2. rész: JSP-k és szervletek készítése

Bakay Árpád
NETvisor kft
(30) 385 1711
arpad.bakay@netvisor.hu

Emlékeztető

- Servlet: Java kód, amely HTML outputot ír ki
  - Pl. `println()` parancsokkal
  - Rendszerint a `javax.servlet.http.HttpServlet` kiterjesztése
- JSP: HTML kód, dinamikus tartalmat generáló kiterjesztésekkel

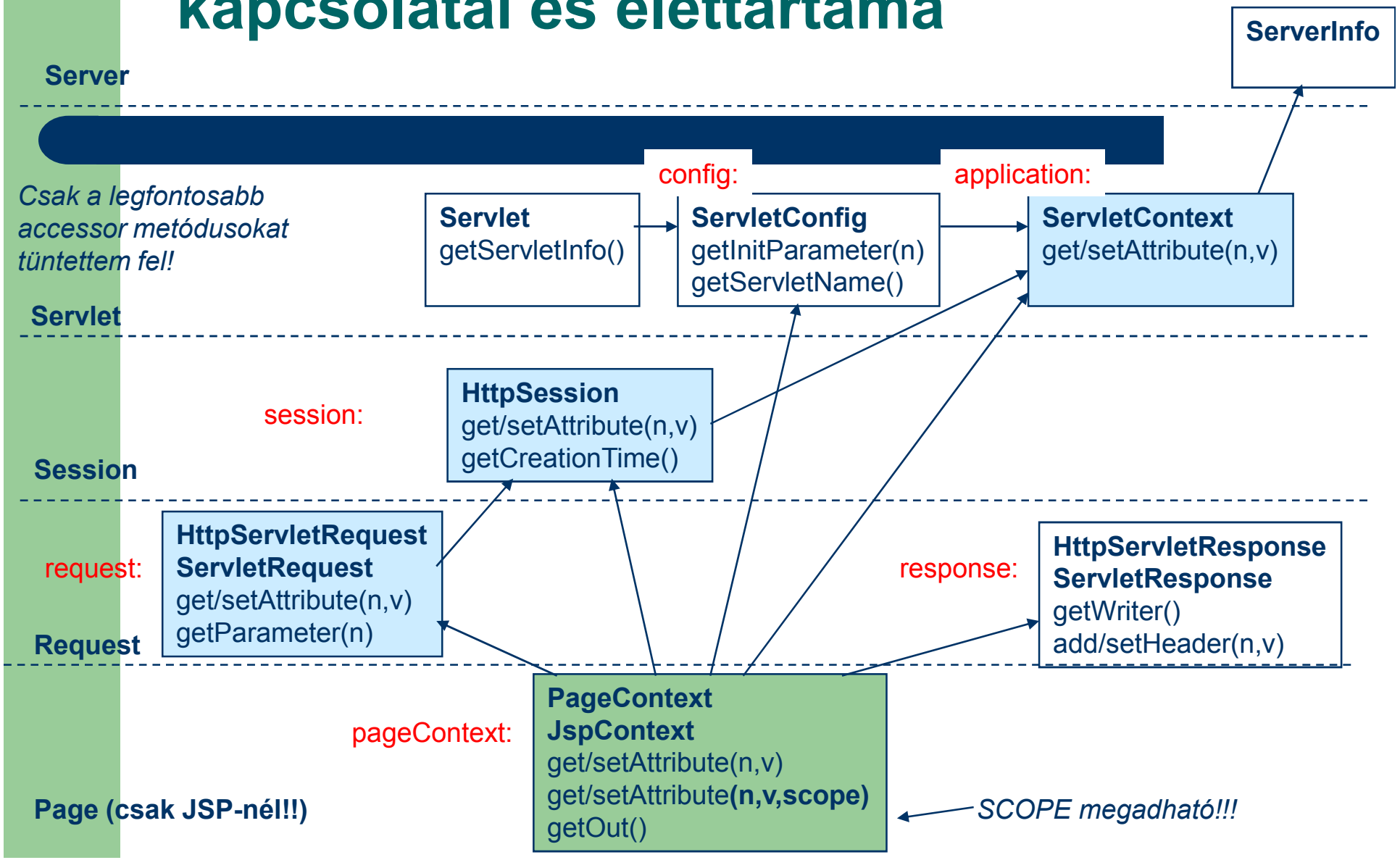
Szervlet példa

- `HttpServlet` kiterjesztése
- `doGet()` (vagy `doPost()`) implementálása
- `ServletContext:` hozzáférés a containerhez
  - Attribútumok különféle scope-okkal
- `request:` a kérés adatai
  - `getParameter(„par1")`
  - plusz, user, header, cookie adatok
- `response:` ennek van a `PrintWriter`-je, amire irunk
  - plusz: header, encoding, stb. dolgok

Szervlet példa - folyt

- `getServletContext().getRequestDispatcher(…).include(…)`
  - Másik servlettel, JSP-vel, HTML-lel generáljuk a tartalom egy részét
    - Pl: Banner
- `getServletContext().getRequestDispatcher(…).forward(…)`
  - Másik servlettel, JSP-vel, HTML-lel generáljuk az egész tartalmat
    - Pl: servlet számol és cselekszik, utána JSP megjelenít.

A servlet és JSP futtatókörnyezet beépített objektumai, property-jei kapcsolatai és élettartama



A JSP

- HTML oldal dinamikus tartalommal
- Szintaxisa a HTML kiterjesztése
  - (minden originál HTML valid JSP!)
  - A HTML 4-féle módon lett kiterjesztve:
    - Direktívák <%@ ….. %>
    - Új html elementek (tagok): <jsp:out …../>
    - Expression Language (EL): ${ bean.attr }
    - Scripting: <%! … %>, <% … %>, <%= … %>
- Szervletté fordul a telepítés után, és szervletként fut
  - Itt kell keresni: jdev/mywork/***app_name***/webapp/classes/.jsps/***_pagename***.java

K1: Direktívák

- Parancsok JSP fordítónak, ill. a futtató containernek
  - `<%@ page …. %>`

    Attrs: „import”, „session”, „errorPage”, „contentType” stb.
  - `<%@ include file=„URL” %>`
    - Behelyettesít egy másik file-t
  - `<%@ taglib uri=„aa.bb.com/dd” prefix=„aa”>`
    - Itt deklaráljuk, hogy akarunk majd vmilyen új tag-eket használni

K2: Tag.ek és tag library-k

- Tag-ek: Java nyelven definiálható nyelvi kiterjesztések

```
<html:image src="en.jpg"/>
```

- tag library
  - A tag szintaxist és viselkedést definálja
  - általában több logikailag összefüggő tag-et definiál, pl: `<c:choose/>`, `<c:if/>`, `<c:foreach/>`

Tag-ek használata

- Deklarálás: `<@ taglib … @>` ld. a direktíváknál
- Használata:

  <prefix:tagname attr_name=value />

  `Pl: <jsp.getproperty name="product" property="price"`
- Tag-ek és taglib-ek típusai
  - alap, beépített: `<jsp:xxx />` nem kell deklarálni
  - standard tag library-k (Java Standard Tag Library, JSTL)
    - Komponensei: core (**c:**), function (**fn:**), xml (**x:**), SQL (**sql:**), i18n (**fmt:**)
  - custom taglib: készen kapjuk, pl. a webapp framework-höz (JSF, Struts)
  - házilag, saját célra fejlesztett taglib

(K2/A) Alap Taglib

- `jsp:` namespace :
  - *jsp:useBean*: egy tetsz. javabean típusú változó deklarálása
  - *jsp:getProperty, jsp:setProperty*: bean property-k olvasása/írása
  - *jsp:import*: file befűzése
    - *jsp:param:* további paraméterek
  - *jsp:forward*: váltás másik lapra
  - *jsp:plugin*: applet, stb. meghívása
    - browser-független módon

Kitérő: JavaBean

- Elsősorban adattárolásra használt Java objektumok
- Célok: újrahasználhatóság, integrálhatóság, scripting és fejlesztői környezetek támogatása
- Követelmények (konvenciók)
  - Skalár vagy tömb példányváltozók, getter/setter metódusokkal
  - Van üres konstruktora

JavaBean – megjegyzések, furcsaságok

- Hogy valami JB-e az nem feketén-fehéren eldönthető:
- ‘Tiszta’ JB: van üres konstruktora, és csak szokásos property-k setter/getter metódusokkal.
  - **Csak adattárolásra használható**
  - **Itt nem vitás, hogy JB-ről van szó**
- ‘Trükkös’ JB: lehet más metódusa is, mint a setter/getter, és a property-k között is akadhat olyan, ami nem csupán beállít egy értéket, de valami viselkedést is indíthat!!
  - **Van enkapszulált viselkedése**
  - **Igazából nem JB, de JB-ként is használhatjuk…**
- **Az EJB-k nem JavaBean-ek!!!!**

JavaBean és JSP

- A javabean-ek jól kezelhetőg a JSP-kből
- A jsp: tag-ek használatával

```
<jsp:useBean id="status"
   class="hu.elte.pgy3.topicselect.web.statusbean"
                    scope="session"/>
…
 <jsp:setProperty property="username" name="status"
   param="uname"/>
 <jsp:setProperty property="password" name="status"
   value="mypwd"/>
…

<jsp.getProperty property=„userfullname” name=status”>
```

jsp:usebean példa

```
<jsp:useBean id="status" class="hu.elte.pgy3.topicselect.web.statusbean"
                scope="session"/>
…
<jsp:setProperty property="username" name="status" param="uname"/>
<jsp:setProperty property="password" name="status" param="pwd"/>

…

<jsp.getProperty property=„userfullname” name=status”>
```

(K2/B) Standard extension tags: JSTL
JavaServer Pages Standard Tag Library

- 5 szekció:
- `c:` (= „core”): logikai vezérlés, változók stb.
  - c:set: EL változó beállítása
    <c:set var=„bb.loginerror" scope="session" value=„true"/>
  - c:foreach ,c:if, c:choose-c:when-c:otherwise
    <c:if test="${bb.loginerror != null} " >
    …body…
    </c:if>
- `xml:, sql:, fmt:`
  - (XML, SQL, formattálás, többnyelvű interfészek)
- `fn:` (függvények)
  - fn:length, fn:trim, fn:substring, etc.

Expression Language -- EL kifejezés (mindjárt tanuljuk)

(K2/D) Saját taglib készítése és használata

- Tag handler: Java nyelven
  - Tag Library Definition: XML file rögzíti a paramétereket, stb. az egyes tag-ekhez
  - Tag handler: implementáció
    - Minden tag-hez egy (v. több) osztály
- Tagfile: JSP nyelven (JSP 2.0 óta)
  - TLD opcionális (magától generálja az információt)
  - Alkalmas újrahasznosítható JSP „modulok” készítésére (és scriptletek elrejtésére)

K3: Expression Language (EL)

- Szintaxis:

  ${ expr }

  - Kifejezés, mellékhatás nélkül!!!
  - Szokásos logikai és aritmetikai operátorok
  - Automatikus típuskonverzió (mint JavaScript)
  - Property-k és collection member elérése ‘.’ v. ‘[ ]’ operátorokkal
  - JSTL függvények: fn:length(„asddf”)
- Használható:
  - Korábban csak a tag-ek attribútumaiban
  - Most már (JSP 2.0) mindenhol

    Pl: <c:if test=”**${bb.loginerror != null}**” >

    Bejelentkezési hiba: **${bb.loginerrormessage}**

    </c:if>

Implicit Objects in EL

Egy EL azonosító általában valamelyik scope-ban definiált változóra hivatkozik, de ezek foglalt nevek:

- **pageContext** *javax.servlet.jsp.PageContext*
  The context for the JSP page. Provides access to various objects, e.g. request, response, session
- **pageScope, requestScope sessionScope applicationScope**
  java.util.Map Map scoped variable names to their values.
- **param, paramValues** *java.util.Map* request parameter maps for accessing single and multi-valued parameters
- **header, headervalues, cookie**
  java.util.Map request header and cookie information
- **initParam**
  java.util.Map servlet context initialization parameters

K4: JSP Scripting

- JSP Scripting elements
  - declaration: `<%! String name; %>`
  - scriptlet `<% if(name="windows") { %>`
  - expression: `<%= myVar %>`
- Sokoldalú, de áttekinthetetlen → kerüljük, ha lehet!

Adatok elérése JSP, az EL és a scriptlet technológiával

- Nem triviális:

JSP: *Bean property értékadás request paraméter alapján:*

```
<jsp:setProperty name="bor"
    property = "name" param="borName"/>
```

EL: *Request paraméter és bean instance member lekérdezése:*

```
${param.borName},
${bor.name}
```

Scriptlet: *Request paraméter és bean objektum lekérdezése:*

```
request.getParameter("borName")
session.getAttribute("bor")
```

Példa2: session-szkópú bean használata a 3 technikával

JSP:

```
<jsp:usebean id="bor" scope="session" />
<jsp:getProperty name="bor" property="year")
```

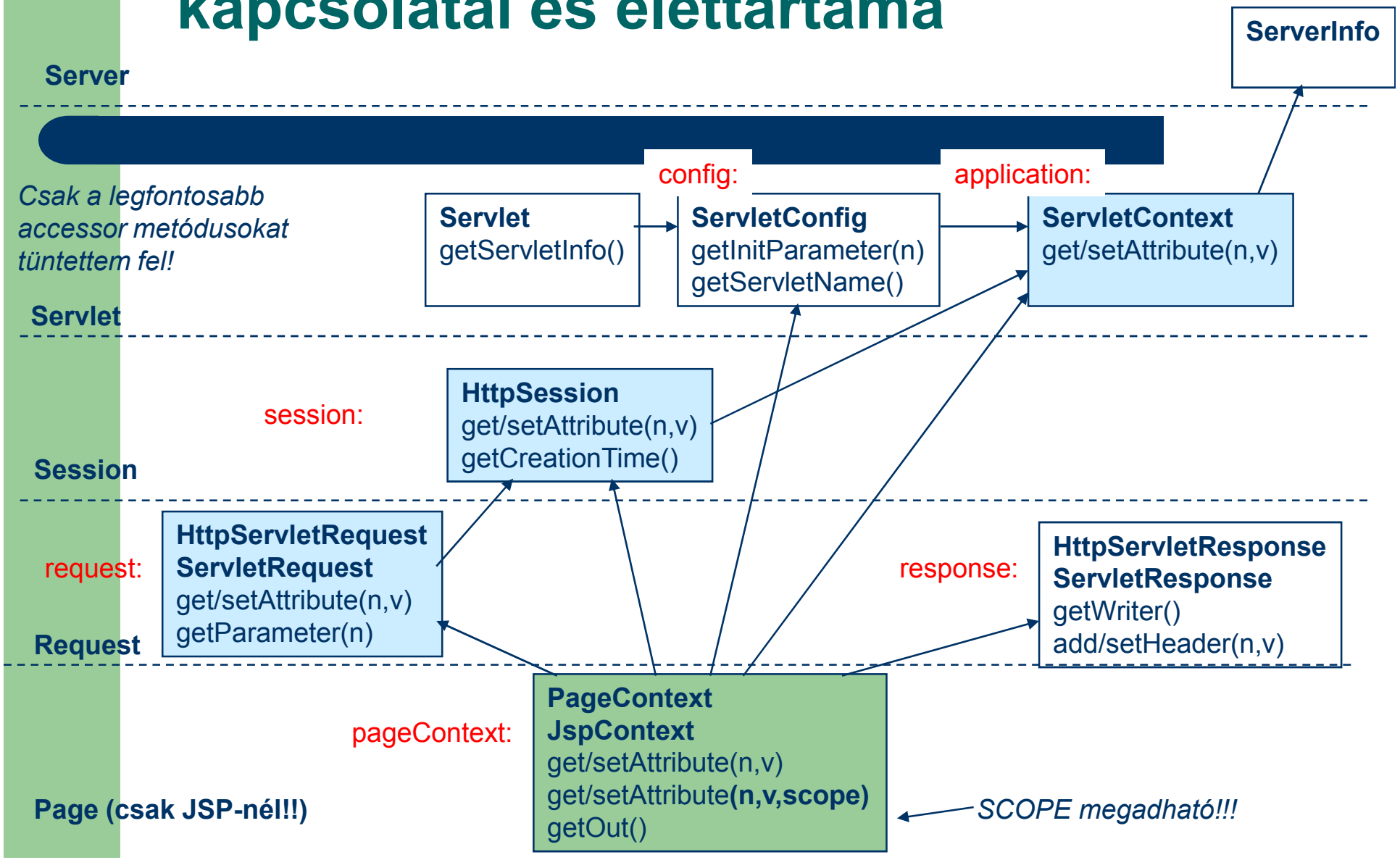
EL:

```
${bor}  ${bor.year}
```

JSP Scriptlet ill. Scriptlet Expression:

```
<% BorBean mybor=new BorBean();
   session.setAttribute("bor", mybor);
// vagy: pageContext.setAttribute("bor", mybor, SESSION_SCOPE);
%>
<%=
 ((BorBean)session.getAttribute("bor")).getYear()
%>
```

A servlet és JSP futtatókörnyezet beépített objektumai, propety-jei kapcsolatai és élettartama



Demo, példaprogram

- Servletek és JSP-k indítása HTML form-ból
  - Servlet
  - JSP alap `jsp:` tag-ekkel + JavaBean
  - JSP scriptinggel
  - JSP saját fejlesztésű custom tag használatával

A technológiák összehasonlítása

- Servletet prezentációra ne használjunk
  - Csak az adatváltoztatási műveletek kezelésére, a megjelenítendő adatok előkészítésére
- Jsp: tagek + JavaBean
  - Egyszerű lapokhoz tiszta struktúra, de nem túl rugalmas, kell írni egy Bean-t.
- Saját taglib
  - Egyetlen alkalmazáshoz egy komplett taglb jellemzően nem éri meg
    - Webapp frameworkhoz már igen!
- Scripting
  - Kusza keverék kód, de semmi más nem kell hozzá, csak a JSP. Ha lehet kerüljük!!

További megjegyzések

- Technikák akár egy lapon belül is keverhetők
  - Adatok átjárhatóságát megbeszéltük!
- Nagy alkalmazást rendszerint vmilyen web application frameworkkel fejlesztenek: JSF, Struts, stb.

Köszönöm a figyelmet!

- arpad.bakay@netvisor.hu