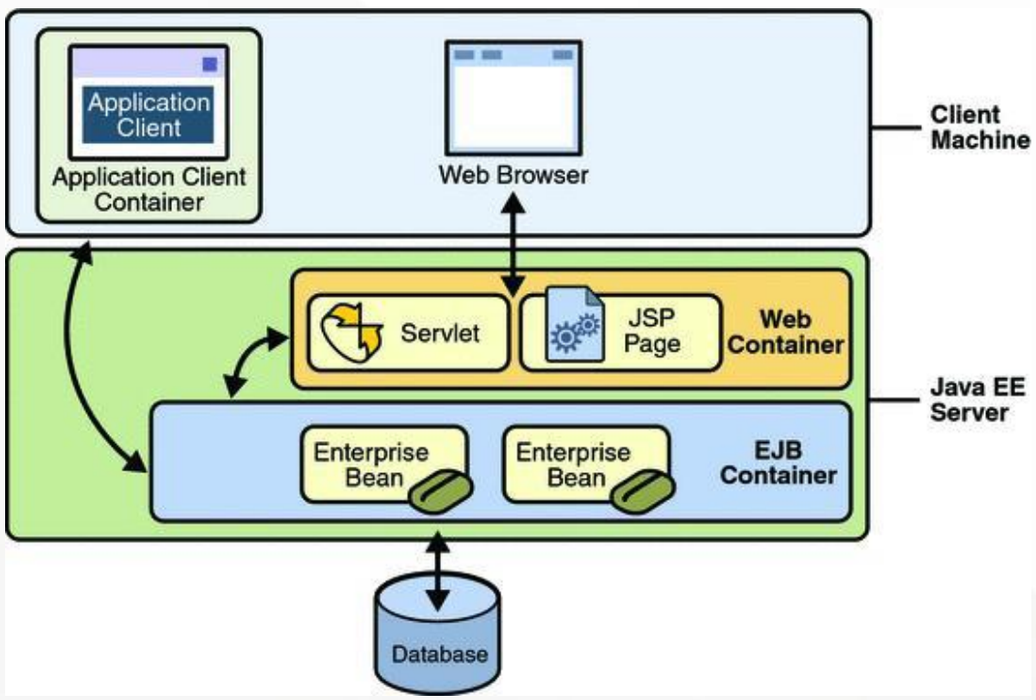
Java web-alkalmazások 1.

A Java Servlet API

Simon Károly
simon.karoly@codespring.ro

Webes alkalmazások

- Kliens réteg: a kliens gépén futó alkalmazások.
- Alkalmazás réteg:
  - Web réteg: a Java EE szerveren futó webes komponensek, a szerver web-konténerében futnak (Servlet, JSP).
  - Üzleti logika réteg: a Java EE szerveren futó üzleti logikát megvalósító komponensek, a szerver EJB konténerében futnak (EJB).
- Adathozzáférési réteg: Enterprise Information System (EIS) szerver



Java web-alkalmazás részei

- A web-kliens a szervertől érkező oldalakat mutatja meg (HTML, XML stb. alapú dinamikus weboldalak, amelyeket a szerveroldali komponensek generálnak) (böngésző/browser).
- Java EE komponensek (szerver oldal): önálló funkcionalitásokat biztosító, egymással kommunikáló szoftverkomponensek. Java osztályok és erőforrás állományok, amelyek összerakásuk (assembly) után egy alkalmazásszerverre lesznek telepítve (deployment). Meg kell felelniük a vonatkozó Java EE specifikációnak.
  - Web komponensek (Servlet, JSP): a szerver web-konténerében (pl. Tomcat) futnak.
    - Servlet: kéréseket dinamikusan feldolgozó, azokra dinamikusan választ generáló Java osztályok.
    - JSP: a tartalom létrehozására, szöveg-alapú dokumentumok generálására szolgáló komponensek (a háttérben tulajdonképpen servlet-ként futnak).
  - EJB (Enterprise JavaBeans): szerveroldali, üzleti logikáért felelős komponensek (session beans, message-driven beans), az alkalmazásszerver EJB konténerében futnak
- Összeállítás: különböző konténer beállítások, konfigurációs állományokban → a szerver ezek alapján biztosítja szolgáltatásait (tranzakció-kezelés, biztonság stb.).

Web-alkalmazás működése

- A web-kliens egy http kérést (request) küld a szervernek.
- A web-konténer a kérést egy HttpServletRequest objektumba alakítja.
- Az objektumot megkapja a megfelelő web-komponens (Servlet/JSP).
- A web-komponens az EJB-ekkel (vagy más üzleti logikát megvalósító komponensekkel) együttműködve elvégzi a megfelelő műveleteket.
- A web-komponens továbbíthatja a kérést más web-komponensekhez (forward).
- A válasz felépítésekor a web-komponens dinamikus tartalmat generál, egy HttpServletResponse típusú objektumot épít fel.
- Az objektumot a web-szerver http válasszá (response) alakítja, és visszaküldi a kliensnek.

Web-alkalmazás

- Web-alkalmazás részei:
  - Web-komponensek
  - Telepítés-leíró (deployment descriptor)
  - Java osztályok és jar csomagok
  - Statikus erőforrások (képek, statikus html oldalak stb.)
- Szerverfüggetlenség: bármilyen konténerbe telepíthető, ha az megfelel a Java Servlet/JSP specifikációnak
- Létrehozás és futtatás:
  - Web-komponensek és segédosztályok implementációja.
  - Telepítés leíró létrehozása.
  - Az osztályok lefordítása és a web-alkalmazás létrehozása, építő (build) eszköz (pl. Ant, Maven) segítségével.
  - Az alkalmazás telepíthető egységbe (deployable unit) történő csomagolása (.war állomány) – opcionális.
  - Az alkalmazás telepítése a web-konténerbe.
  - Az alkalmazásra hivatkozó URL meghívása a böngészőből.

Web-modul

- A web-konténer megfelelő katalógusába lesz telepítve: egy könyvtár, vagy egy .war állomány.
- A gyökérben találhatóak a JSP oldalak és statikus erőforrások.
- A gyökér tartalmaz egy WEB-INF katalógust, melynek tartalma:
  - web.xml – a telepítés-leíró.
  - classes könyvtár – szerver oldali osztályok (Servlet-ek és segédosztályok).
  - lib könyvtár – csomagok, amelyeket a szerver oldali komponensek használnak.

Servlet

- Java osztály, megvalósítja a Servlet interfészt.
- Kérés-válasz (request-response) modellre épül, leginkább web-kérések kiszolgálására használjuk.
- A Servlet technológia http-specifikus osztályokat (is) biztosít.
- javax.servlet és javax.servlet.http csomagok
- Servlet életciklusa (a web-konténer kezeli):
  - Ha egy kérés érkezik a Servlet-hez, és annak még nem létezik példánya, akkor a web-konténer betölti a servlet osztályt, létrehoz egy példányt és inicializálja (init metódus).
  - Az init metódus egyszeri műveletek elvégzésére használható: konfiguráció/erőforrás beolvasása stb.
  - A web-konténer meghívja a servlet service metódusát, átadva neki a kérés és válasz (request és response) objektumokat.
  - A servlet eltávolításakor meghívja annak destroy metódusát, mely az erőforrások felszabadítására alkalmas.
  - Míg az init és a destroy csak egyszer, a service minden kéréskor meghívásra kerül.

Információmegosztás

- A web-komponensek kommunikációja történhet segédosztályok segítségével, a komponensek továbbíthatnak kéréseket más komponensekhez (forward) és megoszthatják nyilvános hatókörű (public scope) objektumok attribútumait (get/set metódusok segítségével).
- Nyilvános hatókörű objektumok:
  - Web környezet (kontextus) – ServletContext: a web-alkalmazáson belüli web-komponensekből érhető el.
  - Munkamenet - Session/HttpSession: egy adott munkameneten belüli kéréseket feldolgozó web-komponensekből érhető el.
  - Kérés – ServletRequest/HttpServletRequest: az adott kérést kezelő web-komponensekből érhető el.
  - Oldal környezete – JspContext: az oldalt létrehozó JSP-ből érhető el.

Szolgáltatás metódusok

- Szolgáltatás (service) metódusok megvalósításához doMethodName alakú metódusokat kell implementálnunk.
- doGet, doPost, doPut, doDelete, doOptions
- A metódusok a kérés objektumból kinyerik az információt, külső erőforrásokat érnek el és felépítik a válaszobjektumot.
- A válaszobjektum felépítésekor: a metódus kér egy adatfolyamot (a getOutputStream, getWriter metódusok segítségével) és feltölti azt a fejlécekkel, illetve a tartalommal (törzs/body).

Kérés objektum

- A ServletRequest interfészt valósítja meg, a kliens által a szerver felé küldött (a http protokollnak megfelelően) adatokat tartalmazza.
- ServletRequest interfész:
  - A paraméterek elérésére szolgáló metódusok.
    - A paraméterek a kliens által küldött adatok (form kitöltése).
    - Pl. String name = request.getParameter("personName");
    - A getInputStream (bináris adatok esetén, pl. állomány feltöltése) vagy getReader metódusok segítségével a kérésnek megfelelő bemeneti adatfolyamot is lekérhetjük, manuálisan feldolgozva azt.
  - Objektum attribútumok lekérdezésére
    - Servlet által létrehozott kérés objektumba behelyezett objektumok attribútumainak lekérdezése, pl. include/forward műveletek esetén.
  - Információk lekérdezése a protokollról, kliensről, szerverről.
- Kérés URL: **`http://[host]:[port]/[request path]?[query string]`**
  - Kérés útvonala (request path): a web-alkalmazás neve (a servlet-et tartalmazó alkalmazás gyökere) és (/-el elválasztva) a servlet elérési útvonala (a komponenst aktiváló kérésnek megfelelő map-elés, lásd web.xml).

Paraméterek és környezet

- Paraméterek:
  - Query string: a paraméterek neveit és a nekik megfelelő értékeket tartalmazza. A kérés objektumból a paraméterek a getParameter metódus segítségével kérdezhetőek le.
  - A query string megjelenhet az URL-ban:

    Pl.
    ```
    <a href="/servletPath?param1=1">Text</a>
    String parameter = request.getParameter("param1");
    ```
  - Ha egy html form elküldése (submit) a http GET metódussal történik, a query string hozzáadódik az URL-hoz.
  - Http POST metódus esetében a paraméterek a kérés törzsében (body) helyezkednek el.
- Környezet:
  - A web-kontextus egy ServletContext interfészt megvalósító osztály példánya, a getServletContext metódus segítségével kérhetünk rámutató referenciát.
  - Segítségével többek között elérhetőek az inicializáló paraméterek, az objektum attribútumok, és megvalósítható a naplózás.

web.xml

- A Servletek esetében meg kell adnunk a Servlet nevét, a megvalósító osztályt és megadhatunk inicializáló paramétereket:

```
<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-et hozzá kell rendelni egy (vagy több) web erőforráshoz, URL minták segítségével:

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

Munkamenetek követése

- Szükséges lehet, hogy az azonos felhasználótól érkező kéréseket összekapcsoljuk egymással (pl. bevásárlókosár) és, mivel a http protokoll állapot nélküli (stateless), a web-alkalmazások felelősek ennek megvalósításáért.
- A Java Servlet technológiának megfelelően a munkamenetet egy HttpSession objektum reprezentálja, amely a kérés objektumtól lekérhető a getSession metódussal. A metódus visszaadja az aktuális munkamenet-objektumra mutató referenciát és, ha még nem volt létrehozva az objektum, létrehozza azt.
- A munkamenethez tartozó attribútumok bármelyik web-komponens által lekérdezhetőek, amennyiben a komponens az adott munkamenethez tartozó kéréseket dolgozza fel.
- Az alkalmazás értesítheti a munkamenethez rendelt objektumokat bizonyos eseményekről:
  - Egy objektum hozzáadódik a munkamenethez vagy eltávolítódik a munkamentből (a figyelőknek a HttpSessionBindingListener interfészt kell megvalósítaniuk).
  - Az objektumhoz hozzárendelt munkamenet aktiválása vagy inaktiválása (HttpSessionActivationListener interfész).

Munkamenetek követése

- A munkamenet felhasználóhoz való hozzárendelése web-konténer specifikus.
- A kliens és a szerver között minden esetben egy azonosító lesz elküldve.
- Az azonosító eltárolható egy sütiben (cookie), vagy minden egyes URL-ben, amelyet a kliens megkap.
- Ha az alkalmazás munkameneteket használ, javasolt biztosítania, hogy a követés működni fog a sütik kikapcsolása esetén is.
  - Ezt az URL átírásával valósíthatjuk meg, az encodeURL(URL) metódus segítségével, meghívva azt minden URL-re, amit a servlet visszaad. A metódus hozzáfűzi a munkamenet azonosítóját (ID) az URL-hez, ha a sütik ki vannak kapcsolva.

Hozzáférés más erőforrásokhoz

- Közvetett, vagy közvetlen módon:
  - Indirekt hozzáférés: a web-komponens válasza tartalmaz egy URL-t, amely egy másik komponensre mutat.
  - Direkt hozzáférés: a web-komponens magába foglalhatja egy másik web-komponens tartalmát (include), vagy továbbíthat kérést egy másik komponenshez (forward).
- Egy erőforrás eléréséhez először egy RequestDispatcher objektumot kell kérnünk a getRequestDispatcher(URL) metódus segítségével, a kérés objektumtól, vagy a web-kontextustól. Az első esetben az URL lehet relatív útvonal, a második esetben abszolútnak kell lennie. Nem elérhető erőforrás esetén null értéket kapunk.

Hozzáférés más erőforrásokhoz

- Web-komponens beszúrásához a dispatcher include(request, response) metódusát használhatjuk.
  - A beszúrt komponens írhat a válasz tartalmába, de nem módosíthatja a fejléceket. A kérés el lesz küldve a beszúrt komponensnek, amely elvégzi műveleteit (a megkötések betartásával), majd a keletkezett tartalom be lesz szúrva a külső servlet által generált válaszobjektumba.
  - A mechanizmus több esetben hasznos lehet (pl. jogvédelmi információk beszúrása stb.).
- Kérések továbbításához a dispatcher forward metódusát használhatjuk.
  - A mechanizmus több esetben hasznos lehet, pl. amikor egy komponens elő-feldolgozást végez, majd az eredmény függvényében továbbít.
  - Ha a kimeneti adatfolyamokat módosítjuk (ServletOutputStream, PrintWriter) a továbbítás előtt, IllegalStateException típusú kivételt kapunk.

Kérések és válaszok szűrése

- A szűrő (filter) egy funkcionalitás, amely hozzárendelhető web-komponensekhez és módosíthatja a kérés/válasz objektumok tartalmát. Nem önálló web-komponens (nem generál választ, csak módosítja azt), és nem függ a hozzárendelt web-erőforrástól.
- Alkalmazási lehetőségek:
  - Jogosultságok ellenőrzése (be van-e jelentkezve a felhasználó), átirányítás más oldalakra (valamilyen feltétel függvényében).
  - Kérés/válasz fejlécének vagy adatainak módosítása (kibővített/testreszabott kérés és válasz osztályok alkalmazásával), kommunikáció külső erőforrásokkal.
  - Azonosítás, naplózás, tömörítés, titkosítás, transzformációk stb.
- Egy erőforráshoz több szűrő is hozzárendelhető, megadható a sorrend.
- Szűrő használatának lépései: szűrő megírása, kibővített kérés/válasz osztályok megírása (szükség esetén), a telepítéskor minden web-erőforrás esetében a hozzárendelt szűrőlánc megadása.

Filtering

- Fontosabb interfészek: Filter, FilterChain, FilterConfig.
- A szűrők a Filter interfészt valósítják meg.
- A doFilter metódus paraméterként kapja a kérés, válasz és szűrőlánc objektumokat. Létrehozza a kibővített kérés és válasz objektumokat (szükség esetén), majd meghívja a többi szűrő (lánc) doFilter metódusait, a kibővített objektumokat adva át paraméterként, vagy blokkolja a kérést, nem hívja meg a további metódusokat, de akkor ő felelős a válasz felépítéséért. A visszakapott kibővített objektumokkal módosíthatja a kérés és válasz objektumokat.
- A kérés kibővítéséhez a HttpServletRequestWrapper, a válasz kibővítéséhez a HttpServletResponseWrapper osztályokat kell kiterjeszteni.
- A doFilter metóduson kívül az init metódus is implementálható, ez akkor lesz meghívva, amikor a konténer példányosítja a szűrőt. Az inicializáló paramétereket a FilterConfig paraméter segítségével kaphatjuk meg.

Szűrők hozzárendelése

- Hozzárendelhetünk egy szűrőt egy web-komponenshez a név alapján, vagy web-erőforrásokhoz URL minták alapján. A szűrők a hozzárendelések sorrendjében lesznek meghívva.
- A telepítés-leíróban deklarálni kell a szűrőt: nevet kell adni neki, meg kell adni az osztályt, amely implementálja, inicializáló paramétereket lehet megadni:

```
<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 szűrőt hozzá kell kapcsolni egy web-erőforráshoz, vagy URL mintához:

```
<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>
```