Java programozási nyelv 2012-2013/ősz

7. óra

Menetkövetés lehetőségei

Menetkövetés technikái

Java szolgáltatások

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

Menetkövetés lehetőségei

Témakörök

Szervlet életciklusa, menetkövetés

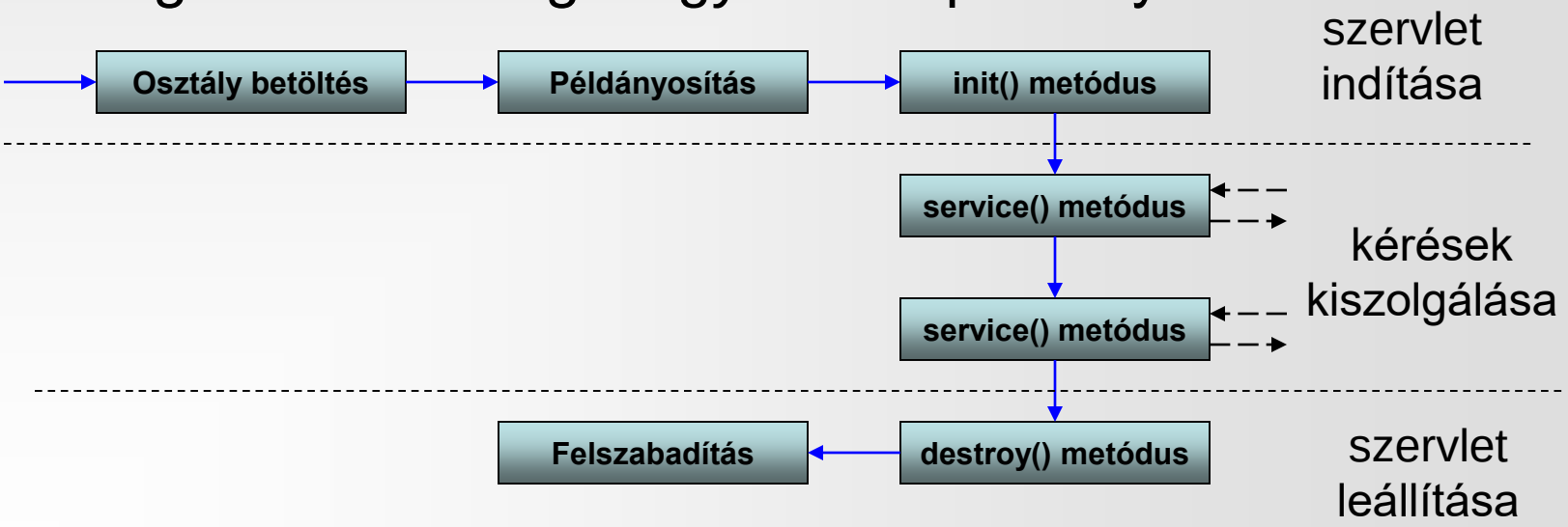
Rejtett űrlapmezők/URL újraírás

Cookie

Java Session

Szervlet életciklusa (5. óra)

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

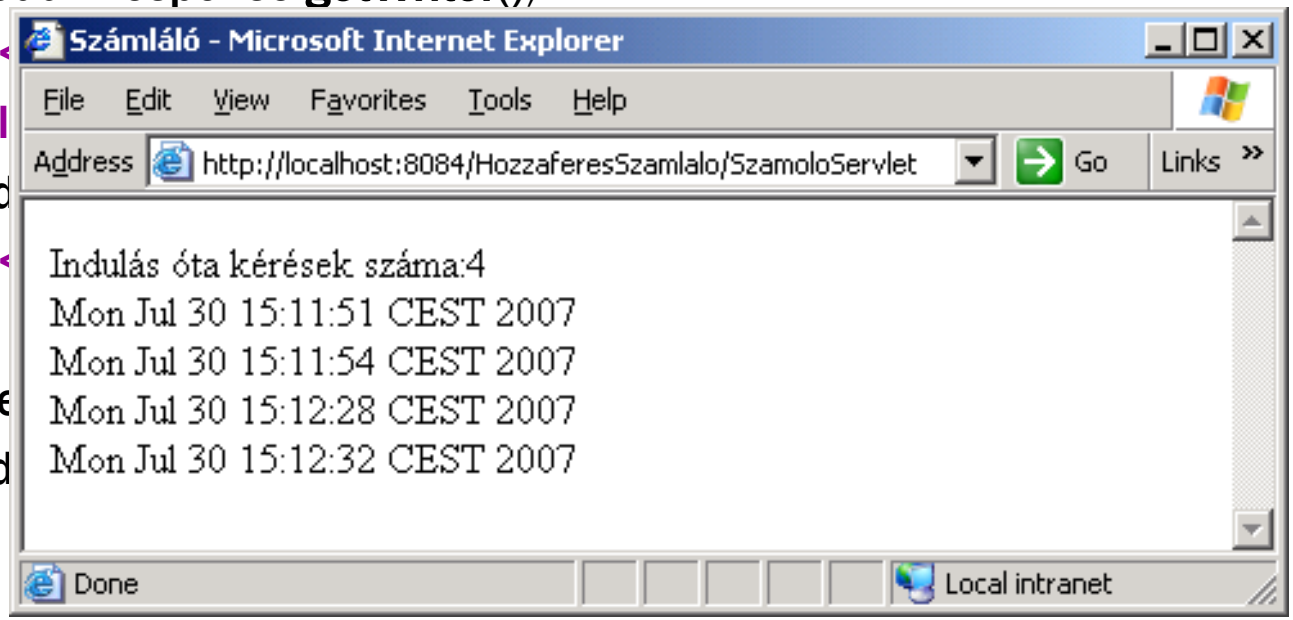


Szervlet életciklusa során élő változók

- A HTTP egy állapotnélküli protokoll, tehát az egymást követő kérések egymástól függetlenek. Hagyományos módon (CGI) a több kérésen átívelő folyamatok adatait valahogy el kellett tárolni, például:
  - állományokban
  - adatbázisokban
- Mivel a szervlet egy példánya egymás után több kérést is kiszolgálhat, ezért ez az ilyen típusú feladatok során nagy előnyt jelenthet
- A CGIvel ellentétben nem kell minden kérés után újra felépíteni az üzleti logikát tartalmazó struktúrákat, hanem az adatok folyamatosan tárolhatók:
  - a szervlet példányváltozóiban
  - bármelyik osztály statikus változóiban
  - kontextus attribútumaiban
  - session attribútumaiban (később lesz róla szó)
- A szervlet leállításakor ezek természetesen elvesznek

Kérések számát és időpontját tároló szervlet

```
public class SzamoloServlet extends HttpServlet {
    int darab;
    StringBuffer idopontok;
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
        darab++; idopontok.append(new java.util.Date() + "<br/>");
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<
        out.println("I
        out.println(id
        out.println("<
    }
    public void init(Se
        darab = 0; id
    }
}
```



Menetkövetés feladata

- A kérésektől független adatokat ezzel a módszerrel egyszerűen tárolhatjuk. Ezeket gyakran célszerű is folyamatosan a memóriában tartani, pl.:
  - előző órai feladatból a csatornák listája
  - megjelenést/kattintást számláló bannerek
- A gyakorlatban azonban sokszor az egymást követő kéréseken átívelő adatok nem általánosak, hanem csak egy klienshez kapcsolódnak, pl.:
  - azonosított felhasználókkal dolgozó rendszer (csak egyszer kell belépni a felhasználónak, utána a kilépésig tetszőleges számú műveletet végezhet, nem kell mindig újra bejelentkeznie)
  - webáruház esetén a bevásárlókocsi (nem lenne praktikus, ha minden lépés után a felhasználónak meg kellene adnia az összes addig kiválasztott terméket)
- Ilyenkor tehát az adatok folyamatos eltárolásán túl még azt a feladatot is meg kell oldani, hogy egy beérkező kérés során kiválasszuk a tárolt adatok közül azokat, amelyek az aktuális klienshez tartoznak

Feladat – mondat analízise

- Készítsük el az alábbi egyszerű webes alkalmazást:
  - A nyitóképernyőn legyen lehetőség egy mondat megadására
  - Az „Elemzés” gombra kattintva jelenjen meg a következő képernyőn, hogy a megadott mondat hány kis- és hány darab nagybetűt tartalmaz (az egyszerűség kedvéért minden nem kis latin betűt tekintsünk nagynak)
  - Ezt követően legyen lehetőség kiválasztani, hogy a kis- vagy a nagybetűket szeretnénk listázni. A következő oldalon a feltételnek megfelelően jelenjenek meg az első oldalon megadott mondat betűi
- A probléma: a mondatot csak egyszer szeretnénk megadni: a nyitóképernyőn, amit az első kéréskor feldolgozunk. A második kéréshez már nem írjuk be újra, ezért a szervletnek „emlékeznie” kell arra, hogy az előző kérés során milyen mondatot kapott
- Technikailag ez megfelel annak az esetnek, amikor a felhasználó a nyitóképernyőn bejelentkezik, ezt követően pedig a szervlet már „ismeri” őt

Feladat – megoldási lehetőségek

- Tároljuk el a megadott mondatot a szervlet egy mezőjében, ahogy az előző példában láttuk
  - Amennyiben egyidőben többen is csatlakozhatnak az oldalhoz, akkor ez hibás eredményhez vezethet. Ha az A felhasználó első és a második kérése között eltelt időben egy B felhasználó is beírt egy mondatot, akkor az A harmadik oldalán is ennek a mondatnak a betűi jelennek meg
- Mivel a kérés adatait tartalmazó objektum tartalmazza a kliens gép IP címét, azonosítsunk ez alapján
  - Bár a Windows rendszereken nem általános, de egyidőben többen is használhatnak egy gépet
  - Proxy szerverek közbeiktatása során a szerver a proxy mögött lévő összes gépet azonos IP címként látja
- Működő megoldások
  - Rejtett űrlapmező/URL újraírás
  - Cookie használata
  - Session objektum használata

Feladat – induló képernyő

Az induló képernyőn lehessen választani a négyféle módszer között:

```
<html>
  <head><title>Mondatanalizátor induló képernyő</title></head>
  <body>
    Rejtett űrlapmezővel
    <form action="MondatSzervletA" method="post">
      A mondat:<input type="text" name="mondat"/><input type="submit" value="Elemzés"/>
    </form>
    <br/>URL újraírással
    <form action="MondatSzervletB" method="post">
      A mondat:<input type="text" name="mondat"/><input type="submit" value="Elemzés"/>
    </form>
    <br/>Cookie használatával
    <form action="MondatSzervletC" method="post">
      A mondat:<input type="text" name="mondat"/><input type="submit" value="Elemzés"/>
    </form>
    <br/>Session objektum használatával
    <form action="MondatSzervletD" method="post">
      A mondat:<input type="text" name="mondat"/><input type="submit" value="Elemzés"/>
    </form>
</body></html>
```

Feladat – üzleti logika

A mondatok analízisét (megoldástól függetlenül) ez az osztály végzi:

```
public class MondatFelbonto {
	String mondat;
	public MondatFelbonto(String mondat) { this.mondat = mondat; }
	public int megszamlalas(boolean kisbetu) {
		int num = 0;
		for(int i = 0; i < mondat.length(); i++)
			if (Character.isLowerCase(mondat.charAt(i)) == kisbetu) num++;
		return num;
	}
	public String listazas(boolean kisbetu) {
		StringBuffer sb = new StringBuffer();
		for(int i = 0; i < mondat.length(); i++)
			if (Character.isLowerCase(mondat.charAt(i)) == kisbetu)
				sb.append(mondat.charAt(i)+ "<br/>");
		return sb.toString();
	}
}
```

Menetkövetés lehetőségei

Témakörök

Szervlet életciklusa, menetkövetés

Rejtett űrlapmezők/URL újraírás

Cookie

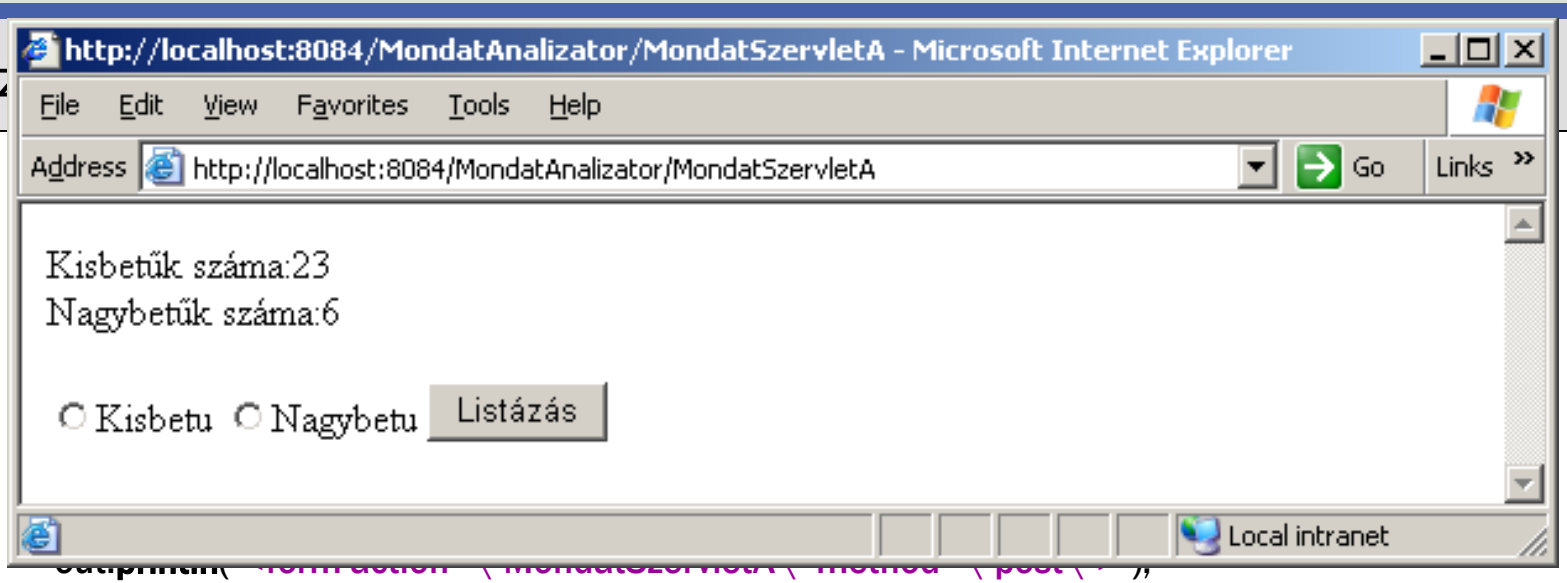
Java Session

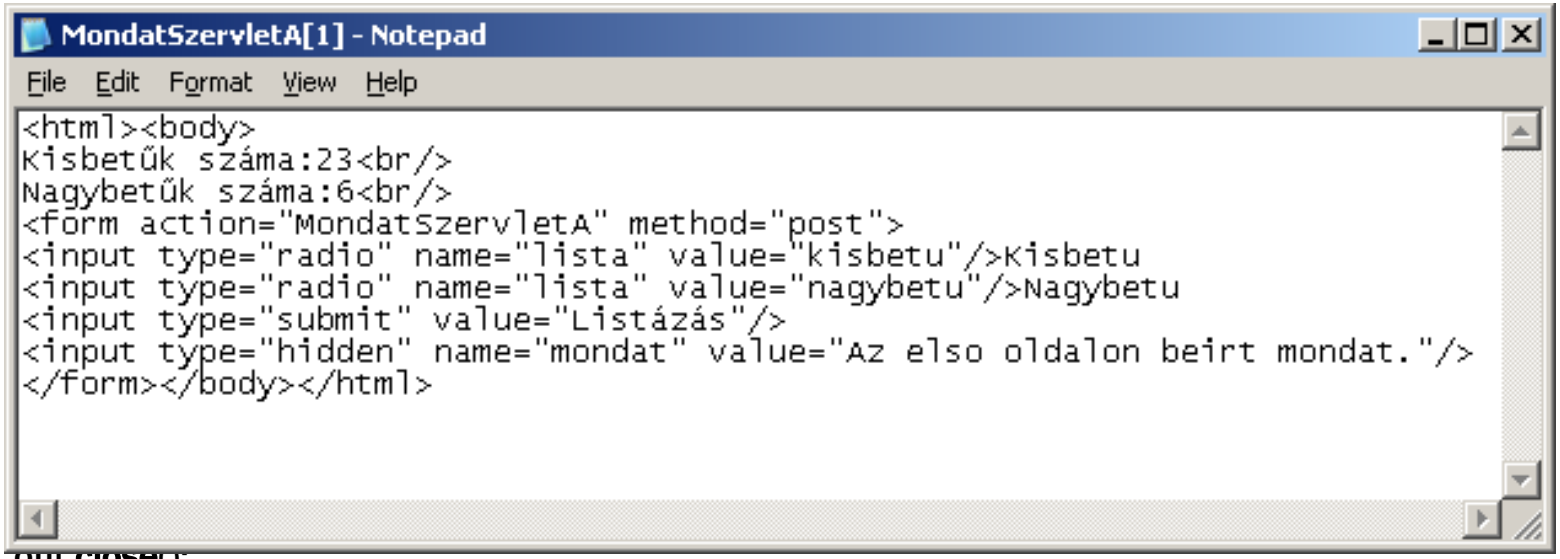
Rejtett űrlapmezők használata

- A feladat megoldásában az jelenti a problémát, hogy a második kérés kiszolgálása során (hiszen az az elsőtől egy teljesen független HTTP csomagként érkezik meg) már nem ismerjük az első oldalon beírt mondatot
- Egyszerű megoldás lenne, ha a felhasználó újra beírná a mondatot, ezt azonban kényelmi szempontokból el szeretnénk kerülni
- Megoldás erre a problémára, ha a második lapon ugyan ismét bekérjük a mondatot, azonban (mivel ezt a lapot az első kérés feldolgozásakor generáljuk, amikor a mondat még ismert) alapértelmezett értékként a mondatot is beleírjuk
- Ha nem szeretnénk, hogy a felhasználó lássa ezt a technikai mezőt (és főleg nem szeretnénk, hogy át tudja írni), használhatunk rejtett űrlapmezőt

Rejtett űrlapmező példa

Az





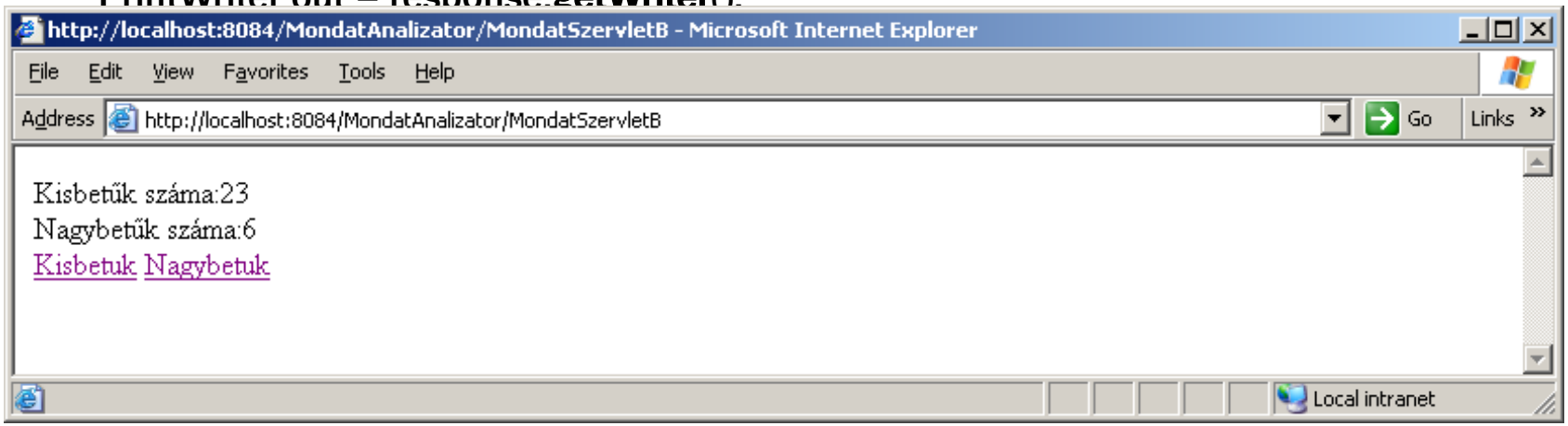
out.close();

URL újraírás példa

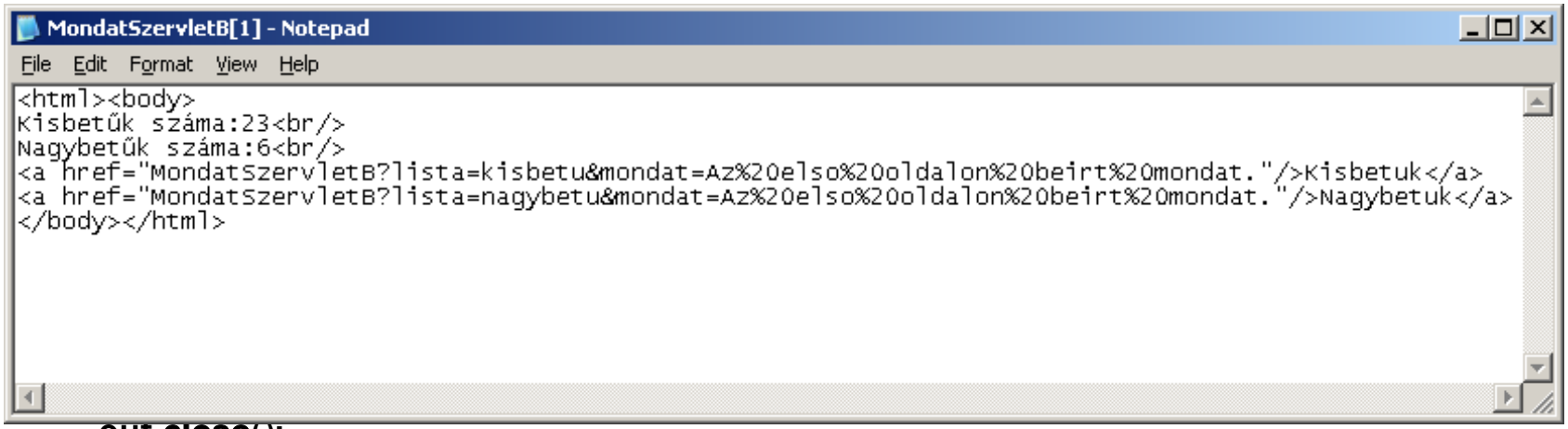
- A rejtett mezők használata megoldotta a problémát, azonban könnyen belátható, hogy ez nem minden esetben használható
- Azokban az esetekben, ahol egy oldalról való továbblépéshez nincs szükség további adatok megadására, általában csak egy hivatkozást használunk. Ilyenkor azonban nincs form, így nincs lehetőség a hidden mező felvételére sem
- Ilyen eset gyakran fennáll:
  - A továbblépéshez csak egy „következő” képre kell kattintani
  - A továbblépéshez több útvonal közül is választhatunk, de nincs szükség további adatok megadására
  - A program folyamatból átlépünk pl. a súgó oldalra, majd vissza
- A továbblépéshez ilyenkor egyszerű hivatkozásokat is elég felvenni, ahol a (GET metódusnál) megismert módon van lehetőségünk paraméterek átadására is: Pl. `http://host/SzervletNev?nev=feri&kor=15`

URL újraírás példa

```
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
```




```
kodoltmondat +"\"/>Kisbetuk</a>");
```




```
out.close();
```

Rejtett űrlapmező/URL újraírás jellemzői

- A megismert módszerek előnyei
  - Minden helyzetben használható, mivel csak a szabványos HTTP/HTML elemeken alapul
  - Egyszerű, könnyen érthető a kezelése (egyszerűsíti a hibakezelést)
- A megismert módszerek hátrányai
  - Nehézkes a kezelése, minden egyes formot ki kell egészíteni a rejtett mezőkkel, illetve minden hivatkozást ki kell egészíteni a kiegészítő paraméterekkel
  - Hivatkozások esetén a böngésző címsorában látható minden átadott paraméter
- Szervlet tulajdonságainak kihasználása
  - Ha több adatot is továbbítani kell, kihasználhatjuk a szervlet folyamatos tárolási lehetőségeit. A felhasználónál ilyenkor csak egy azonosító kulcsot célszerű mindig továbbítani, a szervlet pedig egy hasító táblázat alapján minden kéréshez hozzá tudja rendelni a hozzá tartozó eltárolt adatokat

Menetkövetés lehetőségei

Témakörök

Szervlet életciklusa, menetkövetés

Rejtett űrlapmezők/URL újraírás

Cookie

Java Session

Cookie/süti/pite jellemzői

- A cookie működése
  - A cookie egy rövid szöveg, amelyet a webkiszolgáló küld a kliensnek, amely azt a későbbi kérések során visszaküldi a szervernek. A böngészők általában állományokban tárolják ezek adatait, tehát még egy böngésző újraindítás után is léteznek
  - A cookiet a böngésző csak annak a szervernek küldi el, amelyiktől azt kapta (kivéve, ha a szerver ezt másképp definiálja, például adott domainen belül minden kiszolgáló kapja meg)
  - Egy szerver több cookiet is eltárolhat a kliens gépén
- Cookie osztály
  - Egy cookiet a *javax.servlet.http.Cookie* osztály egy példánya reprezentál
  - Egy cookie az alábbi tulajdonságokkal rendelkezik:
    - név
    - tartalom
    - verziószám, megjegyzés
    - lejárati idő
    - domain, útvonal

Cookie osztály

- Új példány létrehozása a konstruktorral
  - public Cookie(String name, String value)
    Megadott nevű és tartalmú cookie létrehozása
- Tulajdonságok beállítása
  - public void setName(String name)
    Név módosítása. A név csak betűket és számokat tartalmazhat
  - public void setValue(String newValue)
    Tartalom módosítása
  - public void setComment(String purpose)
    Egy megjegyzést lehet meghatározni, ami látható lesz a böngészőben is
  - public void setMaxAge(int expiry)
    A cookie létrehozástól számított élettartamát lehet megadni másodpercben. Az megadott életkor elérését követően a böngésző már nem küldi el a kiszolgáló felé és akár törölheti is (negatív szám megadása végtelen életkort jelent)

Cookie osztály

- Tulajdonságok beállítása
  - public void setPath(String uri)
    Paramétere egy útvonalrészlet, a cookie csak azon oldalak számára lesz látható, amelyek megfelelnek az itt megadott útvonalnak
  - public void getDomain(String pattern)
    Paramétere egy ponttal kezdődő domain név lehet. A böngésző az eltárolt cookiet minden kiszolgálónak el fogja küldeni, amelyik neve megfelel az itt megadottnak. Alapértelmezett a szervletet futtató teljes domain
- Tulajdonságok lekérdezése
  - public String getName()
  - public String getValue()
  - public String getComment()
  - public int getMaxAge()
  - public String getPath()
  - public String getDomain()

Cookie küldése/fogadása

- Cookie küldése során a szervlet választ tartalmazó objektumához (*HTTPServletResponse*) kell csatolni a tartalom mellett az időközben létrehozott cookiekat:
  - public void addCookie(Cookie cookie)
    A paraméterként átadott *cookie* objektumot csatolja a válaszhoz. A kliens böngésző a válasz megérkezésekor automatikusan eltárolja majd a küldött adatokat
- Cookie fogadása ehhez hasonló módon történik. A szervlethez érkező kérés objektum (*HTTPServletRequest*) tartalmazza a böngésző által (a szervletet futtató domain számára elküldendő) cookie objektumokat. Az alábbi metódussal kérdezhetők le:
  - public Cookie[] getCookies()
    A metódus visszatérési értéke a böngésző által elküldött cookiekat tartalmazó tömb. Ha a böngésző nem küldött egyet sem, akkor implementációtól függően a metódus visszatérési értéke null vagy egy üres tömb (mindkettőt érdemes ellenőrizni) Név szerinti lekérdezés nincs, a tömbben kézzel kell keresni

Megoldás cookie használatával

```
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
if (request.getParameter("lista") == null) {
        Cookie mondatCookie = new Cookie("mondatcookie", request.getParameter("mondat"));
        response.addCookie(mondatCookie);
        MondatFelbonto mf = new MondatFelbonto(request.getParameter("mondat"));
        out.println("<html><body>");
        out.println("Kisbetűk száma:"+ mf.megszamlalas(true) +"<br/>");
        out.println("Nagybetűk száma:"+ mf.megszamlalas(false) +"<br/>");
        out.println("<a href=\"MondatSzervletC?lista=kisbetu\"/>Kisbetuk</a>");
        out.println("<a href=\"MondatSzervletC?lista=nagybetu\"/>Nagybetuk</a>");
        out.println("</body></html>");
} else {
        for(int i = 0; i < request.getCookies().length; i++)
            if (request.getCookies()[i].getName().equals("mondatcookie")) {
                MondatFelbonto mf = new MondatFelbonto(request.getCookies()[i].getValue());
                out.println("A kért betűk listája:<br/>");
                out.println(mf.listazas(request.getParameter("lista").equals("kisbetu")));
            }
}
out.close();
```

Cookie jellemzői

- A módszer előnyei
  - Nincs szükség az oldalak tartalmának módosítására a továbbítandó adatok miatt, ezek a generált tartalomtól függetlenül tárolódnak
  - Ugyanennek köszönhetően több oldalon átívelő folyamatok is egyszerűen megoldhatók segítségével (áttekinthető kód)
  - Nyelvi szinten magasszintű támogatás
- A megismert módszerek hátrányai
  - Működésének feltételei nem minden esetben adottak, pl.:
    - felhasználó a böngésző beállításai között letiltotta a sütiket
    - a böngésző nem tudja kezeli a cookiekat (pl. mobiltelefonok)
  - Cookieban közvetlenül tárolható adatmennyiség korlátos
  - Nyomkövetés, hibakezelés meglehetősen nehezen kivitelezhető
- Szervlet tulajdonságainak kihasználása
  - A cookieban is csak egy azonosítót célszerű tárolni, ehhez pedig a szervleten belül rendelni a többi adatot
  - Mindig vegyük figyelembe, hogy a cookie a kliens gépen egyszerűen hozzáférhető és módosítható!

Menetkövetés lehetőségei

Témakörök

Szervlet életciklusa, menetkövetés

Rejtett űrlapmezők/URL újraírás

Cookie

Java Session

Servlet API támogatása

- Menetkövetés támogatása
  - Az eddig megismert technikák közvetlen használata helyett felhasználható a Servlet API által nyújtott támogatás
  - Az egyes folyamatok kezelését ilyenkor a szervletet tartalmazó konténer végzi
  - Működése hasonló az eddig megismertekhez, egy véletlen menetazonosítót tárol el a felhasználónál, majd a kérések beérkezését követően ez alapján hozzárendeli a szerver oldalon tárolt adatokat
  - Amennyiben a kliens gépen engedélyezett a cookiek használata, az azonosítót ezek segítségével tárolja el. Amennyiben ez nem oldható meg, akkor az URL felülírást alkalmazza
- Java implementáció
  - A szervletekben a *javax.servlet.http.HttpSession* objektumokon keresztül érhető el a fenti szolgáltatás
  - A beérkező kéréshez tartozó *HttpSession* objektumhoz a kérést reprezentáló *HttpRequest* objektumon kereszül lehet hozzáférni

Feladatok a kérés/válasz objektumokkal

- Új példány létrehozása
  - public HttpSession getSession(boolean create)
    A kéréshez tartozó menetobjektumhoz a kérést reprezentáló **HttpServletRequest** objektum metódusán át lehet hozzáférni. Amennyiben a kéréshez még nem tartozik menetobjektum, a paraméter értékétől függően (ha igaz) létrehoz egy újat
  - public HttpSession getSession()
    Hasonló, csak paraméterként igaz értéket feltételez
- Menet kezelése letiltott cookiek esetén
  - public String encodeURL(String url)
    Mivel a szervlet által generált HTML oldal tetszőleges helyen tartalmazhat hivatkozásokat, ezeket a konténer automatikusan nem találhatja meg. Ezért ezeket a **HttpServletResponse** fenti metódusával ki kell cserélni a visszaadott címre
  - public String encodeRedirectURL(String url)
    Az előzőhöz hasonló, de kéréstovábbítás során használandó

HttpSession osztály metódusai

- Tulajdonságok kezelése
  - public void setAttribute(String name, Object value)
    A megadott objektumot hozzákapcsolja a menetobjektumhoz az első paraméterben megadott néven. Ha már létezett ilyen kötés, akkor kicseréli az előző objektumot a most átadottra.
    Nem csak szöveget, hanem összetett objektumokat is eltárolhatunk így, hiszen ez nem kerül el a klienshez
  - public Object getAttribute(String name)
    Az előző metódussal eltárolt objektumokat lehet lekérdezni név alapján. Ha nincs ilyen név, visszatérési értéke null
  - public void removeAttribute(String name)
    Törli a megadott névhez tartozó kötést
  - public Enumeration getAttributeNames()
    Visszatérési értéke egy felsorolás, amelyik tartalmazza a menetkövetési objektumban található aktuális kötések neveit. Visszatérési értéke üres, ha nincsenek ilyenek

HttpSession osztály metódusai

- Lejárati idő kezelése
  - public void setMaxInactiveInterval(int secs)
    Amennyiben egy kérést követően a másodpercben megadott lejárati időn belül nem érkezik újabb kérés, a menetobjektum érvénytelenítődik.
    Negatív érték esetén ez soha nem következik be
  - public int getMaxInactiveInterval()
    Visszaadja az alapértelmezett, vagy az előző metódussal felülbírált aktuálisan érvényes lejárati időt
- Egyéb metódusok
  - public String getID()
    Visszatérési értéke a folyamathoz tartozó technikai azonosító
  - public void invalidate()
    Menet érvénytelenítése (pl. kilépéskor)
  - public long getCreationTime()
    Menetobjektum létrehozásának ideje
  - public long getLastAccessedTime()
    Kliens utolsó kérésének ideje

Megoldás HttpSession használatával

```
response.setContentType("text/html;charset=UTF-8");
PrintWriter out = response.getWriter();
HttpSession sess = request.getSession();
if (request.getParameter("lista") == null) {
        MondatFelbonto mf = new MondatFelbonto(request.getParameter("mondat"));
        sess.setAttribute("szervletunk.felbonto", mf);
        out.println("<html><body>");
        out.println("Kisbetűk száma:"+ mf.megszamlalas(true) +"<br/>");
        out.println("Nagybetűk száma:"+ mf.megszamlalas(false) +"<br/>");
        out.println("<a href=\""+response.encodeURL("MondatSzervletD?lista=kisbetu")+"\"/>Kisbetuk</a>");
        out.println("<a href=\""+response.encodeURL("MondatSzervletD?lista=nagyb")+"\"/>Nagybetuk</a>");
        out.println("</body></html>");
} else {
        MondatFelbonto mf = (MondatFelbonto)sess.getAttribute("szervletunk.felbonto");
        out.println("A kért betűk listája:<br/>");
        out.println(mf.listazas(request.getParameter("lista").equals("kisbetu")));
}
out.close();
```

Vegyük észre, hogy nem a beírt mondatot, hanem magát a mondatfelbontó objektumot tároltuk el!

HttpSession jellemzői

- A módszer előnyei
  - Egyszerűen használható, rövid kódot eredményez
  - Szabványos technika, így használata jól áttekinthető
  - OOP környezetbe jól illeszkedik:
    - Közvetlenül objektumokat tárolhatunk
    - *HttpSession* osztályból származtatott osztályok is használhatók
  - Cookiek megléte esetén azt használja, de hiányuk esetén is működőképes
  - Menetkövető objektumban tárolt adatok mennyiségére nincs korlát (hiszen nem kerülnek át a klienshez)
  - Szervletek között átirányított kérések esetén is könnyen hozzáférhetők az itt tárolt adatok
- A megismert módszerek hátrányai
  - Ha minden helyzetben működő változatra van szükség, az első módszerhez hasonlóan minden hivatkozásnál szükség van az URL felülírásra
  - Nyomkövetés, hibakezelés meglehetősen nehezen kivitelezhető (működésére nincs sok befolyásunk)

Gyakorló feladat

Egészítse ki az előző órán készített valutaváltó programot az alábbiakkal:

- Mivel a szervlet folyamatosan él, és a valutaárfolyamok a beérkező kérésektől függetlenek, ezért oldja meg, hogy az induló árfolyamokat tartalmazó tömb a szervlet inicializálásakor jöjjön létre, és éljen egészen a szervlet leállításáig
- Amennyiben a felhasználó megadta az átváltáshoz szükséges adatokat, a következő oldalon a kiszámolt fizetendő összeg adatai alatt továbbra is jelenjen meg az első oldalon megismert űrlap, így lehessen kezdeményezni újabb tranzakciót
- Az első váltást követően az oldal tetején mindig jelenjen meg egy táblázat az eddigi tranzakciók adataival:
  - tranzakció ideje
  - átváltott összeg/valuta
  - kifizetett összeg/valuta
- Egy másik űrlapon egy név/jelszó (legyen mindig admin/margit) megadása mellett legyen lehetőség egy valuta árfolyamának a módosítására
- Minden árfolyammódosítást követően egy táblázatban jelenjenek meg a valuták és az aktuális átváltási árfolyamok

Házi feladat

Készítsen egy egyszerű üzenetküldő alkalmazást az alábbi funkciókkal:

- A szervlet felhasználók és üzenetek nélküli alapállapotban indul
- A kezdőképernyőn legyen lehetőség új felhasználók számára a regisztrációra név/jelszó/megismételt jelszó megadásával
- A már regisztrált felhasználók név/jelszó megadásával beléphetnek
- Belépést követően a felhasználó előtt jelenjenek meg az alábbi funkciók:
  - Jelszó megváltoztatása – szokásos módon (régi/új/ismételt jelszó)
  - Üzenet küldése – egy tárgy és a többsoros üzenet megadása mellett egy listából választhassa ki a regisztált felhasználók közül a címzette(ke)t
  - Üzenetek olvasása – A megjelenő oldalon egy táblázatban jelenjen meg a felhasználónak küldött utolsó 10 üzenet:
    - időpont
    - küldő felhasználó, címzettek
    - tárgy, üzenet

    A megjelenő táblázatban legyen lehetőség bármelyik névre kattintva az illető számára üzenetet küldeni
  - Kilépés – érvénytelenítse a menetet
- A szervlet elkészítése előtt gondosan tervezze meg az „üzleti logikát” alkotó osztályhierarchiát (Pl. Személy, Üzenet, Üzenettároló osztályok)

Ajánlott irodalom

Az óra anyagához kapcsolódó irodalom

- Nyékyné Gaizler Judit: Java 2 útikalauz programozóknak 1.3 II.; ELTE TTK Hallgatói alapítvány, Budapest

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

  218 – 249. 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