Java servlet technológia

Áttekintés

- Bevezetés
- Servlet map-elés web.xml-ben
- Szesszió követés
- include, forward
- Szűrők

Servlet-et használni érdemes, ha

- a kimenet típusa bináris (pl. egy kép)
- a servlet-nek nincs közvetlen kimenete (csak átirányít egy másik komponensre)
- a megjelenítés nagyon változó lehet

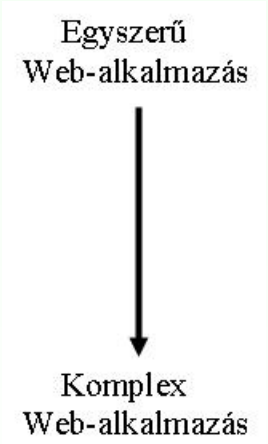
JSP-t használni érdemes, ha

- a kimenet nagyrészt szöveg alapú (pl. HTML, XML)
- a kimenet formátuma többnyire kötött

Servlet-JSP kombináció (MVC) szükséges, ha

- a kérés többféle, kinézetben lényegesen különböző nézetet eredményezhet
- egy nagyobb csapat dolgozik a Web-alkalmazáson (egyesek az üzleti logika-, mások a Web-es felület fejlesztésével foglalkoznak)
- bonyolult adatfeldolgozásra van szükség, ugyanakkor a megjelenítés viszonylag kötött

Web–alkalmazások



- script elemek használata – közvetlenül (JSP-ben deklarált és hivatkozott) beágyazott Java kód
- script elemek használata – közvetetten (segédosztályokban deklarált, JSP-ből hivatkozott) beágyazott Java kód
- beanek használata
- servlet/JSP kombináció (MVC elv)
- MVC + kifejezés nyelv (EL – expression language) használata a JSP-ben
- JSP elemkönyvtárak (custom tag library) használata
- MVC + bean-ek + elemkönyvtárak + keretrendszerek (pl. Struts, JSF)

Servlet

A servlet:

- egy java osztály, mely kérés-válasz (request-response) modellre épül
- leginkább web kérések kiszolgálására használják őket
- a java servlet technológia HTTP-specifikus servlet osztályokat is tartalmaz
- a **javax.servlet** és a **javax.servlet.http** csomagok segítségével írhatunk servleteket
- mindegyik servlet a **Servlet** interfészt kell implementálja

A servlet életciklusa

Egy servlet életciklusát a web-konténer kezeli, melybe az illető servlet telepítve volt.

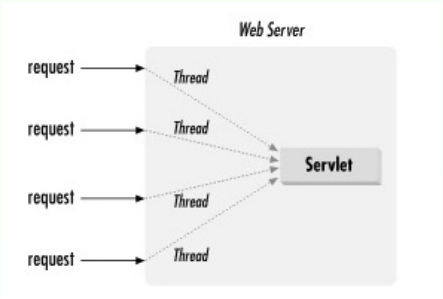
Ha egy kérés érkezik a servlet-hez, a következők történnek:

1. Ha a servlet-nek még nem létezik példánya (instanciája), akkor a web-konténer
   - betölti a servlet osztályt
   - létrehoz egy instanciát, majd inicializálja az **init** metódus segítségével

   Az **init** metódus rendszerint konfigurációbelovasásra, erőforrás inicializálásra (pl. adatbázishozzáférés) használható vagy bármilyen egyszeri művelet elvégzésére.
2. Meghívja a **service** metódusát, átadva neki a kérés és válasz (request, response) objektumokat.
3. Ha a konténer el kell távolítsa a servletet, meghívja a **destroy** metódusát
   (az init párja: erőforrások felszabadítása stb.)

init, destroy, service

Az **init** és **destroy** egyszer hívódik meg, a **service** pedig minden egyes kérésre.



Több szál, de (általában) ugyanaz a servlet-instancia szolgálja ki a különböző kéréseket

Információmegosztás

A web-komponensek –akárcsak a legtöbb objektum– más objektumokkal együttműködve végzik el feladatukat.
Ez a következőképpen történhet:

- segédosztályok segítségével
- nyilvános hatókörű (public scope) objektumok attribútumait oszthatják meg
- más web-komponenshez továbbítanak

Nyilvános hatókörű objektumok (public scope objects)

A web komponensek négy (nyilvános hatókörű) objektum attribútumain keresztül oszthatnak meg információt.
Az attribútumok a
[get—set]Attribute metódusokon keresztül érhetők el

Scope Objektum	**Osztály**	**Elérhető**
Web kontextus	`ServletContext`	a web-alkalmazáson belüli web-komponensekből
Szesszió	`HttpSession`	web-komponensekből, amelyek ugyanazon szesszión belüli kéréseket dolgozzák fel
Kérés	`HttpServletRequest`	az adott kérést kezelő web-komponensekből
(Oldal)	`JspContext`	a JSP-ből, amely létrehozza

Service metódusok írása

Service metódus írása (HTTPServlet objektum esetén):

– egy **do**<`Metódusnév`> fölülírásában (overriding) nyilvánul meg.

A <**Metódusnév**> `lehet:`

Get , Delete , Options , Post , Put, Trace

Egy ilyen metódus –

- a kérés (request) objektumból kinyeri az információkat,
- eléri a külső erőforrásokat,
- beállítja a válasz (response) objektumot ezen információk alapján.

A válasz objektumot úgy állítja be, hogy

- először lekér tőle egy **stream**-et (`getOutputStream` vagy `getWriter` metódus segítségével)
- feltölti azt a
  - válasz fejlécekkel
  - törzs (body) tartalommal

Információ kinyerése a kérés objektumból

A kérés objektum–

- azokat az adatokat tartalmazza, melyeket a kliens (böngésző) küldött a szerver felé HTTP protokollon keresztül.
- a **ServletRequest** interfészt implementálja

Ez az interfész a következő információk elérését szolgáló metódusokat tartalmazza:

- **Paraméterek elérése**: tipikusan a kliens által (a HTML form keretében) küldött információk
  Pl. `String id =request.getParameter("bookID");`
  (lásd. ParameterSnoop) Egy **input stream**-et is lekérhetünk a kérés objektumból és a tartalmát manuálisan feldogozhatjuk.
  Karakter stream lekérésére a `getReader`-t használhatjuk,
  bináris adatokhoz (pl. file upload) pedig a `getInputStream`-et.

· · · ▶

Információ kinyerése a kérés objektumból

A kérés objektum–

- azokat az adatokat tartalmazza, melyeket a kliens (böngésző) küldött a szerver felé HTTP protokollon keresztül.
- a **ServletRequest** interfészt implementálja

Ez az interfész a következő információk elérését szolgáló metódusokat tartalmazza:

- **Objektum attribútumok**: tipikusan egy servlet által létrehozott és a kérés objektumba betett objektumok, melyek így más servlet-ekben is elérhetők lesznek (forward és include).
- Információk a használt protokollról, a kliensről (lásd. HeaderSnoop) valamint a szerverről (lásd. ServerSnoop)

Hozzáférés web-kontextushoz

A web-kontextusnak megfelelő objektum egy `ServletContext` interfészt implementáló objektum.
A servlet `getServletContext` metódusával lehet megkapni.

A `ServletContext`-en keresztül többek között az alábbiak érhetők el:

- Inicializáló paraméterek
- Objektum attribútumok
- Naplózás (logging)

Kérés (request) URL

Egy HTTP kérés URL a következő részekből áll:

`http://[host ]:[port ][request path ]?[query string ]`

A `request path` (kérés útvonala) a következő részekre bontható tovább:

- **Kontextus út** (context path): slash ('/') és a servlet-et tartalmazó web-alkalmazás kontextus gyökere (context root) – a web-alkalmazás neve
- **Servlet út** (servlet path): slash ('/') és a komponenst aktiváló kérésnek megfelelő map-elés

Paraméterek (query string)

A query string összetevői:

- paraméterek
- a nekik megfelelő értékek

Az egyes paramétereket a kérés objektumból a `getParameter` metódussal nyerjük ki.

Kétféleképpen lehet query string-et generálni:

- Egy query string explicit módon megjelenik az URL-ben
  Pl. `<a href="/servletPath?param1=1">Text</a>`.
  A paraméter a következőképpen kapható meg:
  `String parameter =request.getParameter("param1");`
- A query string hozzáadódik az URL-hez, amikor egy HTML form elküldése (submit) a HTTP GET metódussal történik.
  Megj.: HTTP POST metódus esetén a parametérek a kérés törzsében (body) helyezkednek el.

Servlet map-elések megadása a web.xml-ben

- A Servlet-et deklarálni kell:
  - logikai nevet kell neki adni,
  - meg kell adni az osztályt, amelyik implementálja
  - esetleg inicializáló paramétereket adhatunk meg neki

```
<servlet>
   <servlet-name>helloWorld</servlet-name>
   <servlet-class>hello.HelloWorldEx</servlet-class>
   <init-param>
      <param-name>
         initial
      </param-name>
      <param-value>
         10
      </param-value>
   </init-param>
</servlet>
```

a Servlet init paramétereihez való hozzáférés

lásd InitSnoop

- A Servlet-et hozzá kell rendelni (map-elni) egy vagy több web-erőforráshoz vagy **URL mintához**

```
<servlet-mapping>
   <servlet-name>helloWorld</servlet-name>
   <url-pattern>
     /servlet/HelloWorldExample
   </url-pattern>
</servlet-mapping>
```

Megj.: másik alternatíva magyarázó jegyzet (annotation) használata a Servlet osztály definiálásakor: **@WebServlet**

Kliensállapot megőrzése

Sok alkalmazás esetében szükség van arra, hogy az azonos felhasználótól jövő kérések össze legyenek kapcsolva egymással. Pl. bevásárlókosár
A web-alkalmazások felelősek ennek a megvalosításáért, mivel a HTTP protokoll állapot nélküli (stateless).
lásd ShoppingCart

A Java servlet technológia egy API-t kínál a **szesszió** kezelésére.

- A szessziót egy `HttpSession` objektum képviseli.
- Lekérhető a kérés (request) objektumtól a `getSession` metódussal. Ez visszaadja az aktuális szessziót vagy ha még nincs, akkor létrehoz egyet.
- A szesszióhoz objektum-alapú attribútumokat lehet rendelni. Ezek egy adott web-kontextuson belül bármelyik web-komponensből hozzáférhetőek, melyek ugyanahhoz a szesszióhoz tartozó kéréseket dolgozzák fel.

A szesszióhoz hozzárendelt objektumok értesítése

A szesszióhoz hozzárendelt objektumok értesítése

Az alkalmazás értesítheti a web-kontextushoz valamint a szesszióhoz rendelt objektumokat bizonyos események bekövetkeztekor:

- Amikor egy objektum hozzáadódik vagy eltávolítódik a szesszióból.
  → Hogy ezt az értesítést megkapja az objektum a `HttpSessionBindingListener` interfészt kell implementálja.
- Amikor a szesszió, amelyhez az objektum hozzá van rendelve passziválva vagy aktiválva (perszisztensen lementve majd visszatöltve) van.
  → Hogy ezt az értesítést megkapja az objektum a `HttpSessionActivationListener` interfészt kell implementálja.

Szesszió követés

Szesszió követés

Egy web-konténer különbözőféleképpen rendelhet egy szessziót egy felhasználóhoz, viszont bárhogyan is történjen ez, azzal jár, hogy egy azonosító küldődik a kliens és szerver között.
Ez az azonosító eltárolható

- egy sütiben (cookie)
- minden egyes URL-ben, amit a kliens megkap

Ha az alkalmazás szessziót használ, akkor biztosítani kell azt, hogy a szesszió követés működik a sütik kikapcsolása esetében is. (lásd SessionSnoop)

- Ezt az URL-átírással valosíthatjuk meg a válaszobjektum `encodeURL(URL)` metódus-hívásával minden egyes URL-re, amit a servlet visszaad.
- Ez a metódus hozzáfűzi a szesszió ID-t az URL-hez, ha a sütik ki vannak kapcsolva.

Más web-erőforrás hívása

Direkt vagy indirekt módon történik.

- Indirekt módon akkor, ha a web-komponens a válaszban tartalmaz egy URL-t, amelyik egy másik web-komponensre mutat.
- Direkt módon kétféleképpen:
  - egy web-komponens magábafoglalhatja egy másik web-komponens tartalmát (include)
  - továbbíthatja a kérést egy másik komponenshez (forward)

Ahhoz, hogy elérjünk egy erőforrást, amelyik egy web-komponenst futtat, először egy `RequestDispatcher` objektumot kell lekérjünk a `getRequestDispatcher(URL)` metódussal.

A `RequestDispatcher` objektumot két módon lehet lekérni:

- a kérés objektumtól
- a webkontextus objektumtól

- A kérés objektumból lekért `RequestDispatcher` esetében az URL lehet relatív (nem /-el kezdődő),
- A web-kontextustól lekért esetében viszont az URL abszolút kell legyen.
- Ha az erőforrás nem elérhető, null-t kapunk vissza.

Más erőforrások beszúrása a válasz objektumba

Sokszor hasznos lehet, hogy egy web erőforrást beszúrjunk egy másikba pl. jogvédelmi információkat (copyright) → Ehhez a `RequestDispatcher include(request, response)` metódusát használhatjuk. lásd pl. MainPage

Megszorítások a válasz objektum tekintetében:

A beszúrt web-komponens írhat ugyan a response tartalmába (body), de

- nem állíthatja a fejléceket
- nem hívhat olyan metódust, ami a válasz objektum fejlécét érinti. (pl. `addCookie`).

ami ilyenkor történik:

- a kérés objektum el lesz küldve a beszúrt komponensnek
- a beszúrt web-komponens elvégződik
- majd a keletkezett tartalom beszúródik a külső servlet által generált válasz objektumba

Kérés továbbítása egy másik web-komponenshez

Sok web-alkalmazásban van egy web-komponens, mely egy előfeldogozást végez és ettől függően továbbít egy másik komponenshez, amely a választ generálja (lásd **MVC**, **Struts**).

Ahhoz, hogy egy kérést egy másik web-komponenshez továbbítsuk a `RequestDispatcher forward` metódusát használjuk.
lásd pl. SearchLogic

Megszorítások:

Ha a `ServletOutputStream` vagy a `PrintWriter` objektumokat módosítottuk a továbbítás előtt, akkor a továbbításkor `IllegalStateException` hibát kapunk.

Kérés/válasz szűrése (filtering)

A szűrő–

- módosíthatja a kérés és válasz objektumok tartalmát
- ez nem web-komponens abban az értelemben, hogy nem hoz létre választ (response), csak módosítja azt
- egy funkcionalitást ad, amely hozzárendelhető a web-komponenshez
- nem függ a web-erőforrástól, amihez hozzá van rendelve

Főbb alkalmazási területei:

- egy másik weboldalra irányít át, valamilyen feltétel függvényében (pl. annak ellenőrzése, hogy be van-e jelentkezve a felhasználó – lásd: VerifyLogonFilter)
- módosítja a kérés vagy válasz fejlécét vagy adatait (kibővített kérés és válasz osztályok segítségével),
- külső erőforrásokkal kommunikálhat

A gyakorlatban:

- azonosítás
- naplózás (logging)
- adatsürítés
- titkosítás
- XML transzformáció stb.

Egy web-erőforrás esetében bekonfigurálható, hogy nulla, egy vagy több szűrő legyen rá alkalmazva a megfelelő sorrendben. (lásd InitCounter → Filter1, Filter2; Hello → SimpleFilter)

A szűrők használata három részből áll:

- meg kell írni a szűrőt
- meg kell írni a kibővített kérés és válasz osztályokat
- a telepítéskor mindegyik web-erőforrásnak meg kell adni a kívánt szűrő-láncot

Szűrő megírása

A szűrő API a következő főbb interfészekből áll:

Filter, FilterChain, es FilterConfig

Egy szűrő definiálásához a `Filter` interfészt kell implementálni.

A `doFilter` metódus–

- paraméterként kapja a kérés, válasz valamint a szűrőlánc objektumokat
- létrehozza a kibővített kérés és/vagy válasz objektumokat
- meghívja a `doFilter`-t (a további szűrőkre a láncból) paraméterként a kibővített objektumokat adva meg,
- akár blokkolhatja is a kérést úgy, hogy nem hívja meg a következő szűrőt, de akkor ő a felelős a válasz objektum feltöltéséért
- a visszakapott kibővített objektumokkal módosíthatja a kérés valamint válasz objektumokat

A `doFilter`-en kívül még az `init` és `destroy` metódusokat is lehet implementálni.

- Az `init` akkor hívódik, mikor a konténer példányosítja a szűrőt.
- A paraméteként megadott `FilterConfig`-ban megkapjuk az inicializáló paramétereket.

- A kérés kibővítéséhez a `HttpServletRequestWrapper` osztályt kell kibővíteni,
- a válasz kibővítéséhez a `HttpServletResponseWrapper` osztályt

Szűrő hozzárendelések (map-elések) megadása

A web-konténer a szűrő hozzárendelések alapján alkalmazza a szűrőket az egyes web-erőforrásokra.
Egy szűrő map-elés hozzárendel

- egy szűrőt egy web-komponenshez egy név alapján
- egy szűrőt web-erőforrásokhoz URL minták (pattern) szerint

A szűrők olyan sorrendben lesznek meghívva, amilyen sorrendben a szűrő hozzárendelésben megjelennek.

A telepítésleíróban (deployment descriptor):

- Deklarálni kell a szűrőt:
  - nevet kell neki adni,
  - meg kell adni az osztályt, amelyik implementálja
  - inicializáló paramétereket lehet adni neki

Pl.

```
<filter>
   <filter-name>Servlet Mapped Filter</filter-name>
   <filter-class>filters.ExampleFilter</filter-class>
   <init-param>
      <param-name>
         name
      </param-name>
      <param-value>
         value
      </param-value>
   </init-param>
</filter>
```


A telepítésleíróban (deployment descriptor):

- Map-elni kell a szűrőt egy web-erőforráshoz vagy egy URL mintához

Pl.

```
<filter-mapping>
   <filter-name>Servlet Mapped Filter</filter-name>
   <servlet-name>invoker</servlet-name>
</filter-mapping>
<filter-mapping>
   <filter-name>Path Mapped Filter</filter-name>
   <url-pattern>/servlet/*</url-pattern>
</filter-mapping>
```