JAVA nyelvű, alkalmazásszerveres, lekérdezést végző
web-alkalmazás készítésének főbb pontjai
(ha az adatbázis már elkészült)

A fejlesztést top-down módszerrel javasoljuk végigvinni.

Jsp-típusú web oldalak fejlesztése

1. Login weboldal készítése
   - Kinézet kialakítása.
   - Web forma elhelyezése; akció ebben az esetben egy servlet lesz. A servlet „neve" a servlet-et megvalósító osztály teljes neve (package névvel együtt), de a .class kiterjesztés nélkül. Ne felejtsük el, hogy olyan csomagnevet illik választani, mely a készítés helyére utal. Ezért pl. a labor-feladat esetén javasoljuk a "hu.bme.fsz.rinteg.control" csomagnevet (mind kisbetűvel) vagy ehhez hasonlót használni. Osztálynévnek megfelelő pl. a „JspController", amelyik konvencionálisan nagybetűvel kezdődik. Így a servlet teljes neve "hu.bme.fsz.rinteg.control.JspController" lesz és ezt kell az akció neveként is beírni.
   - Input mezők elnevezése és bellesztése.
   - Gondolni kell arra is, hogy hogyan történjék a hibás bejelentkezés visszajelzése. Alkalmazásszerver esetén a hiba-visszajelzést nagyon szépen meg tudjuk oldani ún. request hatókörű bean használatával. Ezt a bean-t természetesen az oldal tetején specifikálni kell, használatára pedig egy jól bevált módszer, hogy az input mező alá írjuk ki ebből a bean-ből a hibaszöveget (amit természetesen az alkalmazásszerver készített el az oldal elemzése során). Így hiba esetén ismét a beviteli oldalt kell kiadni, amin most már a hibás tartalmú beviteli mező alatt a hibajelzés is látszik, amit feltűnően – pl. piros színnel – ki lehet írni.

2. Listázási paramétereket kiválasztó weboldal fejlesztése
   - Kinézet kialakítása.
   - Web forma elhelyezése, akció ebben az esetben is egy servlet lesz. Célszerű azt csinálni, hogy az alkalmazás minden oldalán ugyanazt a servlet-et használjuk. Nyilvánvalóan ekkor meg kell különböztetni, hogy melyik oldalon járunk. Ehhez használjuk a "pageId" hidden-változót, melynek értékét a servlet-en belül lekérdezzük és szétugrunk a megfelelő helyekre.
   - Input mezők elnevezése és bellesztése.
   - A Login oldalról átveendő paraméterek miatt ún. session hatókörű bean-t kell alkalmazni. Ezt a bean-t is az oldal tetején specifikálni kell, egyébként használata teljesen megegyezik a request hatókörű bean-ével.
   - A beviteli hibák kijelzésére az előző oldal elemzésénél ismertetett request hatókörű bean-t célszerű használni. (Ez a bean a login oldal bean-jétől elvileg teljesen különböző, azonban lehet egyetlent bean-t is alkalmazni.). A kiválasztó oldalon tehát két bean-nel kell majd dolgozni!

3. Adatmegjelenítő weboldal fejlesztése
   - Kinézet kialakítása (célszerűen táblázatos formában).
   - Kiléptetés megoldása – vagy ciklikus visszaléptetés.
   - Az előzetesen bevett paraméterek átadására itt is session-bean-t kell használni. Az adatbázisból kinyert adatok kiírására Java-betéteket kell alkalmazni, melyeket <% és %> zárójelek közé kell tenni. A konkrét kiírásra vagy az „out" standard kimeneti változót használjuk a Java-scriplet-en belül, vagy megszakítjuk a Java betétet és "igazi" html kódot teszünk be.

Servlet-ek írása és bevizsgálása

- Az oldalakon specifikált servlet elkészítése Java nyelven történik. A servlet osztályt a "javax.servlet.HttpServlet" alaposztály kiterjesztésével kell létrehozni.
- A servlet lefordított formáját kell a Tomcat alkalmazásszerver (pontosabban servlet-container) megfelelő helyére bemásolni.
- Az Eclipse/Lomboz fejlesztői környezet teljesen automatizálja a Java compiler használatát, így a fordítás különösebb problémát nem fog okozni.

- A servlet fordítása mellett a további csomagbeli osztályokat is le kell fordítani. Javasoljuk, hogy két csomagot használjunk. A MVC tervezési minta vezérlés része – így a servlet, a session bean és a request bean(ek) – a "control" csomagba, a modell rész – azaz a kapcsolódó egyéb elemek, köztük az adatbázisok kezelése – a "model" csomagba kerüljenek.
- A lefordított osztályokat „deploy"-olni kell szerverünkre, ami egy Tomcat 4.x vagy 5.x típusú alkalmazásszerver (pontosabban servlet-container) lesz. Ez az akció szintén az Eclipse/Lomboz fejlesztői rendszerben könnyen megoldható.
- A deploy után a szervert el kell indítani. Ehhez is ad egyszerű eszközt az Eclipse/Lomboz fejlesztői rendszer. Technikailag ez azt jelenti, hogy a Tomcat szerverünk „webapps" alkönyvtárába az Eclipse-szel készített modulunk „.war" kiterjesztésű változata kerül, amit a szerver – újraindítása után – kicsomagol és közönséges alkalmazásként elindít. (A „.war" szerkezet valójában egy tömörített, zippelt adatstruktúra. Érdemes megnézni, hogy mi van benne.)
- A futtatás ***http://host:port/alkalmazasnev/login.jsp*** URL-lel indítható, ha a bejelentkezési oldalt a login.jsp weboldal reprezentálja.
- A servletek tesztelése – intelligens tesztrendszer híján – Tomcat konzollal történik az éles futás során. A konzolra a ***System.out.println("string");*** Java-utasításokkal mindent ki tudunk iratni.

Teljes rendszer bevizsgálása

- Oldalankénti futás tesztelése.
- A servlet tesztelésére az Eclipse debug-módú futtatást ajánl fel, ahol a program különböző helyeire töréspontokat lehet tenni. A megállás után a servlet változóit kényelmesen meg lehet vizsgálni.
- Hiba esetén olyan ún. tesztelő bean-eket is készíthetünk, amikkel csupán a teszt céljainak megfelelő kiírást végezzük a web oldalon!
- Naplózzuk az akciókat, amihez az igen egyszerű, nagyon elterjedt, ingyenesen használható log4j osztályt használjuk. Logok elemzése – ezek alapján a belső hibáinak megkeresése.

Ercsényi András