Java programozási nyelv 2012-2013/ősz
5. óra

Java Servletek alapjai

XHTML, HTTP, Tomcat
Java szervletek

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

Témakörök

Extensible Hypertext Markup Language

Hypertext Transfer Protocol

Apache Tomcat szerver alapjai

Java szervletek felépítése

Extensible Hypertext Markup Language

- Az XHTML egy XML alapú dokumentumleíró nyelv
- A nyelv kialakulása szempontjából lényegesebb események:
  - 1993 HTML (IETF)
  - 1995 HTML 2.0 (W3C)
  - 1999 HTML 4.01 (W3C)
  - 2000 XHTML 1.0 (W3C)
  - Jelenleg elérhető verzió: XHTML 1.1
- A HTML az SGML egy alkalmazása, ami viszont nagy rugalmasságot engedett meg a fejlesztőknek. Az így készült weboldalak megjelenítése túl nagy terhet jelent az webet használni szándékozó új eszközöknek (pl. mobiltelefonok)
- Ezért a W3C egy új, XML alapú nyelvet készített el XHTML néven, ami szigorúbb feltételeket szab az oldalak készítői számára, ezzel azonban egyszerűsíti a megjelenítő eszközök feladát

Streamszervezési módok

- Az XHTML Specifikációját tartalmazó DTD megtalálható a W3C oldalán, ennek megfelelően egy oldal általános szerkezete az alábbi:

```
<?xml version="1.0" encoding="ISO-8859-2"?>
<!DOCTYPE html PUBLIC
                "-//W3C//DTD XHTML 1.0 Strict//EN"
                "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <title>Kötelező fejléc</title>
    </head>
    <body>
        ...
    </body>
</html>
```

HTML és XHTML közötti különbségek

- Az alapvetőbb HTML elemeknek megtalálhatjuk az azonos nevű XHTML megfelelőjét
- Nem lehetnek átfedések az egymásba ágyazott elemek között
  Pl. ABCDEFGH
- Minden elemet le kell zárni
  Pl.
 helyett
- Kis- és nagybetű különbsége
  Pl. <BODY> helyett <body>
- Minden tulajdonságértéket idézőjelbe kell írni
  Pl. <table rows=3> helyett <table rows="3">
- Speciális adatok tárolása a CDATA kulcsszóval oldható meg
  Pl. JavaScript források az oldalban

XHTML formázás – betűtípus beállítása

- Közvetlen formázó parancsok:
  - – betűtípus megadása, attribútumai:
    color – szín
    size – méret
    face – betűkészlet
    Pl. X
  - <i> – dőlt betű
  - <b> – félkövér betű
  - <u> – aláhúzott
  - ...
- Formázás tartalom alapján:
  - <code> – programszöveg
  - <cite> – idézet
  - ...

XHTML formázás – bekezdések

- Bekezdések formázása
  - – bekezdés, lehetséges attribútumai:
    align = ”center” – középreigazítás
    align = ”left” – balra igazítás
    align = ”right” – jobbra igazítás
    Pl. Jobbra igazított bekezdés
  -
 – sortörés
  - <hr/> – vízszintes elválasztó vonal
  - ...
- Címsorok
  - <h1> – 1. címsor
  - <h2> – 2. címsor
  - ...
  - <h6> – 6. címsor

XHTML hiperhivatkozások

- Hiperhivatkozás általános formája
  - Megjelenítendő szöveg
- Abszolút hivatkozás
  - Pl. BMF honlap
  - Pl. Gézának levél
  - Pl. Belépés
- Relatív hivatkozás
  - Pl. Ugrás a másik oldalra
- Oldalon belüli ugrás
  - Pl. Ugrás a bevezetőre
  - Pl. Ugrás a bevezetőre
- Könyvjelző definiálása
  - Pl. <h1 id=”bevezetes”>Bevezetés</h1>
  - Természetesen más elemek is elnevezhetők

XHTML képek beillesztése

- Használható formátumok: GIF, JPEG
- Az <img> elem tulajdonságai
  - src – a beszúrandó képet tartalmazó URL
  - alt – alternatív szöveg, ha a böngésző nem jeleníti meg a képet
  - align – igazítás
  - height, width – magasság, szélesség

  Pl.
- Képen belül hiperhivatkozás mezők kijelölése

  Pl.

```
<map name = "map2">
  <area coords="0,0,50,50" href="apu.html" />
  <area coords="51,0,100,50" href="anyu.html" />
</map>
```

XHTML táblázatok

- <table> – táblázat létrehozása, attribútumai:
  - bgcolor – háttérszín
  - border – keret tulajdonságai
  - cellspacing – cella és szöveg közti távolság
  - width – táblázat szélessége
- <tr> – a táblázat egy sora
  - bgcolor - háttérszín
  - align – igazítás
- <td> – a táblázat egy cellája
  - bgcolor - háttérszín
  - align – igazítás
  - width, height – cella méretei
  - nowrap – szöveg tördelés tiltása
- <th> – fejléc (használata azonos a <td>-el)

XHTML táblázat példa

Példa táblázatra:

```
<table border="1" cellpadding="5" width="50%">
  <tr>
    <th>Név</th><th>Munkakör</th><th>Fizetés</th>
  </tr>
  <tr>
    <td>Kovács József</td><td>Tanár</td><td>50000</td>
  </tr>
  <tr>
    <td>Nagy Piroska</td><td>Takarító</td><td bgcolor="#FF0000">50000</td>
  </tr>
  <tr>
    <td>Kovács Pál</td><td>Tanár</td><td align="center">50000</td>
  </tr>
</table>
```

Név	Munkakör	Fizetés
Kovács József	Tanár	50000
Nagy Piroska	Takarító	50000
Kovács Pál	Tanár	50000

Témakörök

Extensible Hypertext Markup Language

Hypertext Transfer Protocol

Apache Tomcat szerver alapjai

Java szervletek felépítése

Hypertext Transfer Protocol

- Alapfogalmak
  - IP, port, socket
  - Protokoll
  - URL
- A HTTP egy TCP alapú alkalmazásszintű protokoll. A WWW központi protokollja, hypertext dokumentumok letöltésére fejlesztették ki. Ezzel a protokollal kommunikálnak egymással a HTTP szerverek és HTTP kliensek (másnéven: webszerver, böngésző).
- Protokol verziói:
  - 1996 - HTTP 1.0 [RFC1945]
  - 1999 - HTTP 1.1 [RFC2616]
  - A W3C nem fejleszti tovább.
- A kliens és a szerver közti kommunikáció típusos

  Az átvitt információk el vannak látva egy fejléccel, amely tartalmazza az adatok értelmezési módját. Ez a MIME szabványok segítségével történik.

HTTP kérés/válasz részei

- Kliens által küldött kérés adatai
  - Használt HTTP protokoll verziószáma
  - Végrehajtandó parancs
  - Az erőforrás azonosítója, amin az előbbi parancsot végre kell hajtani
  - Fejlécmezők, amelyek tartalmazzák a kérés további paramétereit (nyelv, kódolás stb.)
  - Törzsadat (ha van)

- A szerver válaszában található adatok
  - Használt HTTP protokoll verziószáma
  - Válaszkód
  - Fejlécmezők, amelyek tartalmazzák a válasz további paramétereit (kódolás, lejárati idő stb.)
  - Törzsadat (ha van)

HTTP kérés részletei

- Kliens által kiadható parancsok
  - GET – a kliens a megnevezett erőforrást szeretné letölteni
  - HEAD – a kliens a megnevezett erőforrás fejlécét kéri
  - POST – A kliens módosítani szeretné a megnevezett erőforrást
  - DELETE – A kliens törölni szeretné a megnevezett erőforrást

  ...

- Kliens által elküldhető fejlécmezők
  - Accept

  Azon médiatípusok listája, amelyeket a kliens fogadni képes
  - Accept-Language

  Azon nyelvek listája, amelyeket a kliens fogadni képes
  - User-Agent

  A kliens oldalon használt alkalmazás neve, verziószáma
  - Authorization

  Szerverrel szembeni jogosultság igazolások

  ...

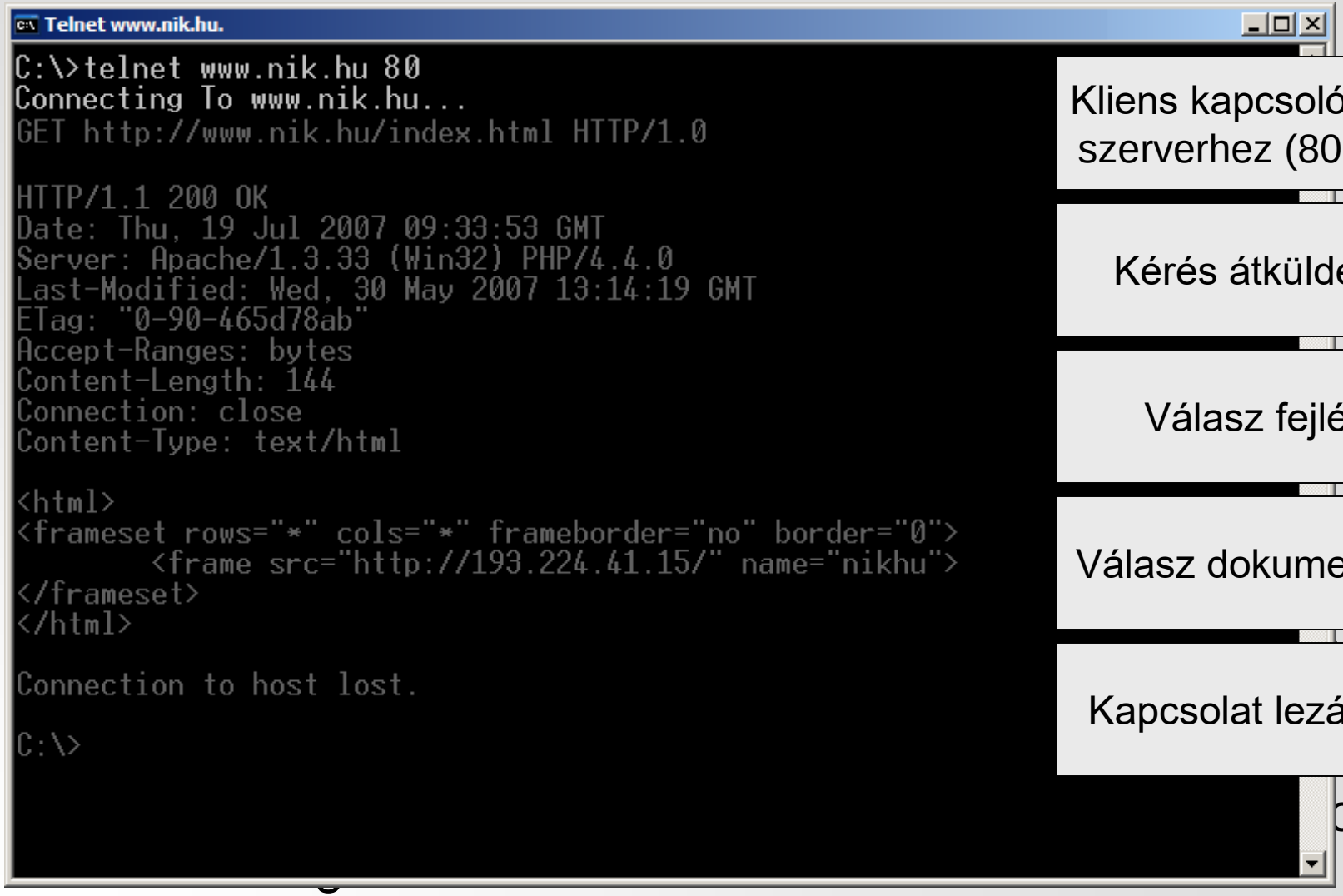
HTTP válasz részei

- A szerver által visszaküldhető státuszkódok
  - 1×× – nem használt
  - 2×× – a kérést sikeresen végrehajtotta
    Pl. 200 – a kérés kiszolgálva
    Pl. 204 – a kérés kiszolgálva, de nincs visszaadott tartalom
  - 3×× – a hivatkozott objektumot más helyre telepítették
    Pl. 301 – tartósan áthelyezve
    Pl. 302 – ideiglenesen áthelyezve
  - 4×× – a szerver nem tudta végrehajtani a kérést
    Pl. 400 – hibás kérés
    Pl. 401 – authorizációs hiba
    Pl. 404 – erőforrás nem található
  - 5×× – webszerver hiba
    Pl. 501 – belső kiszolgálóhiba
    Pl. 501 – a kiszolgáló nem támogatja a megadott parancsot
    Pl. 503 – a szolgáltatás ideiglenesen nem érhető el

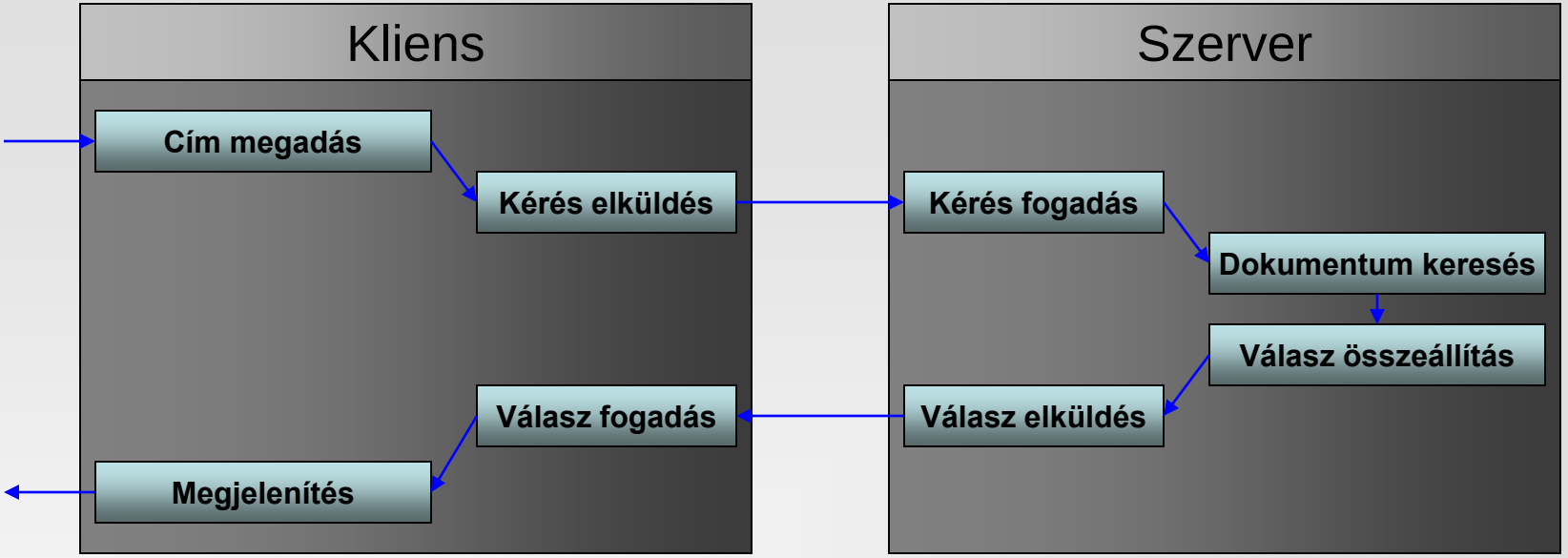
HTTP válasz részei

- Szervertől érkező csomagok fejlécmezői
  - Content-Type
    A HTTP törzsadat MIME típusa
  - Content-Encoding
    A válasz során használt kódolási mód megnevezése
  - Content-Length
    A HTTP törzsadat hossza
  - Date
    Az üzenet létrehozásának ideje
  - Expires
    Az üzenet elévülési időpontja
  - Last-Modified
    Az utolsó módosítás időpontja
  - Location
    Ha az erőforrást áttelepítették, akkor annak a helye
  - Cache-Control
    Egyéb utasítások a gyorsítótárazással kapcsolatban
    ...

Hypertext Transfer Protocol

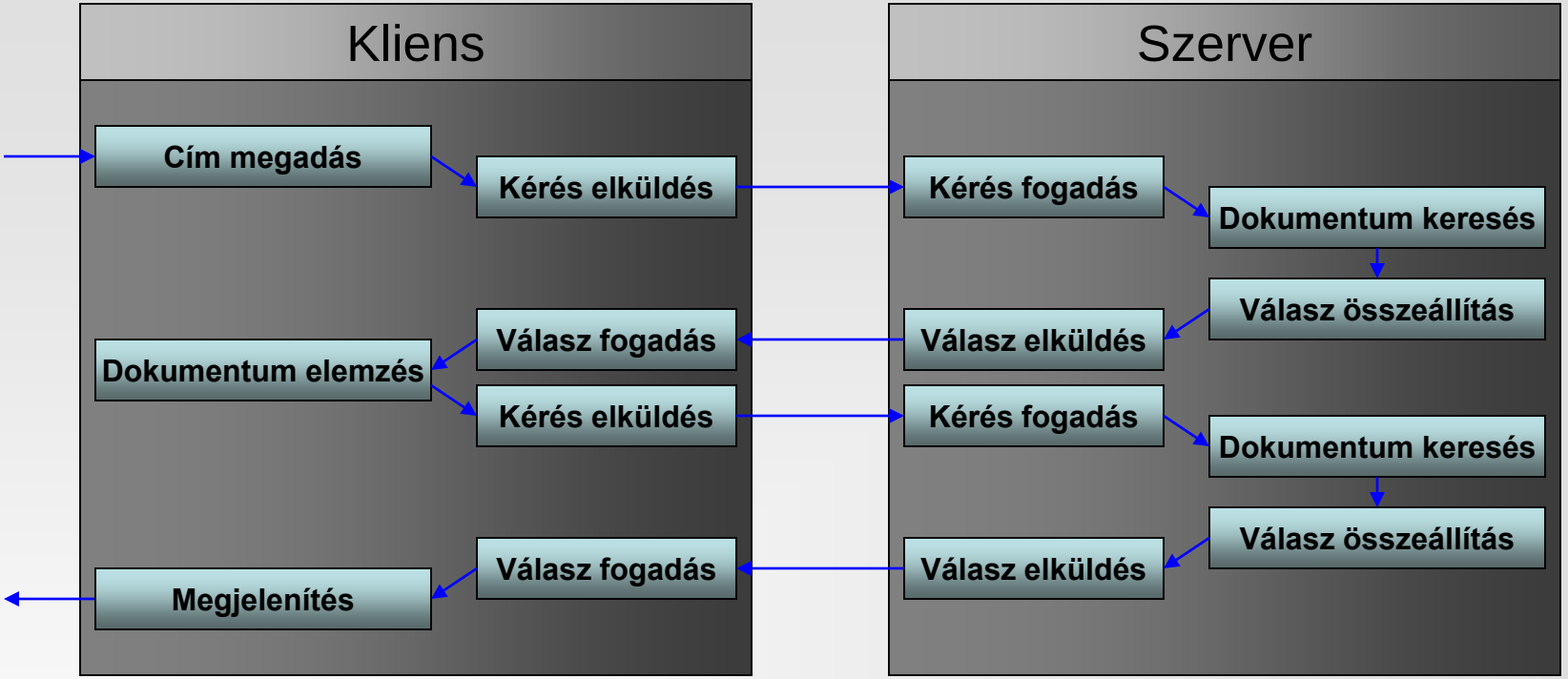


Egyszerű dokumentum letöltése



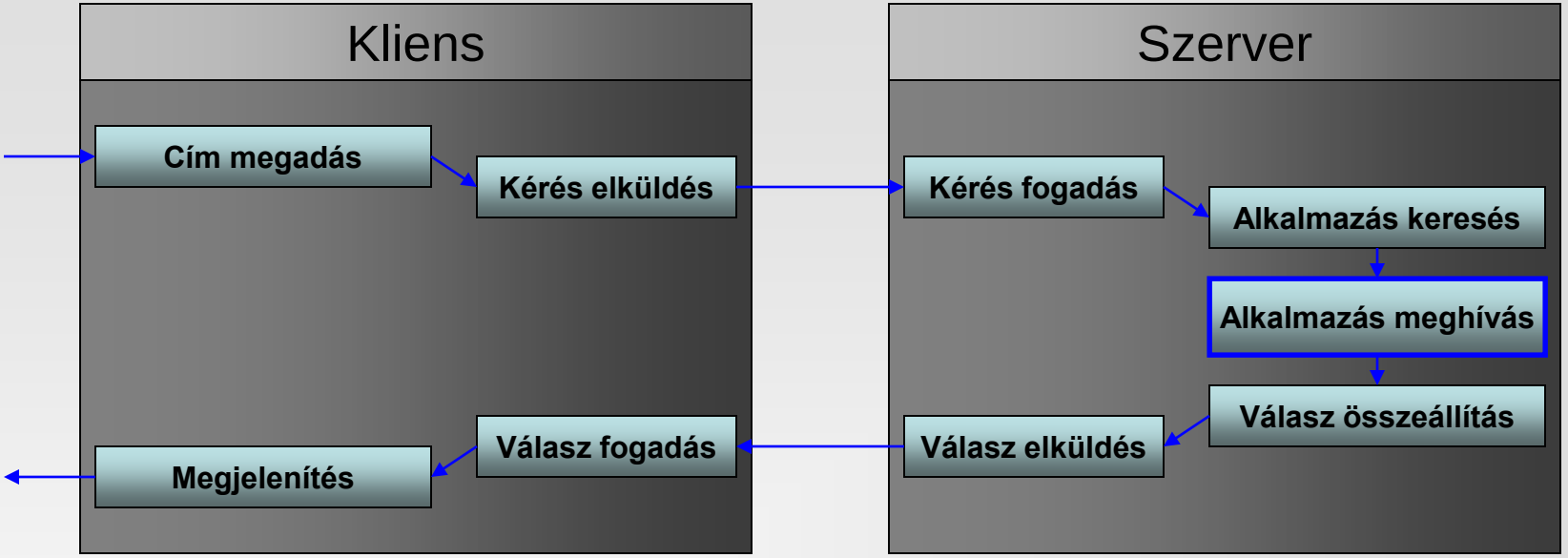
- Egy egyszerű dokumentum lekérdezése
  - A felhasználó a böngészőben megadja az elérni kívánt URL-t
  - A böngésző felveszi a kapcsolatot a címben található szerverrel és elküldi a kérést
  - A szerver a kérés alapján megkeresi az igényelt erőforrást. Előállítja a HTTP választ, majd visszaküldi a kliensnek
  - A kliens oldalon a böngésző beolvassa a dokumentumot és megjeleníti

Hivatkozásokat tartalmazó dokumentum



- Hivatkozásokat (pl. képeket) tartalmazó dokumentum
  - A dokumentum letöltése azonos az előzővel
  - Ha a dokumentum letöltése után a böngésző abban hivatkozásokat talál, akkor a megjelenítés előtt (közben) hasonló módon letölti az azokhoz tartozó erőforrásokat is
  - A szerver szempontjából a két kérés egymástól független!

Dinamikus oldalgenerálás



- Dinamikus oldalgenerálás
  - A gyakorlatban nem elég a fájlrendszer statikus állományaihoz való hozzáférés, szükség lehet a kérés paramétereitől függő oldalak azonnali létrehozására is
  - A webszerver konfigurálható úgy is, hogy bizonyos kérések esetén ne magát az erőforrást adja vissza, hanem azt indítsa el (átadva a kérés paramétereit) és annak kimenetét küldje vissza
  - A kliens számára mindez láthatatlan

Dinamikus oldalgenerálás

- CGI alkalmazások
  - Az első és legegyszerűbb megoldás erre a feladatra
  - A CGI alkalmazások egyszerű futtatható állományok, szinte tetszőleges programozási nyelven készíthetők
  - A CGI a paramétereket a szabványos bemeneten, vagy környezeti változókon keresztül kapja meg
  - A CGI a kimenetét a szabványos kimeneten keresztül adja vissza
  - A CGI alkalmazások használata számos hátránnyal jár: nagy erőforrásigény, kevés szolgáltatás
- Java szervletek
  - A Java EE megoldása erre a problémára a Java szervlet
  - Ezek olyan osztályok, amelyek egy megfelelő futtató környezetben léteznek és folyamatosan kiszolgálják a kéréseket
  - Szervletekről később még lesz szó
- Egyéb lehetőségek
  - Webszerver/futtatási környezet specifikus megvalósítások

Témakörök

Extensible Hypertext Markup Language

Hypertext Transfer Protocol

Apache Tomcat szerver alapjai

Java szervletek felépítése

Apache Tomcat

- Az Apache Tomcat egy web konténer, ami megvalósítja a szervlet és a JSP (később lesz róla bővebben szó) specifikációkat
- Fejlesztője az Apache Software Foundation (ASF)
- Rendelkezik egy beépített HTTP szerverrel, tehát a Java szervletek kezelésén túl teljesértékű webszerverként is üzemel
- A motor teljes mértékben Java alapokon nyugszik, így használható minden platformon, amin megvalósították a Java virtuális gépet
- A Tomcat motor üzemeltethető más HTTP szerver részeként is, erre főleg teljesítmény okokból lehet szükség
- Kapcsolódó oldalak
  - ASF – `http://www.apache.org`
  - Tomcat – `http://tomcat.apache.org`

Apache Tomcat indítás/leállítás

- Windows esetén telepíthető a Tomcat szolgáltatásként, ez esetben az indítás és leállítás paraméterei a többi szolgáltatáshoz hasonlóan adhatók meg
- Telepítés során az is választható, hogy egyszerű alkalmazásként működjön a szerver

  Ebben az esetben a bin alkönyvtárban található *tomcat.exe* állománnyal lehet elindítani
- Ez utóbbit végzi el a virtuális gép asztalán található „Tomcat server” ikon is
- A NetBeans telepített változata tartalmaz egy beépített Tomcat szervert, a későbbiekben ezzel dolgozunk. Ez a szerver elérhető a NetBeanstől függetlenül (lásd az ikon az asztalon), illetve a fejlesztői környezet működése során is elindíthatja és működtetheti, ha a project úgy kívánja (pl. szervletet szeretnénk nyomkövetni)

Apache Tomcat konfigurálás

- Webes felületen keresztül
  - Adminisztrációs csomag telepítése után érhető el
  - Megfelelő jogosultság esetén az adatok nem csak megtekinthetők, hanem módosíthatók is
  - Általában: `http://szerver/admin/`
- Közvetlenül a konfigurációs fájlokban
  - Ugyanezeket a paramétereket elérhetjük, illetve módosíthatjuk a fájlrendszerben található konfigurációs állományok segítségével
  - A szerver paramétereit XML fájlok tartalmazzák, ezek egyszerűen módosíthatók
  - Külön fájlban találhatók a szerver, illetve azon belül az egyes kontextusok paraméterei
  - Az egyszerűség kedvéért ezt a módszert használjuk

Apache Tomcat könyvtárszerkezet

- A `CATALINA_HOME` környezeti változó tartalmazza a Tomcat telepítési könyvtárat
- Ez alatt az alábbi alkönyvtárak találhatók
  - `bin` – szerver futtatható állományai
  - `common` – általánosan használt osztályok
  - `conf` – konfigurációs állományok
  - `docs` – dokumentáció
  - `logs` – napló állományok
  - `server` – csak a szerver számára elérhető osztályok
  - `webapps` – alapértelmezett alkalmazás könyvtár
  - `shared` – osztályok csak az alkalmazások számára
  - `work` – szerver munkakönyvtára

Apache Tomcat konfigurációs állományok

- `conf/catalina.policy`

  Alapvető biztonsági beállítások

- `conf/catalina.properties`

  Osztálykönyvtárakat érintő beállítások

- `context.xml`

  Alapértelmezett környezeti beállítások

- `conf/server.xml`

  A szerver alapvető beállításait tartalmazza

- `conf/logging.properties`

  Naplózási beállítások

- `conf/tomcat-users.xml`

  Alapvető felhasználói beállítások

- `conf/web.xml`

  Alapértelmezett (minden alkalmazásra vonatkozó) telepítési leíró állomány

server.xml néhány beállítása

- HTTP portszám beállítása

```
<Connector port="80" maxHttpHeaderSize="8192"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" redirectPort="8443" acceptCount="100"
  connectionTimeout="20000" disableUploadTimeout="true" />
```

- HTTPS portszám beállítása

```
<Connector port="8443" maxHttpHeaderSize="8192"
  maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" disableUploadTimeout="true"
  acceptCount="100" scheme="https" secure="true"
  clientAuth="false" sslProtocol="TLS" />
```

Saját kontextus készítése

- Saját webalkalmazás kontextus használata esetén az erre vonatkozó paramétereket az alábbi XML állomány tartalmazza

  `conf/[Szervíz]/[Host]/[Kontextus].xml`
- Új állomány létrehozása után a szerver ezt automatikusan használatba veszi
- Például a virtuális gépeken található `hweb.xml` fájl tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context docBase="H:\HallgatoWeb" path="/hweb"/>
```

Ez alapján a web szerver a `/hweb` kontextushoz tartozó gyökérkönyvtárat a `h:/HallgatoWeb` határozza meg

Ez alapján érthető a következő oldalakon leírt gyakorlati működés

HTTP server a gyakorlatban

- A Tomcat a globális szerver beállítások mellett lehetőséget ad kontextusok definiálására, amelyek saját beállításokat használhatnak
- A virtuális gépeken található Tomcat beállításai között szerepel, hogy a `/hweb` kontextus gyökérkönyvtáraként a `h:\HallgatoWeb` könyvtárat tekintse
- Kontextus meghatározása az URL-ben

  `http://szerver/kontextus/erőforrás`

  Például:

  `http://localhost/hweb/index.html`

  A böngészőbe beírva letölthetjük a szerverről a `h:\HallgatoWeb\index.html` álományt
- Van lehetőség a kontextus gyökérkönyvtáron túl alkönyvtárak készítésére és elérésére is

Témakörök

Extensible Hypertext Markup Language

Hypertext Transfer Protocol

Apache Tomcat szerver alapjai

Java szervletek felépítése

Java servletek

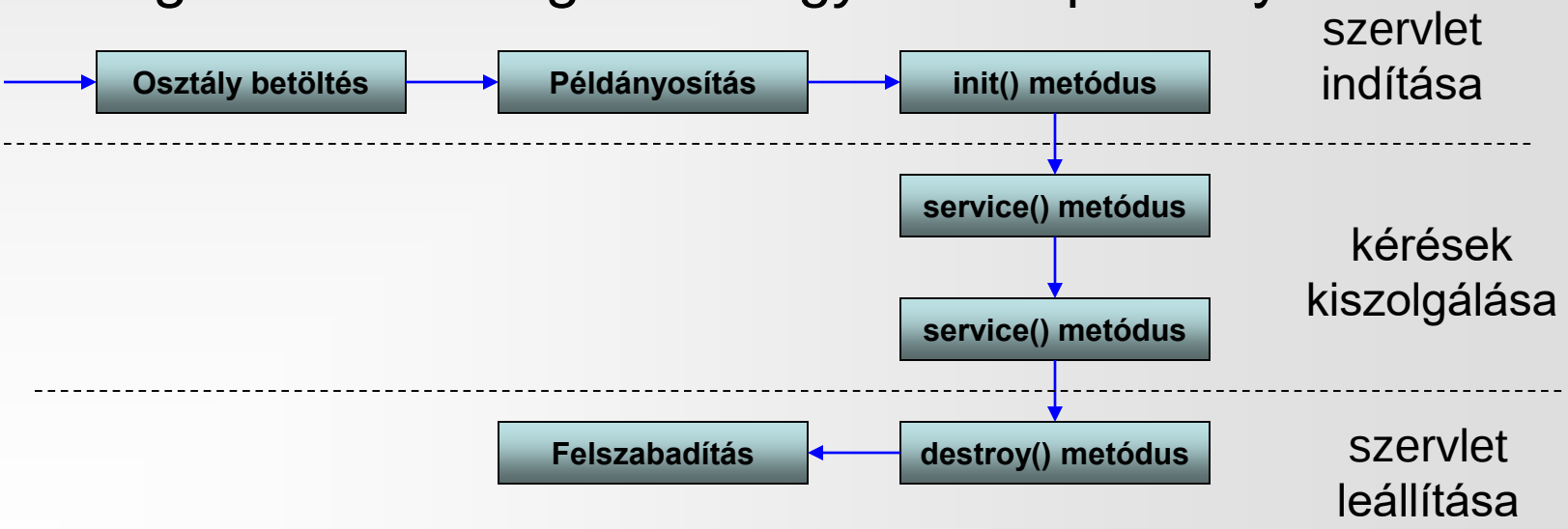
- A Java szervletek segítségével egyszerűen tudunk dinamikus weboldakat generálni a szerveren
- A szervlet egy egyszerű Java osztály, csak implementálnia kell a *javax.servlet.Servlet* interfészt
- Mivel az interfész minden metódusát kötelezően implementálni kell, általában célszerűbb a *javax.servlet.GenericServlet* osztályokból származtatni
- A Java EE szabályainak megfelelő szervleteket minden arra alkalmas konténerbe egyszerűen tudjuk telepíteni

  Szervletek futtatására alkalmas konténerek:
  - Apache Tomcat
  - Oracle server
  - Websphere

    ...

Servlet életciklusa

- A CGI-vel ellentétben a szervlet osztály példányai nem jönnek létre/szünnek meg minden egyes kliens hívás alkalmából
- A webkonténer tetszőleges időpontban példányosíthatja a szervlet objektumot (célszerűen a szerver indításakor vagy az első hozzáféréskor), majd ezt követően ezt „életben tarthatja”, így egymás után több beérkező kérés kiszolgálását is elvégezheti ugyanaz a példány



Servlet interfész lényeges metódusai

- Szervlet inicializálása
  - public abstract void init(ServletConfig config) throws ServletException;

    A szervletet tartalmazó konténer hívja meg a szervlet példányosításakor. Csak egyszer fut le, garantáltan az első kérés kiszolgálása előtt.
    A paraméterként kapott *ServletConfig* objektum tartalmazza a szervletet tartalmazó környezet adatait.

    Amennyiben a metódus nem tudta inicializálni a szervletet, egy *UnavailableException* kivételt kell dobnia. A szervletet tartalmazó konténer ezen keresztül értesülhet a bekövetkezett hiba típusáról.

    Az alábbi információkat lehet továbbitani:
    - A hiba rövid ideig tartó vagy hosszú ideig tartó
    - Hosszabb ideig tartó hiba esetén egy hibaüzenet (ilyenkor nincs értelme újra próbálkozni)
    - Rövidebb ideig tartó hiba esetén egy időpont, hogy mennyi idő után érdemes újra megpróbálni az inicializálást

Servlet interfész lényeges metódusai

- Kérés kiszolgálása
  - public abstract void service(ServletRequest request, ServletResponse response) throws ServletException, IOException;

    A szervletet tartalmazó konténer hívja meg a szervletet érintő kérés beérkezésekor. A kérés paraméterei a *request* objektumon keresztül érhetők el, a választ a *response* objektumon keresztül lehet elküldeni.
    Részletesebben a *HTTPServlet* osztály ismeretésekor tárgyaljuk
- Szervlet megszüntetése
  - public abstract void destroy();

    A szervlet eltávolításakor hívja meg a konténer. Ezt követően nem jöhet kérés kiszolgálás.

    Érdemes itt felszabadítani a különböző erőforrásokat, például folyamatosan életbentartott adatbáziskapcsolatokat, megnyitott állományokat stb.

HTTPServlet osztály

- Egy szervletnek az interfész implementálásán túl további feltételeknek is meg kell felelnie. Ezért is célszerű a *javax.servlet.GenericServlet* osztályból származtatni (amikor ez lehetséges), ami ezeket a feladatokat már megoldja
- A HTTP alapon elérhető szervletek esetén egy még specifikusabb osztályból származtathatunk, ez a *javax.servlet.http.HTTPServlet*

  Ez megvalósítja az összes szervletekkel kapcsolatos követelményt, illetve számos szolgáltatást nyújt a HTTP kérések feldolgozásához
- Ezek az osztályok nem részei az alap JSDK osztálykönyvtárnak, ezért az őket tartalmazó JAR állományt külön kell hozzárendelnünk a projecthez (`servlet-api.jar` megtalálható a feltelepített Apache webszerver `lib` alkönyvtárában)

HTTPServlet osztály szolgáltatásai

- Mivel a HTTP protokoll számos parancsot ismer, a *HTTPServlet* service(...) metódusa megoldja helyettünk ezek azonosítását.
- Az osztály így az alábbi metódusokkal rendelkezik, amelyek opcionálisan felülírhatók, ezzel tudjuk a leszármazott szervletünk működését meghatározni:
  - protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, java.io.IOException
  - protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, java.io.IOException
  - protected void doHead(HttpServletRequest req, HttpServletResponse resp) throws ServletException, java.io.IOException
  - protected void doDelete(HttpServletRequest req, HttpServletResponse resp) throws ServletException, java.io.IOException
  - protected void doOptions(HttpServletRequest req, HttpServletResponse resp) throws ServletException, java.io.IOException

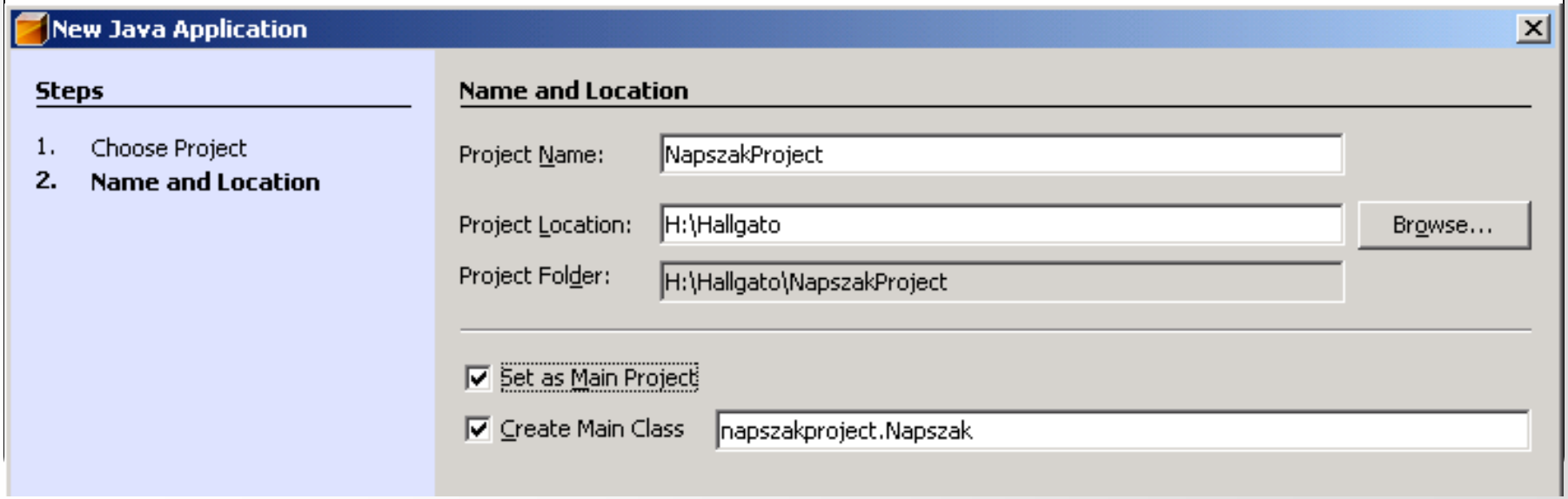
    ...
- Természetesen elérhető közvetlenül a *service(...)* metódus is, de célszerű a fenti metódusok használata

Készítsünk dinamikus weboldalt

Példa – Weboldal, ami napszaknak megfelelően köszön

```
package napszakproject;
import javax.servlet.*; import javax.servlet.http.*; import java.util.*; import java.io.*;

public class Napszak extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse resp)
                        throws ServletException, IOException  {
        int ora = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
```



Szervlet telepítése

- A lefordított osztály nem futtatható, hiszen nincs main(...) metódusa. A futtatáshoz el kell helyezni a konténerben, ahol majd a böngészővel érjük el
- Ha `Napszak.class` állományt bemásoljuk a `HallgatoWeb` könyvtárba, akkor az alábbi módon tudunk rá hivatkozni:

  `http://localhost/hweb/Napszak.class`

  Így viszont a webszerver a böngésző felé egyszerűen elküldi az állományt, mint bármelyik más dokumentumot
- A Tomcat szerverrel tehát tudatnunk kell, hogy az osztályt a szervletektől elvárt módon kezelje (betöltés/service metódus hívás)
- Ehhez helyezzük át az állományt az alábbi könyvtárba:

`h:\HallgatoWeb\WEB-INF\classes\napszakproject`

A böngészőben pedig az alábbi URL-t adjuk meg:

`http://localhost/hweb/servlet/napszakproject.Napszak`

Miért pont oda?

- A webalkalmazások (al)könyvtárszerkezete bizonyos szinten előre definiált. Ez alapján egy ilyen alkalmazásnak az alábbiakat kell tartalmaznia:
  - `webroot/`
    A HTTP-n keresztül közvetlenül elérhető állományok
  - `webroot/WEB-INF/classes/`
    A webalkalmazáshoz tartozó osztályok (pl. a szervletek)
    Természetesen itt is betartva a csomagtól függő konvenciókat!
  - `webroot/WEB-INF/lib/`
    A webalkalmazáshoz tartozó JAR állományok
  - `webroot/META-INF/`
    JAR állományokhoz hasonlóan metaadatok
  - `webroot/egyéb/`
    Szintén HTTP-vel közvetlenül elérhető állományok
  - `webroot/WEB-INF/web.xml`
    Ez az XML állomány tartalmazza a webalkalmazás paramétereit
- A „webroot” a Tomcat számára megadott alkalmazás gyökérkönyvtár. A `/hweb` esetén ez a `h:\HallgatoWeb`

Szervletek regisztrációja

- A `web.xml` fájlban minden szervletet regisztrálni kell. Jelenlegi példánkhoz az alábbi két bejegyzés (lenne) szükséges
- Szervlet regisztrációja

```
<servlet>
    <servlet-name>ElsoKoszonoServletunk</servlet-name>
    <servlet-class>napszakproject.Napszak</servlet-class>
</servlet>
```

Ez alapján tudja a Tomcat, hogy az *ElsokoszonoServletunk* nev alatt a *napszakproject.Napszak* osztályt értjük

- URL hozzárendelése (leképezés)

```
<servlet-mapping>
    <servlet-name>ElsoKoszonoServletunk</servlet-name>
    <url-pattern>/Koszonj</url-pattern>
</servlet-mapping>
```

Tehát ha egy kérés a `/Koszonj` alkönyvtárra vonatkozik, akkor valójában nem a fájlrendszerre gondolunk, hanem erre a szervletre

- Ezek alapján a szervlet így lenne elérhető:
  `http://localhost/hweb/Koszonj`

Szervletek regisztrációja, 2. lehetőség

- A példában nem volt szükség ezen lépések elvégzésére, ugyanis van egy lehetőség a szervletek egyenkénti regisztrációjának elkerülésére (jelenleg ez látható a `web.xml` állományban):

```
<servlet>
    <servlet-name>invoker</servlet-name>
    <servlet-class>org.apache.catalina.servlets.InvokerServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>invoker</servlet-name>
    <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

- Tehát a `/servlet`-re való hivatkozás valójában a megadott *InvokerServlet*-et indítja el. Ez egy speciális szervlet, aminek feladata, hogy elindítson egy paraméterként kapott másik servletet.

  Jelen esetben ez jelentősen egyszerűsíti az életünket ezért a gyakorlatok folyamán ennél maradunk, de iskolai példákon túl lehetőleg ne használjuk
- Ez alapján érthető, miért működik ez regisztráció nélkül:

`http://localhost/hweb/servlet/napszakproject.Napszak`

Házi feladat

A,
Készítsen függvénytáblázathoz hasonló matematikai táblázatokat készítő szervletet.
A lekérdezéskor generált oldal egymás alatt tartalmazza az alábbiakat:

- Egy 360 soros táblázat, ami tartalmazza egész fokonként a megfelelő
  - szinusz értéket
  - koszinusz értéket
  - tangens értéket
- Egy 10x10-es szorzótábla

B,
Készítse el egy áruház nyitóoldalát (még semmit sem tudunk az ügyfélről).

- Ehhez előbb készítsen egy Aru osztályt, ami tárolni tudja egy áru
  - nevét,
  - árát, illetve hogy akciós-e,
  - logikai értéket, hogy tartozik-e hozzá kép, és ha igen, akkor egy hivatkozást a képhez.
- Töltsön fel egy tömböt Aru objektumokkal, majd oldja meg az alábbiakat:
  - Az oldal lekérdezésekor véletlenszerűen válasszon ki 5 terméket, ezek jelenjenek meg
  - Az egymás alatt megjelenő áruk adatai azonosan formázva jelenjenek meg : név, alatta ár, jobbra kép (ezt célszerű nem látható táblázatokkal megoldani)
  - Amennyiben az áru éppen akciós, piros keretben vagy háttérrel jelenjen meg mindez
  - Amennyiben az áruhoz tartozik kép, akkor az oldalon ez is jelenjen meg, amennyiben nem, akkor pedig egy erre a célra fentartott képet ágyazzon be (a képeket a webalkalmazás /kepek alkönyvtárába másolja be, az oldalakon belül pedig értelemszerűen csak hivatkozásokat kell elhelyezni ezekre az erőforrásokra)

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

  15 – 86. 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