Java szervlet technológia

Igény a **dinamikus** tartalomra...

Az első ilyen technológiák:

- kliens oldalon:
  - appletek
- szerver oldali technológiák:
  - CGI, JSP, PHP, ASP

a CGI-ről

CGI (Common Gateway Interface):

- először ezt használták szerveroldali dinamikus tartalom generálására
- a CGI szabvány lehetővé tette a kiszolgálónak, hogy külső programmal kommunikáljon, melyet a webszerver egy megfelelően konfigurált könyvtárába (pl. **cgi-bin**) kell bemásolni
- minden egyéb – hálózatkezeléssel kapcsolatos– problémát a webszerver kezel
- ez a külső program kommunikálhat bármilyen külső erőforrással (pl. adatbázis) és dinamikus tartlamat hoz létre.
- annakidején eléggé elterjedt

Hátrányok

A CGI hátrányai:

- platformfüggőség
- többszöros kérések kiszolgálásához való alkalmazkodás (scalability) hiánya
  (minden új kérés egy operációs rendszer szintű folyamat elindítását eredményezte)
- a HTML elemek a programozási nyelv kódjába vannak beágyazva
  (a programozó és grafikus felüet tervező (designer) munkáját nehéz szétválasztani)

Ezen hátrányok megszüntetésének érdekében kifejlesztették a **Java szervlet** és **JSP** technológiát.

A szervlet:

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

A szervlet életciklusa

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

Ha egy kérés érkezik a szervlethez, a következők történnek:

1. Ha a szervletnek még nem létezik példánya (instanciája), akkor a web-konténer
   - betölti a szervlet 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 szervletet, meghívja a **destroy** metódusát
   (az init párja: erőforrások felszabadítása stb.)

A szervlet életciklusa

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

Ha egy kérés érkezik a szervlethez, a következők történnek:

1. Ha a szervletnek még nem létezik példánya (instanciája), akkor a web-konténer
   - betölti a szervlet 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.

init, destroy, service

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

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

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

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 egy szesszión belüli kérésekhez tartoznak

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 egy szesszión belüli kérésekhez tartoznak
Kérés	`HttpServletRequest`	az adott kérést kezelő web-komponensekből

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 egy szesszión belüli kérésekhez tartoznak
Kérés	`HttpServletRequest`	az adott kérést kezelő web-komponensekből
(Oldal)	`JspContext`	a JSP-ből, amely létrehozza

Service metódusok írása

Service metódus írása:

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

A <**Metódusnév**> lehet:

Get , Delete , Options , Post , Put

Egy ilyen metódus –

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

Service metódusok írása

Service metódus írása:

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

A <Metódusnév> lehet:

Get , Delete , Options , Post , Put

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 **output stream**-et
- feltölti azt a
  - válasz fejlécekkel
  - test (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 tartalmaz:

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

- **Objektum attribútumok**: tipikusan a szervlet konténer által biztosított objektumok, illetve egy szervlet által létrehozott és a kérés objektumba betett objektumok, melyek így más szervletekben is elérhetők lesznek (forward és include).
- Információk a használt protokollról, a kliensről valamint a szerverről
- **Lokalizáció**val kapcsolatos információk

Kérés (request) URL

Egy HTTP kérés URL a következő részekből áll:

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

A `request path` a következő részekre bontható tovább:

- **Kontextus út** (context path): slash ('/') és a szervletet tartalmazó web-alkalmazás kontextus gyökere (context root)
- **Szervlet 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 testében (body) helyezkednek el.

Szervlet map-elések megadása a web.xml-ben

- A szervletet 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 szervletet 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>
```

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)
- módosítja a kérés vagy válasz fejlécét vagy adatait (kibővített kérés és válasz osztályok segítségével),
- külső erőforrásokkal kommunikálhat

A gyakorlatban:

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

Egy web-erőforrás esetében bekonfigurálható, hogy nulla, egy vagy több szűrő legyen rá alkalmazva a megfelelő sorrendben.

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 objetumokat 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 objetkumokkal módosíthatja a kérés valamint válasz objetumokat

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

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

Kibővített kérés és válasz osztályok

A szűrő sokféleképpen módosíthatja a kérés és válasz objetkumokat...
PI.

- megadhat egy attribútumot a kérésbe vagy adatot szúrhat be a válaszba

→ Ez utóbbihoz a szűrő el kell kapja a választ még mielőtt az el lenne küldve a kliens felé.

→ Ehhez a szervletnek, amely a választ generálja egy `stand-in` adatfolyamot (stream) kell átadni.
Egy ilyen adatfolyam megakadályozza a szervletet abban, hogy lezárja az eredeti válasz-objektumot mikor végzett, és így megengedi a szűrőnek, hogy utólagosan módosítsa a választ.

→ Ehhez a szűrő létrehoz egy kibővített választ (wrapper), aminek átírja a `getWriter` objektumát, hogy ezt a `stand-in` adatfolyamot adja vissza.

→ Ezzel a kibővített válasszal hívja meg a `doFilter`-t

- 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>
```

A telepítésleíróban (deployment descriptor):

- Korlátozni lehet, hogy mikor legyen a szűrő alkalmazva:
  - REQUEST : amikor a kérés direkt a klienstől jön
  - FORWARD : ha a kérés továbbítva (forward) lett a komponenshez
  - INCLUDE : ha a kérést egy beágyazott komponens kapja
  - ERROR : a kérés az error lappal van feldolgozva

Egy web-erőforráshoz akárhány különböző szűrő rendelhető és ugyanaz a szűrő több web-erőforráshoz is hozzárendelhető.

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.

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. `setCookie`).

ami ilyenkor történik:

- a kérés el lesz küldve
- a beszúrt web-komponens elvégződik
- majd a keletkezett tartalom beszúródik a külső szervlet 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 később **MVC**, **Struts**).

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

Megszorítások:

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

Hozzáférés web-kontextushoz

A kontextus, amelyben a web-komponensek elvégződnek egy `ServletContext` interfészt implementáló objektum.
Ezt meg lehet kapni a szervlet `getServletContext` metódusával.

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

- Inicializáló paraméterek
- A web-kontextushoz rendelt erőforrások
- Objektum attribútumok
- Naplóbeállítások (logging)

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).

A Java szervlet 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.

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

A web-konténer több metódust használhat arra, hogy egy felhasználóhoz egy szessziót rendeljen, ami 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.

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