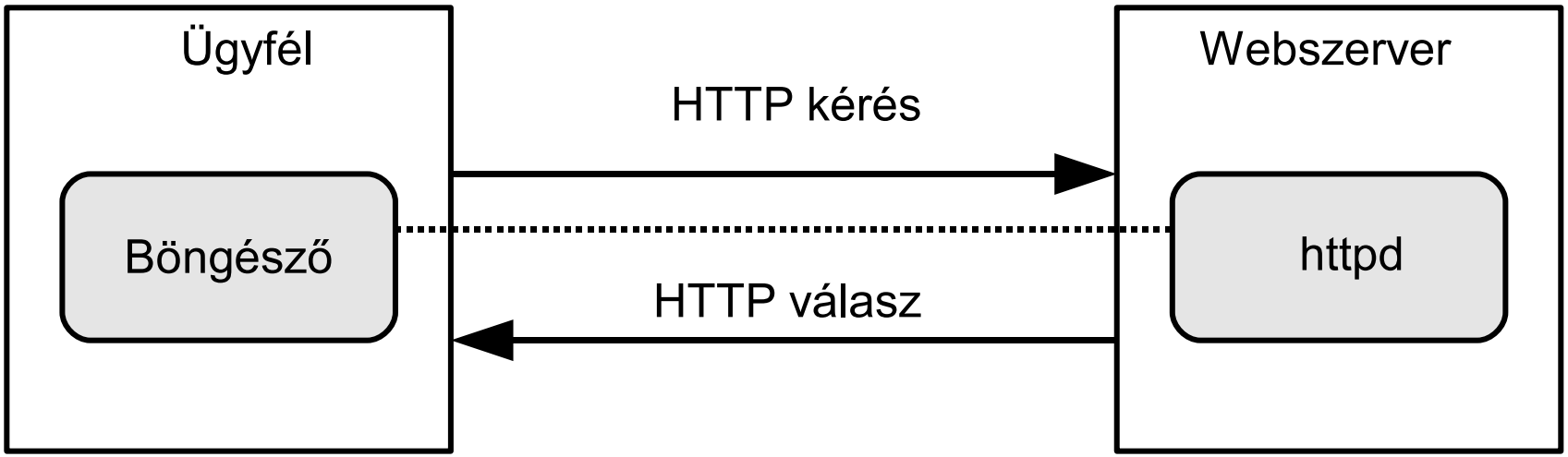
Sapientia - EMTE, Pannon Forrás ,,Egységes erdélyi felnőttképzés a Kárpát-medencei hálózatban”

2010

Szervletek

- HTTP kérés-válasz modell
- A Servlet API
- Szervletek közötti kommunikáció
- Megjelenítési komponens
- Vezérlési komponens

HTTP kliens-szerver architektúra



Fontosabb HTTP metódusok

HTTP metódus	Magyarázat
OPTIONS	Lekérdezi a szerver kommunikációs opcióit.
GET	A megadott erőforrás letöltését kezdeményezi.
HEAD	Ugyanaz mint a GET, csak az üzenet törzsét kihagyja
POST	Feldolgozandó adatot küld a szervernek.
PUT	Feltölti a megadott erőforrást.
DELETE	Törli a megadott erőforrást.

HTTP GET metódus

- Minden egyes hiperlinkre való kattintáskor ez történik.
- Minden egyes média típusú fájlra is hasonló kérést küld a böngésző
- HTTP kérés:

  `<HTTP metodus> <URL> <HTTP verzio>`
- HTTP kérésre példa:

  `GET /list-courses.html HTTP/1.0`

HTTP kérés fejléc elemek

Fejléc	Magyarázat
Accept	Milyen MIME típusokat fogad az ügyfél
Host	A kért erőforrás adatai: szerver és portszám
Referer	Az ügyfél címe
User-agent	Információk az ügyfélről

HTTP válasz

A válasz szerkezete:

1. Státusz sor
2. Fejléc sorok
3. Üres sor
4. Üzenet blokkja

```
HTTP/1.0 200 OK
Content-Type: text/html
Date: Tue, 10 Apr. 2001 10:00:03 GMT
Server: Apache Tomcat/4.0-bl

<HTML>
...
</HTML>
```

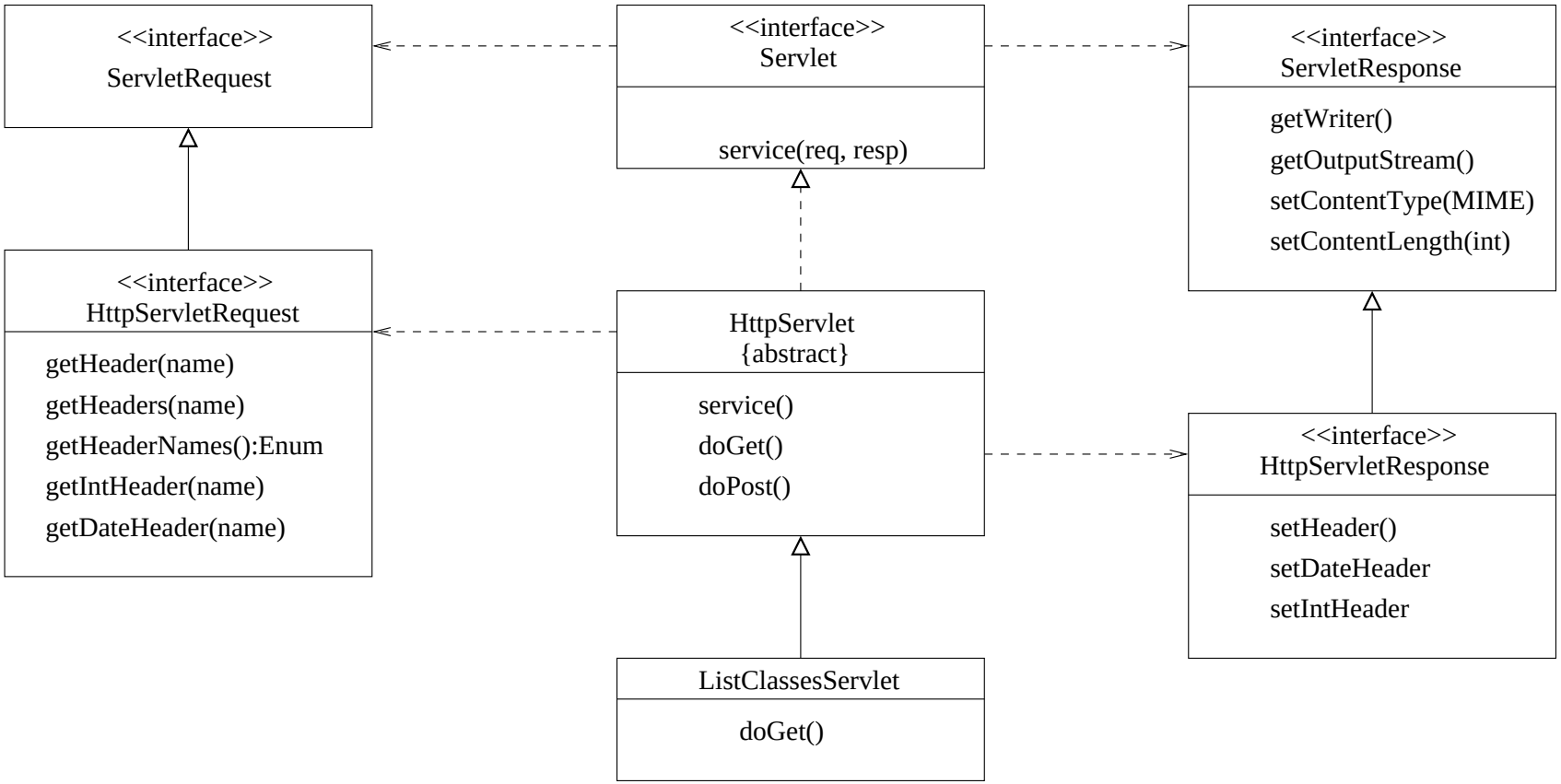
HTTP válasz fejléc

Fejléc	Magyarázat
Content-Type	MIME típus
Content-Length	a hasznos válasz hossza
Server	A választ küldő szerver neve
Cache-Control	A böngészőnek küldött direktíva arról, hogy a válasz betehető-e a gyorsító tárba (cache)

Szervletek

- A szervlet egy Java osztály, melynek feladata egy kérés kiszolgálása
- A szervlet implementálja a Servlet interfészt
- A szervlet életciklusát az a webkonténer kezeli, amelybe telepítve volt az illető szerver

HttpServlet API



HttpServlet metódusok

HTTP metódus	HttpServlet metódus
OPTIONS	doOptions()
GET	doGet()
HEAD	doHead()
POST	doPost()
PUT	doPut()
DELETE	doDelete()
TRACE	doTrace()
CONNECT	doConnect()

A service metódus

A service metódus

1. Két paramétere van: egy kérés és egy válasz objektum, amelyeket a webkonténer hoz létre
2. A kérés `HttpServletRequest` típusú
3. A válasz `HttpServletResponse` típusú

Egyszerű szervlet készítése

```
public class ListCourses extends HttpServlet {
   public void processRequest(HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException {
      //Kod
   }
   public void doGet(HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException {
         processRequest(request, response);
    }
    public void doPost(HttpServletRequest request,
      HttpServletResponse response)
      throws ServletException, IOException {
         processRequest(request, response);
    }
}
```

Szervlet konfigurálása: `web.xml`

- Szervlet definíció

```
<servlet>
  <servlet-name>ListCourses</servlet-name>
  <servlet-class>view.ListCourses</servlet-class>
</servlet>
```

- Szervlet leképzés (mapping)

```
<servlet-mapping>
  <servlet-name>ListCourses</servlet-name>
  <url-pattern>/list-courses.view</url-pattern>
</servlet-mapping>
```

Szervlet leképzés

```
<servlet-mapping>
  <servlet-name>ListCourses</servlet-name>
  <url-pattern>*.view</url-pattern>
</servlet-mapping>
```

A szervlet életciklusa

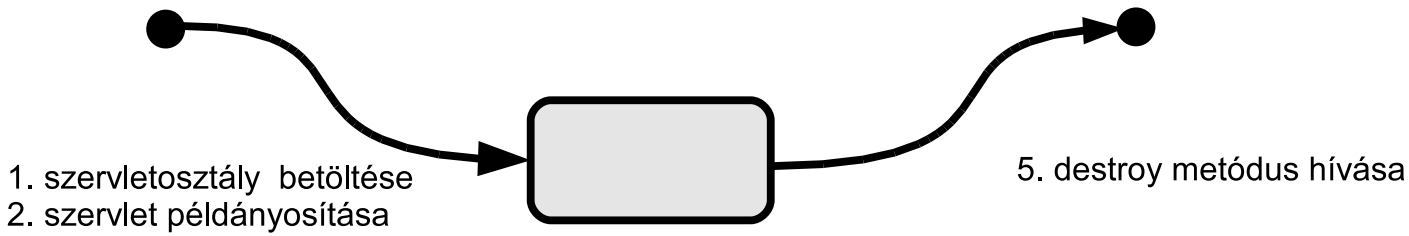
Ha egy kérés érkezik a szervlethez:

- Ha a szervletnek még nem létezik példánya, akkor a web konténer: betölti az osztályt, példányosítja majd inicializálja (init metódus hívása)
- Meghívja a service metódusát
- Ha a konténernek el kell távolítania a szervletet, meghívja a destroy metódusát

FONTOS!!!

Az init és a destroy csak egyszer hívódik meg
A service annyiszor hívódik ahány kérés érkezik az adott szervlethez

A szervlet életciklusa



(4. service metódus hívása)*

A kérés objektum - HttpServletRequest

Fejléc metódusok

- `getHeaderNames()`: az összes név lekérése
- `getHeader()`: valamely névhez rendelt érték lekérése
- `getIntHeader()`: a névnek megfelelő érték átalakítása egész típusúvá
- `getDateHeader()`: a névnek megfelelő érték átalakítása dátum típusúvá

HttpServletRequest példa

```
boolean xhtml= false;
String user=
       request.getHeader("User-Agent");
if( user.startsWith("Mozilla/5.0"))
  xhtml = true;
```

Kérés fejlécek megjelenítése

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)...{
  ...
  PrintWriter out = response.getWriter();
  ...
  Enumeration names = request.getHeaderNames();
  while( names.hasMoreElements()){
    String name =(String)names.nextElement();
    Enumeration values =request.getHeaders(name);
    if (values == null) return;
    while( values.hasMoreElements()){
      String value = (String) values.nextElement();
      out.println(name+": "+value);
    }
  }
  out.close();
}
```

A válasz objektum - HttpServletResponse

- `setHeader()`
- `setIntHeader()`
- `setDateHeader()`
- `getOuputStream()`: bináris tartalom írása
- `getWriter()`: szöveges tartalom írása, pl. HTML

HttpServletResponse példa

```
response.setContentType("text/html");
response.setHeader
      ("Cache-Control","no-cache");
```

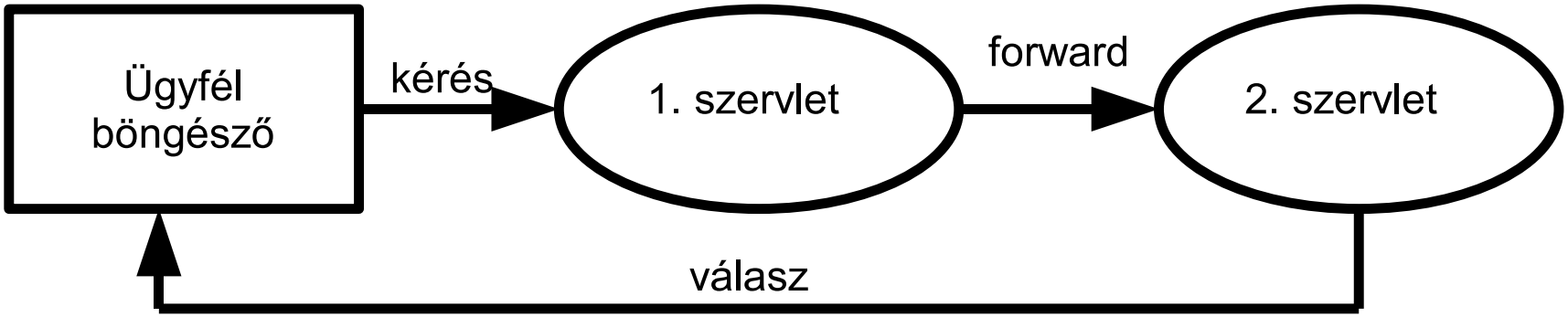
Átirányítás

```
public void sendRedirect(java.lang.String location)
                  throws java.io.IOException
```

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")`

A kérés továbbítása



Vezérlés átadása

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

A `ServletContext` interfész

Biztosítja:

1. attribútumok tárolását az alkalmazás hatókörében (≈globális változók)
2. szervlet paraméterek feldolgozása (≈parancssor argumentumai)
3. a kérés továbbítása
4. hozzáférés erőforrásokhoz
5. naplózás

1. Attribútumok tárolása

```
ServletContext sc =
            getServletConfig().getServletContext();
String name ="Your Name";
sc.setAttribute("name", name);
```

2. Szervlet paraméterek

```
<servlet>
  <servlet-name>AdminViewServlet</servlet-name>
  <servlet-class>view.AdminViewServlet.class</servlet-class
  <init-param>
    <param-name>email</param-name>
    <param-value>admin@distedu.org</param-value>
  </init-param>
</servlet>
```

Adott nevű paraméter

ServletContext sc = getServletConfig().getServletContext(); String email = sc.getInitParameter("email");

Az összes paraméter

Enumeration parameters = sc.getInitParameterNames();

3. A kérés továbbítása

Figyelem

url: csak abszolút URL lehet!!

```
getRequestDispatcher(String url)
```

4. Hozzáférés erőforrásokhoz

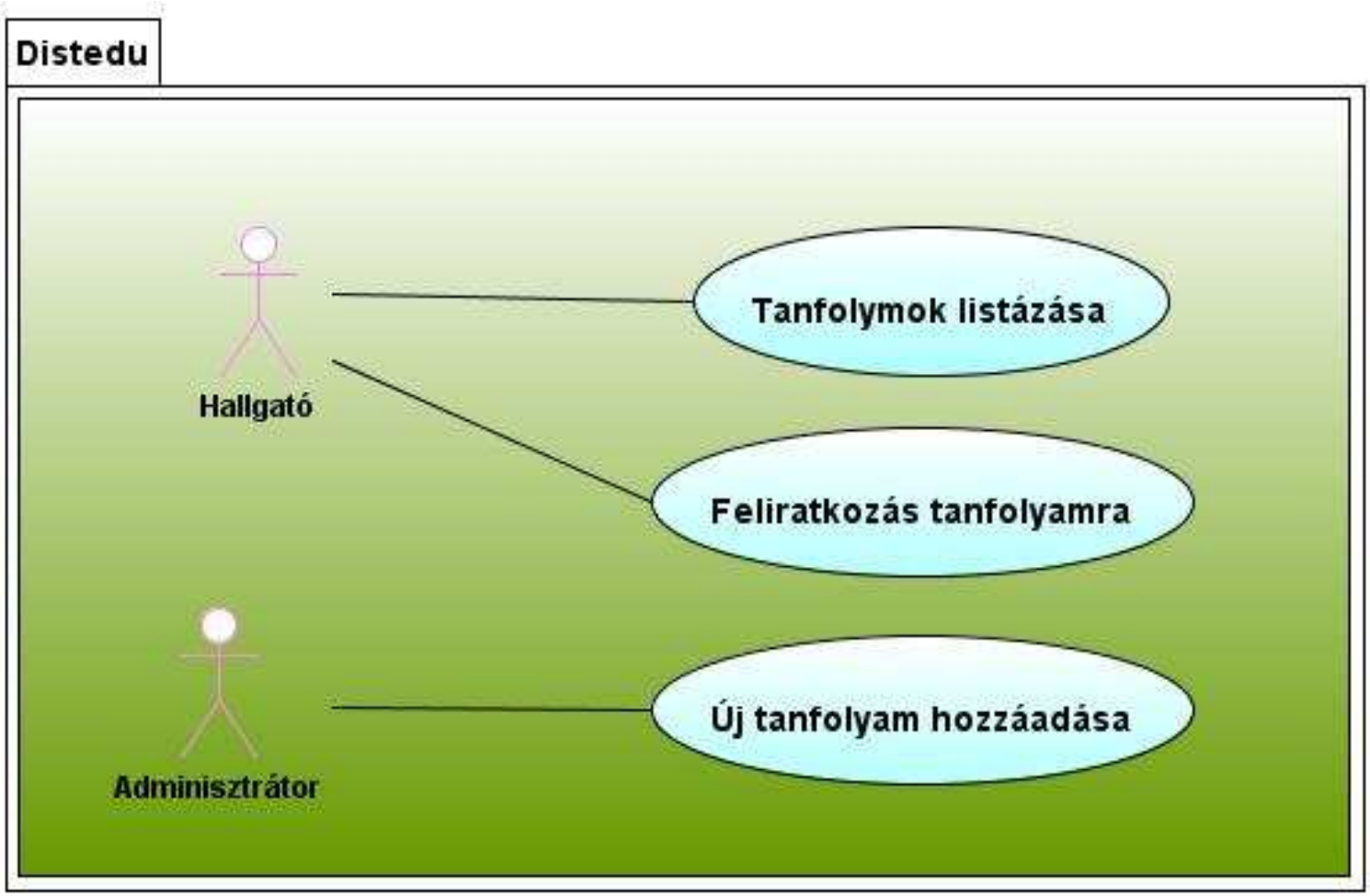
- `String getRealPath( String path )`: egy virtuális névnek megfelelő fizikai nevet ad meg. Fájl erőforrás esetében ez a fájlnév lesz a fájlrendszerbeli abszolút elérési útvonallal együtt.
- `Set getResourcePaths( String path )`: a paraméterként megadott útvonalon található erőforrások listáját adja vissza. Az útvonalnak '/' karakterrel kell kezdődnie.
- `InputStream getResourceAsStream( String path )`: a paraméterként megadott erőforráshoz megnyit egy bemeneti csatornát.
- `java.net.URL getResource( String path )`: a paraméterként megadott erőforrás URL-jét adja vissza

Metódus

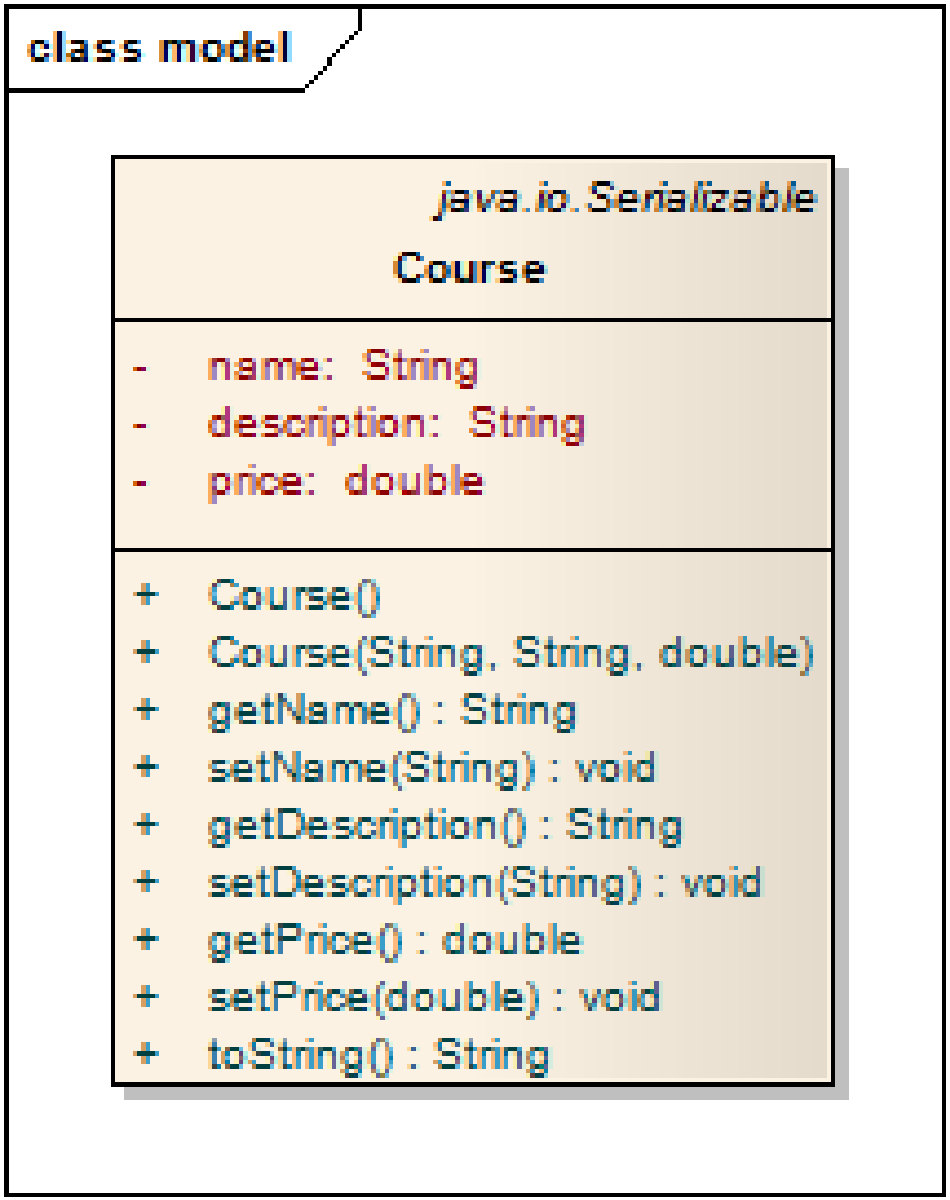
log(String message)

```
ServletContext sc =
  getServletConfig().getServletContext();
String resource ="/error.jsp";
java.net.URL url = sc.getResource( resource );
sc.log("Resource: "+resource+"\t"+" URL: "+url);
```

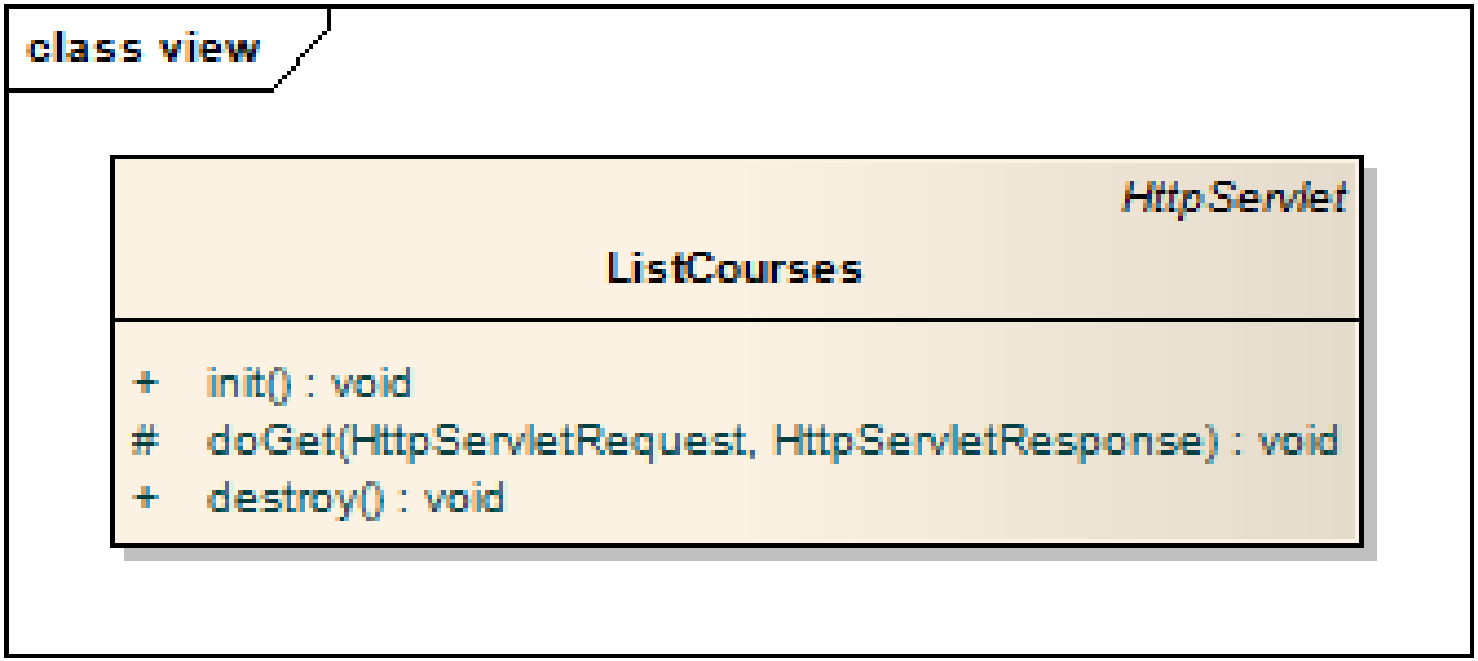
Távoktatás alkalmazás



Modell komponens tervezése



Megjelenítési komponens tervezése



A szervlet konfigurálása

```
<servlet>
  <servlet-name>ListCourses</servlet-name>
  <servlet-class>view.ListCourses</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>ListCourses</servlet-name>
  <url-pattern>/list_courses.view</url-pattern>
</servlet-mapping>
```

Listázó szervlet

```
/WEB-INF/tanfolyamok.txt

public class ListCourses extends HttpServlet {

    public void init(){ ... }

    protected void doGet(){ ... }

    public void destroy(){ ... }

}
```

Az init metódus

```
public void init(){
    List<Course> courselist = new ArrayList();
    String resource= "/WEB-INF/tanfolyamok.txt";
    InputStream is=
          this.getServletContext().
          getResourceAsStream(resource);
    BufferedReader br = new BufferedReader(
                new InputStreamReader(is));
    while( true ){
      //allomany feldolgozasa
    }
    this.getServletContext().
      setAttribute("coursecounter", courselist.size());
    this.getServletContext().
      setAttribute("courselist", courselist);
    ...
 }
```

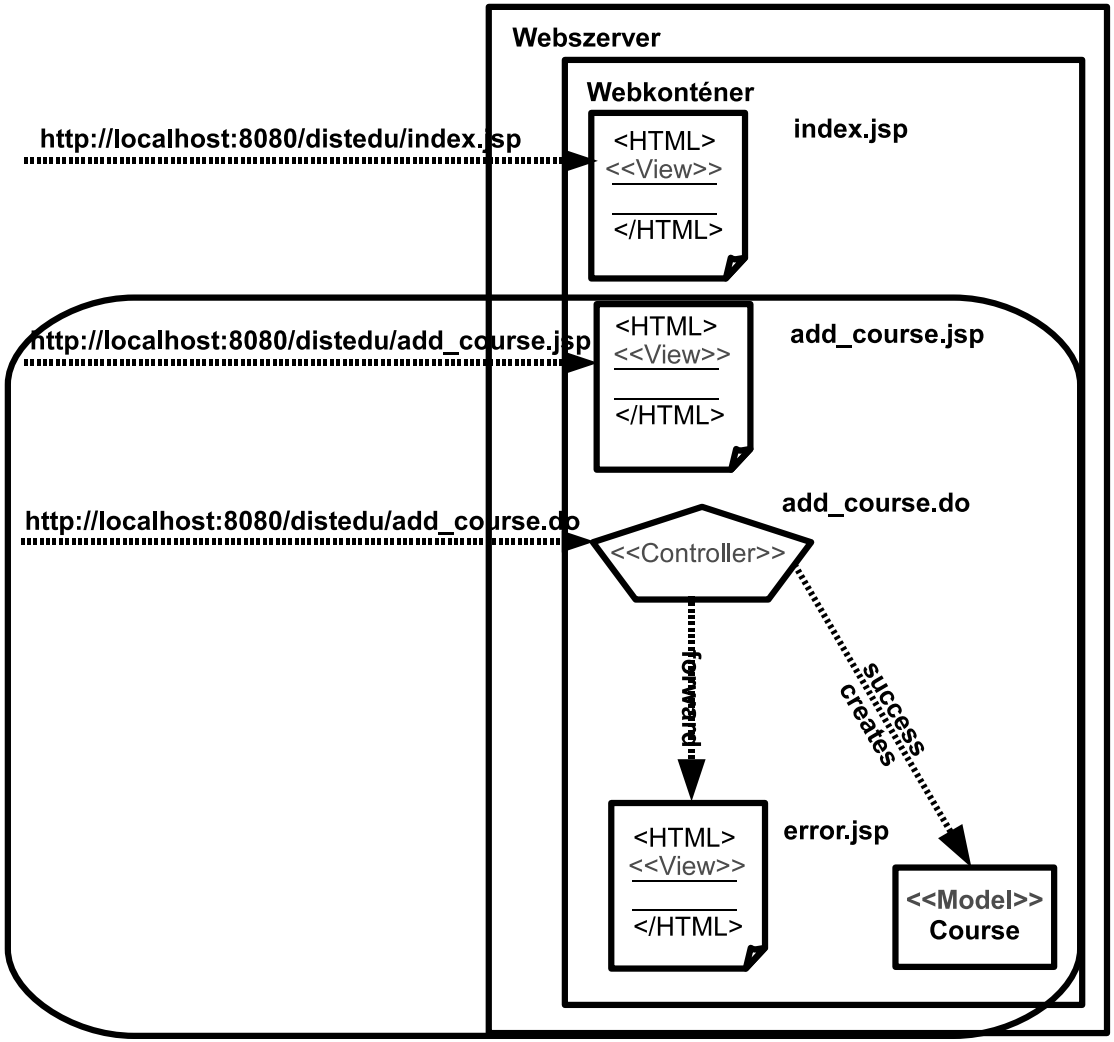
A doGet metódus

```
protected void doGet(...){
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    ...
    out.println("<h1> Course list </h1>");
    out.println("<ul>");
    List<Course> courselist =
     (List<Course>)this.getServletContext().
      getAttribute("courselist");
    Iterator<Course> it = courselist.iterator();
    while( it.hasNext()){
      out.println( "<li>"+it.next()+"</li>");
    }
    out.println("</ul>");
    ...
}
```

A destroy metódus

```
public void destroy(){
    //ha szukseges a mentes
    String resource= "/WEB-INF/tanfolyamok.txt";
    String path = this.getServletContext().
            getRealPath(resource);
    try{
      PrintWriter pw=new PrintWriter(new FileWriter(path));
      Iterator<Course> it = courselist.iterator();
      while( it.hasNext() )
        pw.println( it.next());
      pw.close();
    }
    catch(Exception e){
      e.printStackTrace();
    }
}
```

Vezérlési komponens tervezése



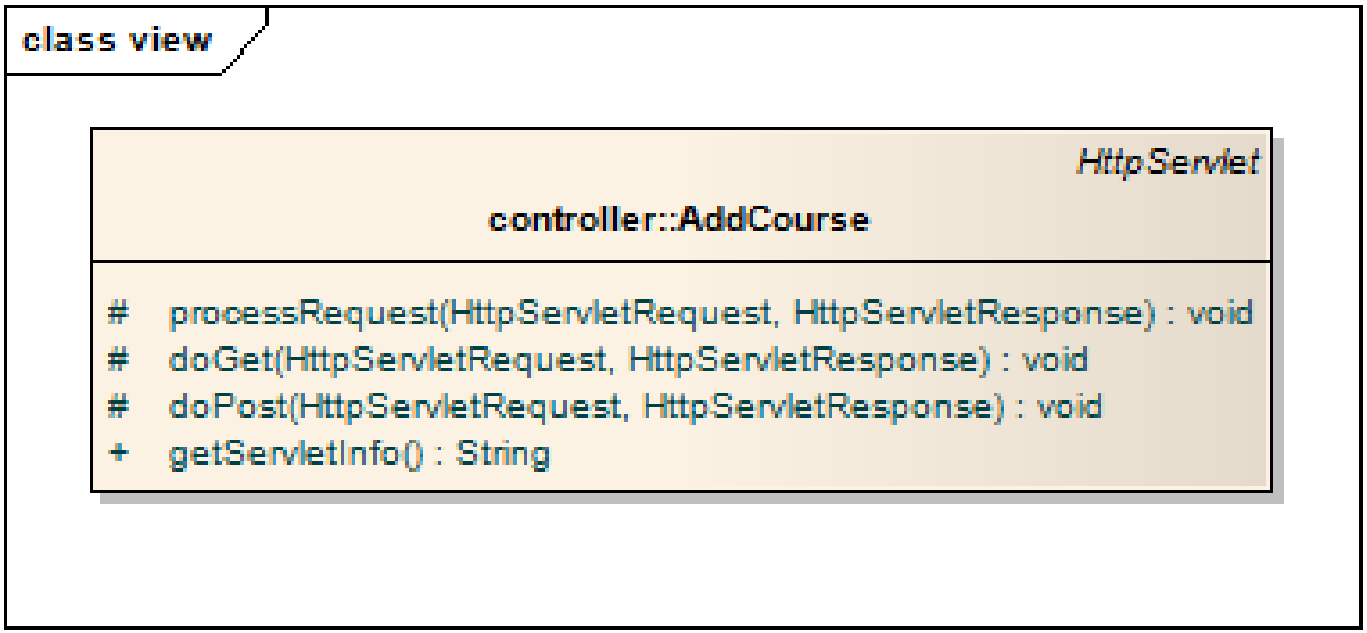
Új tanfolyam bevitele

- Űrlap (megjelenítés): `add_course.jsp`
- Űrlap adatainak feldolgozása (vezérlés): `add_course.do`

HTTP GET é POST összehasonlítása

- 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)

Vezérlési komponens tervezése



Űrlap adatok kinyerése

```
String name = request.getParameter("name");
String description = request.getParameter("description");
String priceStr = request.getParameter("price");
double price=0;
try{
  price = Double.parseDouble( priceStr );
}
catch( NumberFormatException e){
          //Hibakezeles
}
```

Hibakezelés

```
List errorMsgs = new ArrayList();

String name = request.getParameter("name").trim();
if( name == null || name.length() == 0){
   errorMsgs.add("Please eneter the name of the course");
}

...
```

Üzleti logika

```
if( !errorMsgs.isEmpty()){
  request.setAttribute("errorMsgs", errorMsgs);
  RequestDispatcher r = request.
    getRequestDispatcher("error.jsp");
  r.forward(request, response);
}
else{
  Course course = new Course(name, description, price);
  List<Course> courselist =
  (List<Course>)this.getServletContext().
     getAttribute("courselist");
  courselist.add(course);
 }
```

Distedu alkalmazás

- Tanfolyamok listázása: ListCourses.java szervlet, URI: /list_courses.view
- Új tanfolyam felvitele
  - add_course.jsp
  - AddCourse.java (szervlet), URI: add_course.do
  - error.jsp