Debreceni Egyetem

Informatikai Kar

Informatikai rendszerek és hálózatok tanszék

A webes alkalmazásfejlesztés és a Java technológia kapcsolatának fejlődése

Témavezető:
Dr. Kuki Attila
Egyetemi Adjuktus

Készítette:
Csiki Gábor
Programtervező Matematikus

Debrecen

2010

Tartalomjegyzék

Bevezetés

Az internet megjelenésével egy új fejezet nyílt a számítástechnikában, hiszen előtte csak kisebb nagyobb helyi hálózatok léteztek, amelyek legfeljebb pár száz számítógépet foglaltak magukba. A világháló megjelenésével azonban egyszerre nagyon sok számítógép vált elérhetővé, és minden egyes számítógép mögött egy vagy több felhasználó ült, akik felé sokféle információt lehetett továbbítani. Kezdetben az internetes kapcsolatok nem rendelkeztek nagy sávszélességgel, így a korlátozott adatforgalom miatt csak egyszerű szöveges adatok megjelenítésére volt lehetőség. Megszületett a HTML, ami egy statikus szöveges dokumentum, melyet egy program segítségével le lehetett tölteni, és megjeleníteni. Az ilyen programokat böngészőknek hívjuk. A HTML oldalakon szöveges adatok jelentek meg, és lehetett az egyes oldalak között navigálni, de ez még elég kevés felhasználói interakcióra adott lehetőséget. Később a statikus HTML oldalakat dinamikus tartalom megjelenítésére akarták felhasználni, de mivel maga a HTML oldal nem alkalmas a dinamikus adatok megjelenítésre, így kialakultak a kliens-szerver alkalmazások, melyekkel kommunikálva már lehetett befolyásolni a HTML oldalak tartalmát.

A kliens-szerver alkalmazások futtatásához szükség volt egy webszerverre, amelyen a dinamikus tartalomhoz szükséges komponensek futottak, és amellyel a böngészők kommunikálni tudtak, hogy felhasználói interakcióra megváltoztassák a felhasználó felé nyújtott tartalmat. A webszerverek fejlődésével egyre könnyebbé vált a kommunikáció, egyre több olyan komponenst tartalmaztak, amelyek megkönnyítették a kérésekre a válaszadást, így egyre több technológia jött létre, amely kihasználta ezeket a lehetőségeket. A Java nyelv elterjedésével egyre nagyobb igény mutatkozott, arra, hogy ezt a jól kezelhető technológiát webes környezetben is lehessen használni. Így sorra jöttek létre azok a technológiák, melyekkel kihasználhattuk a Java nyelv előnyeit a webes alkalmazások fejlesztéséhez, és ezek a technológiák egyre több eszközt biztosítottak a fejlesztőknek, hogy gyorsan és könnyen tudjanak megbízható webalkalmazásokat létrehozni.

Az új technológiák lehetővé tették, hogy a webalkalmazásokat már komoly üzleti dolgokra is lehessen használni, így szükség volt olyan eszközökre, melyekkel könnyen és hatékonyan lehetett a felhasználói interakciót kezelni. A felhasználók által bevitt adatokat ellenőrizni kellett, a kliens felől érkező információt fel kellett dolgozni, és a feldolgozott adatokat vissza kellett juttatni a felhasználó felé valamilyen elfogadható megjelenítésben. Új

problémák merültek fel az alkalmazások fejlesztése közben, például hatékonyan kellett kezelni a felhasználó által elérhető funkciókat, és el kellett választani a megjelenítést az üzleti logikától. Minderre azért volt szükség, hogy a webalkalmazásokat továbbra is hatékonyan lehessen fejleszteni. A rétegek elválasztás továbbá elősegítette a fejlesztést, hiszen a különböző részeken dolgozó fejlesztőknek nem kellet a másik részen történő feladatokhoz érteni, így könnyebbé vált a fejlesztés.

Később, az asztali számítógépek fejlődésével egyre több feladatot lehetett a kliensre bízni, szebbé lehetett tenni a megjelenítést, és idővel a webes alkalmazások kezdték átvenni az asztali alkalmazások tulajdonságait. Az új technológiákkal szebb, és használhatóbb webalkalmazásokat lehet fejleszteni, melyek teljesen új élményt nyújtanak a felhasználó felé..

Jelen dokumentumban megnézzük, hogyan is néztek ki a weboldalak kezdetben, majd megvizsgálunk néhány olyan technológiát, melyek a Java nyelv segítségével eszközöket biztosítanak a webalkalmazások fejlesztéséhez. Végül megnézzük milyen új technológiák állnak rendelkezésre, melyek új szemléletmódot hoztak a weblkalmazások terén.

A HTML

A statikus weboldalak

A HTML (HyperText Markup Language – hyperszöveg jelölő nyelv) egy jelölő nyelv, melyet weboldalak készítéséhez fejlesztettek ki, és ami mára már szabvánnyá nőtte ki magát. A HTML olyan eszközöket biztosít, amellyel strukturált dokumentumokat hozhatunk létre, melyben elhelyezhetünk fejléceket, bekezdéseket, listákat, más weboldalakra mutató linkeket valamint egyéb elemeket is. Egy oldalba beágyazhatunk képeket és objektumokat, és létrehozhatunk interaktív űrlapokat is. A HTML kód olyan szimbólumokat tartalmaz, melyek leírják a megjelenítő/feldolgozó programnak, hogy hogyan is kell értelmezni az adott állomány tartalmát. A feldolgozó program HTML kód esetén általában egy böngészőt takar, de léteznek más programok is, melyek nem weboldallá alakítják a tartalmat, hanem például elektronikus levélnek megfelelő formátummá konvertálják azt.

A HTML dokumentum leíró elemeit négy csoportba sorolhatjuk:

- Strukturális szimbólumok, melyek az adott tartalom célját határozzák meg. Például a <h2>Kezdetek</h2> elem egy második szintű bekezdést reprezentál. A strukturális elemek nem határozzák meg, hogy a konkrét megvalósításuk hogyan történjen, de a legtöbb böngészőnek létezik egy alapértelmezett stílusa, mellyel megformázza ezeket az elemeket.
- Prezentációs szimbólumok, azok az elemek, melyek az adott tartalom megjelenését határozzák meg, az adott elem funkciójától függetlenül. Ilyen elem például a következő: fontos, mely azt jelzi, hogy a tartalmat félkövér betűvel kell megjeleníteni. A legtöbb prezentációs szimbólum az aktuális 4.0-s HTML verzióban elavulttá vált, helyette a CSS alapú formázás a javasolt.
- Úgynevezett hypertext szimbólumok, melyekkel kapcsolat létesíthető a dokumentum egyes elemei, és más dokumentumok között.
- Egyéb olyan elemek, melyek a felhasználóval való interakció megvalósítására használhatók, például gombok és beviteli mezőket reprezentáló szimbólumok.

A HTML oldalak struktúrája

A HTML kód elemeit tag-eknek nevezzük. Egy elem általában tartalmaz egy nyitó tag-et, a tartalmat, és egy záró tag-et. A felsorolt elemek mind tag-ek, és két alapvető tulajdonságból állnak, melyek az attribútumok, és a tartalom. Mind a két tulajdonság meg kell feleljen bizonyos előírásoknak, hogy a HTML dokumentum érvényes legyen. Az attribútumokat a nyitó tag-ban lehet elhelyezni, a tartalom pedig a nyitó- és a záró tag közé kerül. Léteznek olyan HTML tag-ek, melyeknél a záró szimbólum használata nem kötelező.

Maga a HTML dokumentum három fő részre bontható:

- A dokumentum típus definíció (DTD) a dokumentum legelején található, és ezzel a résszel adjuk meg, hogy a dokumentum milyen elemeket tartalmazhat.
- A HTML fejléc, amely technikai és dokumentációs adatokat tartalmaz. Ez a rész nem jelenik meg a felhasználó előtt.
- A HTML törzs, amely a megjelenítendő információkat tartalmazza.

Manapság több törekvés is létezik, hogy a HTML szabványt lecseréljék egy jobban használható szabványra, mert a böngészők eltérően dolgozzák fel a HTML dokumentumokat, így azonos dokumentumokból különböző kinézetű weboldalakat állíthatnak elő a különböző böngészők. Az egyik ilyen szabvány az XHTM lehet, amely lényegében a formai követelményeket szigorítja, például minden esetben használni kell a tag-ek nyitó és záró szimbólumát.

JavaScript

A web script nyelve

Bár a Java nyelvhez csak a neve miatt kapcsolódik, érdemes megemlíteni, mert segítségével a kliens oldalon végezhetünk el feladatokat. A JavaScript egy típus nélküli nyelv , melyet elsődlegesen HTML oldalakon belül használhatjuk, általában olyan funkciók megírásához, amik az oldal DOM struktúrájára hatnak. Ilyen műveletek lehetnek például egy előugró ablak megnyitása, az oldal kinézetének a megváltoztatása, vagy a beviteli adatok validálása még azelőtt, hogy továbbítjuk őket a szerver felé. Mivel a JavaScript kliens oldalon fut, ezért nagyon gyorsan lehet vele a felhasználói eseményekre reagálni. Az egyik hátránya, hogy minden böngésző egyedi módon implementálja a különböző metódusokat, így ugyanannak a műveletnek az elvégzése különböző módon hajtható végre a különböző böngészőkön. Bizonyos alkalmazásokkal ki lehet használni, hogy a JavaScript képes az oldalon történő változtatásokat érzékelni, és ezekre tud reagálni anélkül, hogy elnavigálna a felhasználó az oldalról. Egy ilyen technológia például az Ajax, amelyet a Java technológiával együtt használva új megközelítésbe lehet helyezni a web alapú fejlesztéseket, és a GWT technológia, amely az Ajax technológiára épülő Java alapú webes alkalmazás fejlesztésére alkalmas technológia.

A Szervlet

Mi is az a szervlet?

A szervlet egy olyan Java kódot tartalmazó modul, amely egy (web)szerveren fut, és kliensektől jövő kéréseket szolgálja ki. A szervletek nem kötődnek egy konkrét kliens-szerver protokollhoz, de a legtöbb esetben HTTP protokollt használnak, és ezért a szervlet kifejezés alatt, általában a HTTP protokollt használó szervletet értjük.

A szervletek a Java standard osztályai közül a javax.servlet – a szervletek alapjait biztosító keretrendszer – és a javax.servlet.http – az alap keretrendszert kiterjesztő, HTTP kérések kiszolgálást biztosító – csomagokban található osztályok leszármaztatásával hozhatóak létre. Mivel a szervletek Java nyelven íródnak - amely platform független - és egy standard keretrendszer alapján épülnek fel, így a segítségükkel kifinomult módon hozhatunk létre szerverek funkcionalitását kiterjesztő modulokat, operációs rendszertől és a szerverektől független módon.

A HTTP szervletek általános felhasználási módjai:

- Form-ok által bevitt adatok feldolgozása és tárolása
- Dinamikus tartalom előállítása, például valamilyen adatbázisból nyert tartalom megjelenítése
- Az állapotmentes HTTP protokoll felé egy saját állapotkezelő rész építése, például egy online webbolt esetén melyet egy időben több vásárló használ, szervletek segítségével megoldhatjuk, hogy a megfelelő vásárlóhoz a megfelelő bevásárló kosarat kezeljük

A szervlet előnyei a CGI-vel szemben.

A hagyományos módja, hogy plusz funkcionalitást adjunk egy webszerverhez az a CGI (Common Gateway Interface). Ez nem más, mint egy nyelv független interfész, ami lehetővé teszi a szervernek, hogy elindítson egy külső folyamatot, amely környezeti változókon, parancssoron, a standard bemeneten, és a saját beviteli csatornáján keresztül

kaphat információkat egy kéréstől, és az adatokat pedig a standard kimenetre irányítja. Ebben a folyamatban minden egyes kérést a CGI program egy különálló példánya különálló szálban szolgál ki.

A szervletek a CGI programokhoz képest számos előnnyel rendelkeznek:

- Egy szervlet egy szálként fut, így nem kell több kéréshez több szálat indítani, ami erőforrást takarít meg az alkalmazásnak
- A szervlet a memóriában marad a kérések kiszolgálása között, míg a CGI programot minden egyes kérés kiszolgálása előtt be kell tölteni, és el kell indítani
- Egy szervletből egyetlen példány létezik, ami egyidejűleg szolgálja ki az összes hozzá érkező kérést, így kevesebb memóriát igényel, és könnyebben lehet a perzisztens adatokat kezelni
- A szervleteket futtathatjuk egy korlátozott szervlet motor alatt, ami lehetővé teszi, hogy az olyan szervleteket, amelyek nem teljesen biztonságosak, vagy esetlegesen veszélyt jelenthetnek, biztonságos környezetben futtassunk

A szervlet architektúra alapjai

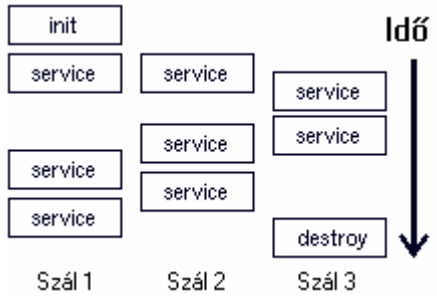
Egy szervlet legáltalánosabb formája egy olyan osztály példánya, amely implementálja a javax.servlet.Servlet interfészt. A legtöbb esetben azonban a szervletek egy olyan osztály kiterjesztései, melyek az előbb említett interfész egy standard implementációját valósítják meg, mint például a javax.servlet.GenericServlet, vagy a javax.servlet.http.HttpServlet.

Egy szervlet inicializálásákor a (web)szerver betölti a szervlet osztályt - valamint az általa hivatkozott osztályokat -, és példányosítja a szervletet az alapértelmezett konstruktorának paraméterek nélküli meghívásával. Ezek után hívja meg az init metódusát. Ebben a metódusban történhetnek meg a szervlet működéséhez szükséges egyszeri inicializálások, és itt kell létrehozni a ServletConfig objektumot, amelyet a későbbiekben a getServletConfig() metódussal lehet majd elérni. Ha egy szervlet a GenericServlet osztály kiterjesztése, akkor nem kell mást tennünk, mintsem az init metódus elején meghívjuk a szülő osztály ugyanazon metódusát. A ServletConfig objektum tartalmazza a szervlet paramétereit.

Mikor egy szervlet elindult, minden egyes kérés kiszolgálásakor a service(ServletRequest req, ServletResponse res) metódusa hívódik meg. Ez a metódus egyidejűleg több kérést is kiszolgálhat, így az implementálásakor ügyelni kell arra, hogy a különböző kéréseket egymástól függetlenül biztonságosan kezeljük.

Ha szükség van egy szervlet eltávolítására, például szerver leállításkor, vagy ha új verziót akarunk a szervletből kitelepíteni, a destroy metódus hívódik meg. Létezhet olyan eset, amikor a destroy metódus meghívásakor még egyes szálak futtatják a service metódust, így itt is ügyelni kell az adatok biztonságos kezelésére. A destroy metódusban kell felszabadítani minden egyes erőforrást, amit az init metódusban lefoglaltunk a szervlet számára.

Egy tipikus szervlet életciklus:



A HTTP protokoll és a szervletek

Mivel a web böngészők a HTTP protokollt használják, hogy kéréseket küldjenek a web szerver felé, ezért érdemes áttekinteni a HTTP protokoll és a szervlet kapcsolatát.
A HTTP egy kérés-válasz orientált protokoll. A HTTP kérés tartalmaz egy request metódust, egy URI-t, egy fejlécet, és egy törzset, ami lehet üres is. A HTTP válasz tartalmaz egy eredmény kódot, egy fejlécet és egy törzset.

Egy HttpServlet service metódusa a különböző típusú HTTP kéréseket különböző Java metódusoknak továbbítja. A HTTP kérések közül a GET, HEAD, PUT, POST, DELETE, OPTIONS és a TRACE metódusokat ismeri, egyéb metódusok esetén hibaüzenetet kapunk. A HTTP kéréseket a metódus alapján a megfelelő Java metódushoz irányítja (például egy GET metódusú kérés esetén a doGet Java metódus hívódik meg). Mivel az OPTIONS, TRACE és

HEAD metódusú kérések kiszolgálását végző Java metódusok implementációi már léteznek a HttpServlet osztályban, így általában nincs rá szükség, hogy felülírjuk ezeket. A fejlesztőnek már csak az a feladata, hogy a maradék négy metódus közül legalább egyet implementáljon. Ha egy kérés típushoz tartozó metódust nem implementálunk, és a szülőosztályban sincs implementálva, akkor a metódus alapértelmezetten hibaüzenettel fog visszatérni.
Minden egyes kéréshez tartozó metódus két paraméterrel hívódik meg, egy HttpServletRequest és egy HttpServletResponse objektummal. Az előbbi a kérés adatait tartalmazza, míg a válasz tartalmát a másodikban adhatjuk meg.

Konklúzió

A szervletek bár önmagukban nem biztosítanak sok funkcionalitást, egyéb eszközök segítségével nagyon hatékonyan használhatóak webalkalmazások fejlesztésére. Ennek következtében a szervlet technológia vált a Java alapú vebalkalmazások alapvető eszközévé.

A JSP

A HTML oldalak továbbfejlesztése

A JSP (Java Server Pages) technológiát szintén arra találták ki, hogy a segítségével egyszerűen és gyorsan hozhassunk létre dinamikus tartalommal rendelkező weboldalakat, és mint a Java technológiának a része, könnyen alkothassunk olyan web alapú alkalmazásokat, melyek nagyszámú webszerverrel, alkalmazás szerverrel, web böngészővel és fejlesztést elősegítő eszközzel képesek együttműködni.

A fejlesztést elősegítő tulajdonságok a következők:

- *A tartalom generálása elválik a megjelenítéstől*
  A JSP technológia használatakor a fejlesztők HTML vagy XML tag-eket használnak a tartalom megjelenítéséhez, míg a tartalom előállításához JSP tag-eket vagy scriptleteket. A logika, ami alapján a tartalom generálódik, tag-ekbe és Java bean-ekbe csomagolódik (A Java bean-ek szerializálható Java objektumok, melyek getter és setter metódusokkal hozzáférést biztosítanak a tulajdonságaikhoz), és a szerver oldalon fut, így a tartalom módosítása nélkül lehet változtatni a megjelenítésen. A szerveren a JSP motor dolgozza fel a tag-eket és scriptleteket, létrehozza a tartalmat, és HTML vagy XML formájában visszaküldi az eredményt a böngészőnek. Így biztonságosan dolgozhatunk a szerveren anélkül, hogy bárki hozzáférhetne a kódhoz, és a hordozhatóság sem sérül a web böngészők felé.
- *Könnyedén lehet újrahasznosítható komponenseket létrehozni*
  A legtöbb JSP oldal újrahasznosítható komponensen alapul, melyeket a fejlesztők megoszthatnak egymással, így nem kell ugyanazon funkciókat többször lefejleszteni.

- *A weboldal fejlesztést tag-ekkel egyszerűsíti*
  A weboldal fejlesztők nem mindig ismerik a scriptnyelveket. A JSP technológia a dinamikus tartalom generálásához szükséges funkcionalitás nagy részét JSP specifikus XML tag-ekbe rejti el. Ezek a tag-ek példányosíthatnak Java bean-eket, hozzáférhetnek az általuk tartalmazott adatokhoz, és egyéb olyan feladatokat láthatnak el, melyek leprogramozása időigényes lehet.

Léteznek egyéb előnyök is, amit a JSP technológia szolgáltat. Mivel a JSP oldalak natív scriptnyelve a Java programozási nyelven alapul, és az oldalakból Java szervletek készülnek (amikor hivatkozás történik egy JSP oldalra, egy előfordító segítségével létrejönnek a szükséges Java osztályok), a JSP technológia rendelkezik a Java technológia összes előnyével, mint például a robosztus memória-kezelés, és a biztonság. A Java technológia széleskörű elterjedése folyamán sok web szerver támogatja a JSP technológiát, és sok fejlesztő eszköz létezik, ami megkönnyíti a JSP oldalak fejlesztését.

Hogyan is néz ki egy JSP oldal?

Egy JSP oldal szinte teljesen ugyanúgy néz ki, mint egy átlagos HTML vagy XML oldal, azzal a különbséggel, hogy olyan elemeket is tartalmaz, amelyeket a JSP motor kiszűr, feldolgoz, és általában a helyükre valamilyen webes tartalom generálódik. A JSP komponensek a következők lehetnek:

- *JSP direktívák*
  A direktívák a JSP motor számára tartalmaznak utasításokat. Léteznek az oldallal kapcsolatos direktívák (page direktíva), mint például a pufferéléshez vagy hibakezeléshez kapcsolódó információk. A nyelvi (language) direktívák a scriptnyelvet határozzák meg, a hozzá tartozó egyéb kiterjesztésekkel. Az include direktívát arra használhatjuk, hogy külső dokumentumot adjunk hozzá az oldalunkhoz. A taglib direktívával pedig saját tag-eket tartalmazó könyvtárat adhatunk meg, melyet az oldalon belül használhatunk.

- *JSP tag-ek*

  A JSP tag-ek tartalmazzák a JSP motor által feldolgozott tartalmat. A dinamikus tartalom ezekbe a részekbe kerül. Egy JSP tag mögött egy olyan Java osztály található, amely a Tagsupport osztály egy leszármaztatása. Egy ilyen osztály különböző metódusai szolgálnak arra, hogy a JSP tag helyére legeneráljuk a szükséges webes tartalmat. Léteznek standard tag-ek, amelyeket minden JSP oldalon használhatunk, de létrehozhatunk saját tag-eket is, melyeket újra felhasználhatunk, és megoszthatunk másokkal, hogy gyorsítsuk a fejlesztést.

- *Script elemek*

  A JSP oldalak tartalmazhatnak kisebb scripteket, amiket scripleteknek nevezünk. A scriptlet olyan kódrészlet, amelyet a kérés időpontjában dolgozunk fel. A scriptletekben használhatunk statikus elemeket, melyek az oldalon találhatók, és így előállíthatunk dinamikusan generált oldalakat. A scriptleteket a <% %> jelek között helyezhetjük el. A hátránya, hogy ilyenkor egy interpreter dolgozza fel a kódot, ami csökkenti az alkalmazás sebességét.

A JSP technológiát a legegyszerűbben egy kódrészlettel lehet bemutatni:

```
<HTML>
<%@ page language=="java" imports=="java.util.Calendar" %>
<P>A mai nap </P>
<jsp:useBean id=="clock" class=="Calendar" />
<UL>
<LI>Év: <%==clock.getYear() %>
<LI>Hónap: <%==clock.getMonth() %>
<LI>Nap: <%==clock.getDayOfMonth() %>
</UL>
<% if (Calendar.getInstance().get(Calendar.AM_PM) ==== Calendar.AM) { %>
Jó reggelt!
<% } else { %>
Jó napot!
<% } %>
<%@ include file=="alairas.html" %>
</HTML>
```

A kódrészlet a következő tartalmazza elemeket:

- Egy JSP direktívát, amely egy Java osztály használatát engedélyezi
- Fix adatokat, melyeket a JSP motor nem dolgoz fel. Ezek tipikusan HTML vagy XML tagek szoktak lenni.
- Kifejezéseket, melyek az aktuális dátumot jelenítik meg.
- Egy scriptletet, amely a napszaktól függően más üdvözlést jelenít meg.

A JSP oldalak modellje

A JSP oldalakat egy JSP motor futtatja, ami egy webszerven, vagy alkalmazás szerveren fut. A JSP motor kap egy kérést a klienstől egy JSP oldalon keresztül, és ez alapján legenerálja a választ, majd visszaadja a kliensek. A JSP oldalakból Java szervletek készülnek, így a fejlesztő hozzáférhet a Java technológia által biztosított eszközkészlethez. Egy JSP oldal kezdetben nem létezik, hanem az első rá való hivatkozáskor jön létre, mégpedig ekkor készül el az oldalból a szervlet. Ez azt eredményezheti, hogy az első alkalommal, amikor valaki megtekinti az oldalt, akkor lassabban jelenik meg a felhasználó előtt az oldal, de később már nem lesz ilyen probléma, sőt mivel a szervletek a memóriában tárolódnak, ezért a továbbiakban igen gyors válaszidővel jelenhet meg az oldal.

Konklúzió

A JSP technológia egy elég jó alternatívát nyújt a webes fejlesztéshez, mivel egyszerűbbé teszi az oldalak létrehozását, és támogatja az újrafelhasználhatóságot. Ugyan a technológia lehetőséget nyújt inline kódok elhelyezésére egy oldalon belül, de ezeket az elemeket manapság már nem használjuk, így a JSP oldalak csupán a megjelenésért felelősek.

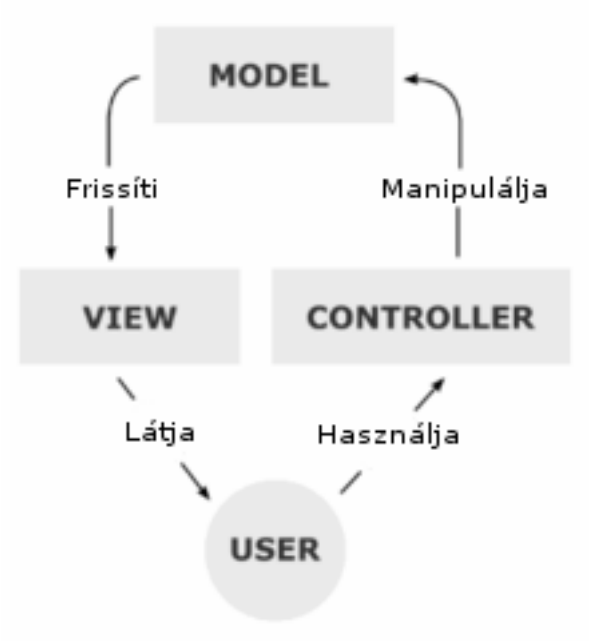
Struts

Miért is jött létre a Struts?

Mikor kitalálták a Java szervleteket, a programozók hamar felismerték a technológia előnyeit, hiszen gyorsabb és hatékonyabb megoldás volt, mint a CGI programok írása. A JSP technológia megkönnyítette a megjelenítéshez szükséges programok előállítását, viszont a visszájára fordította a szervletek fejlesztését, hiszen a JSP oldalunkból jött létre a szervlet. A technológia segítségével már egyszerűen lehetett ötvözni a HTML kódot a Java kóddal, de még nem oldotta meg a folyamatok vezérlését, és egyéb olyan problémákat, amelyek a webalkalmazások sajátos problémái voltak. Azonban ha a két technológiát egymás mellett használjuk már lehetséges egyéb problémák megoldására is használni őket, a JSP oldalak felelhetnek a megjelenítésért, míg a szervletek megoldhatják a folyamatok vezérlését és az egyéb problémákat. Ez a rendszer pedig követi az MVC (Model View Controller) módszertant, amely a Smalltalk MVC keretrendszeréből ered.

A Struts és a MVC (Modell Nézet Vezérlő)

Először is tisztázzuk az MVC fogalmát. A lényege az MVC módszertannak, hogy elkülöníti az üzleti logikát a felhasználó felé mutatott megjelenítéstől, és a bejövő kérésektől. A modell vezérli az információkat, és értesíti a felhasználókat, ha bármilyen változás történik. A nézet a modellt olyan formára hozza, ami lehetővé teszi a felhasználói interakciót. A vezérlő rész pedig fogadja a bejövő kéréseket, és továbbítja az utasításokat a megfelelő modell objektumok felé. Így áll össze a modell, nézet, vezérlés.



A Struts modell részét gyakran két alrendszendszerre bonthatjuk: a rendszer belső állapotára, és azokra az eseményekre, melyek ezt az állapotot képesek megváltoztatni. A Java alapú rendszerekben a rendszer állapotát általában Java bean-ekkel szokás ábrázolni. A Java bean-ek tulajdonságai jelentik a rendszer állapotának részleteit. A rendszer bonyolultságától függően, ezek a bean-ek lehetnek egyszerű Java bean-ek, melyek csak a saját tulajdonságaikat ismerik, és ezeket tudják megváltoztatni, vagy lehetnek olyan Java bean-ek, melyek más komponensek segítségével képesek megváltoztatni a rendszer állapotát.

Kisebb alkalmazások esetén lehetséges, hogy a rendszer állapotát befolyásoló folyamatok a vezérlő részbe kerülnek. Ez akkor lehet célravezető, ha az üzleti logika egyszerű, és nincs arra szükség, hogy azt más környezetben is felhasználjuk. Nagyobb komplexitású alkalmazások esetén az üzleti logikát kizárólag a modell részbe érdemes implementálni, így a vezérlő rész a folyamatok kívánalmaitól függően, a megfelelő metódusokat használja a rendszer állapotának manipulálásához.

A Struts alapú webalkalmazások nézet részét általában JSP oldalak alkotják. Ezek az oldalak tartalmazhatnak statikus HTML vagy XML kódot, és lehetséges olyan dinamikus tartalmat is elhelyezni bennük, úgynevezett JSP tag-eket, melyek a kérés idejében értékelődnek ki. A Struts keretrendszer magában foglal néhány egyedi tag könyvtárat, melyek megkönnyítik az olyan felhasználói interfészek létrehozását, és amelyek támogatják a többnyelvűséget, és együttműködnek az ActionForm osztályokkal (Az ActionForm osztályok

a HTML form-ok szerver oldali reprezentációi). Az ActionForm-ok megkapják a rendszer által elvárt beviteli adatokat, és képesek azokat validálni, ha szükség van rá.

A Struts technológia vezérlő részét az ActionServlet és az ActionMappings osztályok alkotják. Igazából ez a rész a leglényegesebb újítás a Struts technológiában. A vezérlő rész feladata, a klienstől bejövő kérések fogadása, majd a bejövő adatok alapján annak eldöntése, hogy milyen üzleti tevékenységet kel elvégezni. Ezután a vezérlő továbbadja a feladatot a megfelelő nézet komponensnek, amely a feladat elvégzése után továbbítja a felhasználó felé a következő lépéshez szükséges felhasználói interfészt.

A Struts keretrendszer vezérlő részének alapvető komponense az ActionServlet osztályú szervlet. Ezt a szervletet ActionMappings-ek csoportjának a definiálásával kell konfigurálni. Egy ActionMappings egy utat definiál, amely egy kérés URI-ja alapján képes meghatározni, hogy mely Action osztályt kell meghívni. Az Action osztályok tartalmazzák azokat a metódus hívásokat, melyekkel meg lehet változtatni a rendszer állapotát, ők dolgozzák fel a folyamat kimenetelét, és ezek után továbbadják a vezérlést a megfelelő nézet komponensnek, hogy létrehozza a kérésre a választ.

A keretrendszer lehetőség biztosít arra is, hogy a vezérlő rész felé olyan plusz paramétereket továbbítsunk, melyekkel befolyásolhatjuk a vezérlő rész által irányított folyamatot. Ezen túl lehetséges olyan logikai neveket definiálni, melyek segítségével úgy lehetséges bizonyos oldalakra eljutni, hogy nem ismerjük ténylegesen az oldal elérésének a helyét. Ez a megoldás további lehetőséget biztosít a vezérlő és a nézet logikájának elválasztására.

A vezérlő folyamat

Az alkalmazás indításakor a vezérlő feldolgoz egy konfigurációs fájlt (struts-config.xml), és ez alapján létrehozza a további szükséges vezérlő objektumokat. Ezeket az objektumokat együtt nevezzük Struts konfigurációnak. A Struts konfiguráció tartalmazza többek között az ActionMappings objektumok halmazát. A vezérlő komponens az ActionMappings objektumok alapján irányítja a HTTP kéréseket a keretrendszer további objektumaihoz, melyek lehetnek JSP oldalak, vagy esetleg a fejlesztő által létrehozott osztályok, melyek az Action osztályból származik. Általában egy kérés először egy Action

osztály felé továbbítódik, majd a válasz egy JSP oldal formájában vagy esetleg más reprezentációs formában jelenik meg.

A webes alkalmazások egyik kulcsproblémája, hogy két kérés között a felhasználó által bevitt adatokat hogyan nyerjük ki, és hogy dolgozzuk fel. A Struts keretrendszer biztosít számunkra egy olyan Java bean-t, amely segítségével egyszerűen kezelhetjük a felhasználói inputot. Ezek az osztályok az ActonForm osztály leszármazottai, és segítségével könnyen tárolhatjuk és validálhatjuk a kérés által bejövő adatokat.

A Struts keretrendszer biztosított tag könyvtárakat úgy tervezték meg, hogy támogassa a többnyelvű rendszerek fejlesztését. A megjelenítendő szövegeket MessageResource-ok segítségével érhetjük el, melyek nem csak a többnyelvűség miatt hasznosak, hanem így az összes üzenetet és a feliratot egy közös helyen kezelhetjük, és egy változtatás esetén nem kell minden oldalon módosítanunk, hogy mindenhol megjelenjen a változás.

Konklúzió

A Struts keretrendszer sok olyan problémára ad megoldást, amelyekk tipikusan a webalkalmazások problémái. A később létrehozott technológiák közül is sok épül a Struts által kidolgozott elemekre. Azonban a Struts keretrendszer használata kissé nehézkes, hiszen sok információt konfigurációs fájlokban tárol, ezért érdemes más technológiákkal együtt használni, amelyek megkönnyítik a konfigurációs fájlok karbantartását.

Beehive

Az annotációk kihasználása

A Beehive technológia azért jött létre, hogy megkönnyítse az webalkalmazások fejlesztését. Metaadatok felhasználásával elérhetjük azt, hogy a konfigurálás elvégzéséhez, és a programozás deklaratív részéhez kevesebbet kelljen kódolni, és így lecsökkentse a fejlesztésre fordított időt. A Beehive technológia több különböző projektet foglal magába, amiket használhatunk egyben, vagy akár külön-külön is attól függően, hogy az alkalmazásunknak mire is van aktuálisan szüksége. A két legfontosabb része a technológiának a NetUI projekt, és a Controls projekt. A NetUI projekt a Struts technológia használatára épül, és az MVC módszertanban megismert vezérlő részének a programozását könnyíti meg. A Controls projekt pedig olyan komponenseket tartalmaz, amelyek a nézet rész fejlesztését teszik egyszerűbbé, és átláthatóbbá. Ezen túl a Beehive technológia tartalmaz még olyan komponenseket, melyek a J2EE erőforrás kezelő részének az absztrakciói.

NetUI

A NetUI a Beehive technológia része, amely a web alkalmazásunk azon részét képezi, amely a felhasználói interfész és az alkalmazás üzleti logikája közötti kapcsolatot irányítja. Két részt tartalmaz, egy PageFlow részt, és JSP tag-ek egy csoportját, melyek nagyszerűen használhatóak a fejlesztéshez. De miért is jó ez nekünk, hiszen a Struts technológia is éppen erre való. Először is, mivel a NetUI a Struts technológián alapul, ezért követi az MVC módszertant, tehát itt is különválik a megjelenítés a folyamat vezérlésétől annak minden előnyét magába foglalva. Másodszor, bevezeti a PageFlow programozási modell fogalmát, amely segítségével olyan PageFLow-kat hozhatunk létre, amelyeket egyéb folyamatoknál is felhasználhatunk, mint azoknak a része. A legfontosabb pedig, hogy egyetlen osztályban egységesíti a folyamat logikáját, az állapotok és metaadatok kezelését, amely átláthatóbbá és kezelhetőbbé teszi a fejlesztést. A nézet oldalon pedig további eszközöket biztosít, amelyekkel színesíthetjük a felhasználó felé biztosított lehetőségeket.

Főbb jellemvonások

A NetUI teljesen leegyszerűsíti a Java alapú webalkalmazások fejlesztését. Ha a NetUI segítségével fejlesztjük az alkalmazást, akkor Java osztályokat kell implementálni, ezen túl JSP oldalakat fejleszteni és semmi mást. A konfigurációs fájlokkal és egyéb beállításokkal egyáltalán nem, vagy csak nagyon kis mértékben kell foglalkozni. A NetUI ráadásul teljesen szétválasztja a megjelenítési réteget és az adat feldolgozásához használt réteget, így az eddig átláthatatlan JSP kódot sokkal egyszerűbbé és érthetőbbé tehetjük. Az adat feldolgozást és az alkalmazás konfigurálását pedig egyetlen Java osztállyal, és egy egyszerű deklaratív programozási modellel kezelhetjük.

- *A deklaratív programozási modell*
  A legtöbb olyan programozási feladatot, ami magával a web alkalmazással kapcsolatos egy deklaratív programozási modellel végezhetjük el, amit a Java 5-ös verziójának új eszközének a segítségével, annotációval helyezhetünk el a Java osztályunkban. Így nincsen szükség a továbbiakban arra, hogy a konfigurációs fájlokkal foglalkozzunk. A navigációt, a kivételkezelést és a bevitt adatok validálását is egyetlen osztály segítségével meg tudjuk oldani.
- *Intelligens PageFlow kezelés*
  Ha egy felhasználó belép a PageFlow hatáskörébe, akkor a PageFlow-t kezelő osztályból létrejön egy példány. Amíg a felhasználó nem hagyja el a PageFlow hatáskörét, a releváns adatok mind a kezelő osztályban tárolódnak, amihez minden kapcsolódó rész, például az Action-ök és kivételkezelő részek, hozzáférnek. Ha a felhasználó elhagyja a PageFlow hatáskörét, a tárolt információk felszabadulnak. A viselkedést ugyan lehet PageFlow-nként befolyásolni, de az automatikus erőforrás felszabdítás segít abban, hogy kevés memóriát használjunk.
- *Moduláris PageFlow-k*
  A web alkalmazások több PageFlow-t is tartalmazhatnak, így nem kell egyetlen hatalmas osztályt használnunk a folyamatok megvalósítására.

- *Öröklés és megosztott PageFlow-k*

  A PageFlow-kat ugyanúgy lehet egymásból leszármaztatni, mint egy egyszerű Java osztályban. Ezen túl lehetséges megosztott PageFlow-kat létrehozni, melyek olyan metódusokat tartalmaznak, amelyeket más PageFlow-k is felhasználhatnak.
- *Beágyazott PageFlow-k*

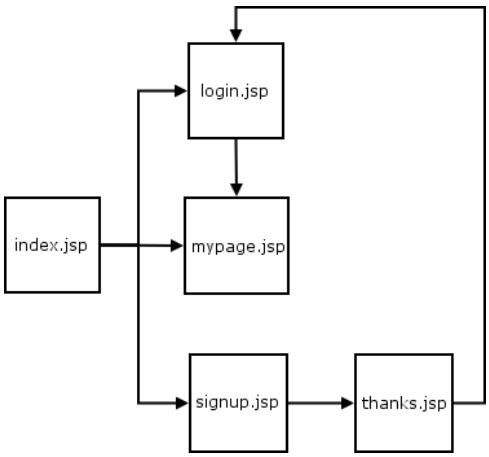
  Mivel a PageFlow-k modulárisak, így lehetőség van egyiken belül egy másikat elhelyezni. Igazából a folyamat nagyjából úgy néz ki, hogy az eredeti PageFlow-t ideiglenesen félrerakjuk, és addig az új PageFlow-t használjuk, majd ha befejeztük a munkát az új PageFlow-n belül, akkor elővesszük újra az eredeti PageFlow-nkat. A nagy előnye ennek a folyamatnak, hogy így az eredeti munkamenet adatai megmaradnak, így nem kell nekünk bajlódni ezek letárolásával, majd betöltésével.
- *Kivételkezelés és validálás*

  Az annotációkkal megvalósított deklaratív programnyelv eszközöket biztosít arra is, hogy a kivételkezelést és a bevitt adatok validálását elvégezzük. Így nem kell konfigurációs fájlokban megadni ezeket a részeket.
- *Integráció*

  A NetUI képes együttműködni a JSF (Java Server Faces) technológiával, és mivel a Struts technológiára épül, teljes mértékben együttműködik vele. Egy PageFlow-t kezelő osztályból egy Struts modul fordítódik.

Példa

A PageFlow müködését leginkább egy példával lehet szemléltetni



A felhasználó az index.jsp által lép be a PageFlow hatáskörébe. Innen több út lehetséges. Ha még nincs regisztrálva a felhasználó, a signup.jsp-re lép tovább, ahol regisztrálhatja magát, majd miután ez sikeresen megtörtént a thanks.jsp oldalra lép tovább. Ezután ha akar, bejelentkezhet a login.jsp oldalon. Ugyanerre az oldalra úgy is el lehet jutni, ha már regisztrálva volt a felhasználó, akkor egyből a login.jsp oldalra navigál a kezdőoldalról. A login.jsp és signup.jsp oldalakon validálások és hibakezelések mennek végbe, hogy a felhasználó megfelelő adatokat küldött-e el az alkalmazásnak. Ha nem, akkor ugyanazokra az oldalakra jut vissza, csak értesítést kap, hogy van olyan adat, ami nem volt megfelelő, így próbálja újra. Ha sikeres volt a bejelentkezés, a mypage.jsp oldalra lehet eljutni, ahol az aktuális PageFlow hatásköre megszűnik, hiszen érdemes szétválasztani a különböző részeket a programban, így a bejelentkezés folyamatát egyetlen osztállyal tudjuk vezérelni.

Controls

A Beehive technológia másik nagyobb projektje a Controls projekt, ami azért jött létre, hogy a nagy komplexitású rendszerek esetén egyszerűbbé tegye a kliens által indítható

folyamatok szétválasztását. Egy egyszerű folyamat esetén, ahol a felhasználó csak annyit tesz, hogy elindít egy folyamatot, a háttérben lehet, hogy egy olyan komplex folyamat indul el, ami több különböző technológia használatát is igényli. Bár lehet, hogy ez a folyamat részekre bontva egyszerűnek tűnik, de a megvalósítása eléggé bonyolult lehet egy programozónak, hiszen több különböző technológiát is meg kell ismernie hozzá, és ráadásul ezt úgy kell megoldania, hogy az egy folyamathoz tartozó részek egy helyen legyenek, ami nem egyszerű. A Controls projektben ezt Java bean-ekkel lehet megoldani, a kliens felé egy folyamat indítása pedig egy osztály példányosításával jár, és ennek a példánynak egy metódusának a meghívásával. Így egyszerűen lehet kezelni a felhasználó által indított folyamatokat.

Ezen felül a Controls projekt biztosít egyéb konfigurálási lehetőséget, amely segítségével kiválaszthatjuk, hogy mely erőforrások használatával tudjuk a legoptimálisabban kiszolgálni a klienst. Ennek elérése érdekében lefoglalhatunk bizonyos erőforrásokat, melyek a kapcsolatok és egyéb szükséges részek használatához szükségesek, majd a folyamat végén ezeket elengedjük, mindezt úgy, hogy az a kliens számára mindez el van rejtve, azaz neki nem kell azzal foglalkozni, hogy hogyan is valósul meg mindez. Ráadásul lehetőség van arra is, hogy ezeket az erőforrásokat paraméterezve használjuk, így különböző kérések esetén különböző folyamatokat indíthatunk el, melyek csak annyiban különböznek, hogy más szerverrel, vagy más adatbázissal kommunikálnak. A Controls projekt célja tehát nem az, hogy egy teljesen új módot hozzon létre az erőforrások elérésére, hanem hogy egységesítse azok használatát, hogy ne kelljen mindegy programozónak, minden egyes technológiához érteni, hogy használhassa azokat.

Konklúzió

A BeeHive technológia nagyban leegyszerűsíti, és átláthatóbbá teszi a Java alapú veb alkalmazások fejlesztését. Mivel a Struts technológiára épül, megvalósítja az MVC modellt, így remekül használható. Ennek ellenére már nem fejlesztik tovább helyette más technológiák irányába mozdult el a közösség.

Java Server Faces (JSF)

Általánosságban a JSF-ről

A JSF technológia két alapvető eszközt tartalmaz:

- Egy API csomagot, amely a felhasználói interfész komponensek megjelenítéséhez és állapotainak menedzseléséhez szükséges eszközöket tartalmazza, továbbá lehetővé teszik az események kezelését, a bementi adatok validálását, definiálja az oldalak közti navigációt, és eszközöket biztosít a többnyelvűség kezelésére.
- Olyan JSP saját tag könyvtárat, amely lehetővé teszi a JSF kifejezések használatát egy JSP oldalon belül.

A JSF technológiát úgy tervezték, hogy rugalmas legyen, és használja ki az ez idáig létező szabványos felhasználói interfészekre vonatkozó és web szintű koncepciókat anélkül, hogy a programozókat egy kitüntetett jelölő nyelv, protokoll vagy kliens eszköz használatára korlátozná. A JSF technológia által tartalmazott UI komponens osztályok magukba foglalják a komponens funkcionalitását, de a kliens specifikus megjelenítési részt nem, így lehetővé teszik, hogy az UI komponenseket különböző kliensekkel használhassuk. Az UI komponensek funkcionalitását egyedi rendererekkel kombinálva - melyek definiálják a meghatározott klienshez szükséges egyedi tulajdonságokat -, a fejlesztők egyedi tag-eket hozhatnak létre a meghatározott klienshez. A JSF technológia a kényelem kedvéért tartalmaz egy egyedi renderert és egy egyedi JSP tag könyvtárat a HTML kliensek kiszolgálásához, így a technológia segítségével egyszerűen fejleszthetünk web alapú alkalmazásokat a Java nyelv segítségével.

Mivel a JSF technológia kialakítása közben a használhatóság volt a fő szempont, ezért a megjelenítés és az alkalmazás üzleti logikája élesen elválik az architektúrában, de mégis könnyen össze lehet kapcsolni a megjelenítési réteget az üzleti logika kódjával. Ez a kivitelezés így lehetővé teszi, hogy a webalkalmazás fejlesztésén dolgozó csapat minden egyes tagja a saját fejlesztési fázisával foglalkozzon, és egy egyszerű modellt biztosít a részek

összekapcsolásához. Például egy weboldal fejlesztő, akinek nincs programozási tapasztalata, a nélkül tud használni egy JSF UI komponens tag-et, és ezzel összekapcsolni az alkalmazás üzleti logikáját a weboldallal, hogy bármiféle scriptet kelljen hozzá írnia.

Hogyan is néz ki egy JSF alkalmazás

A JSF alkalmazás legnagyobb része teljesen ugyanazokból a részekből áll, mint egy tipikus Java alapú webalkalmazás:

- JSP oldalak egy csoportjából, bár a technológia nem korlátozza a megjelenítési részt, így JSP oldalak helyett bármilyen már reprezentációs technológia állhat.
- Olyan osztályok csoportjából, melyek meghatározzák az UI komponenseknek a tulajdonságait, és funkcionalitását.
- Egy konfigurációs fájlból, amely meghatározza az oldalak közti navigációs szabályokat, és konfigurálja a egyedi komponenseket.
- Egy olyan fájlból, ami leírja a telepítési utasításokat (web.xml)
- Egyedi objektumok egy olyan csoportjából, melyet az alkalmazás fejlesztői hoztak létre.
- És olyan egyedi tag-ekből, melyek az egyedi komponensek reprezentálásért felelősek az oldalon.

A JSF előnyei

A JSF technológiát tehát azért érdemes használni, mert a jól kialakított programozási modell és az egyedi tag könyvtárak jelentősen csökkentik a webalkalmazások fejlesztéséhez és fenntartásához szükséges időt. Továbbá minimális erőfeszítéssel tudunk az oldalakhoz egyedi komponenseket hozzáadni, a komponensek által előhívott eseményeket egyszerűen hozzá tudjuk kötni a szerver oldali logikához, újrafelhasználható és kiterjeszthető komponenseket tudunk alkotni, és az UI állapotát könnyen letárolhatjuk és visszaállíthatjuk bármikor a segítségével. A JSF technológiát gyakran az Ajax technológiával együtt emlegetik, amely segítségével Rich Intenet Alkalmazásokat hozhatunk létre (a későbbiekben

megemlítjük, hogy mik is a Rich Internet Alkalmazások). A JSF kezdetben nem támogatta az Ajax technológiát, és JavaScript segítségével lehetett kihasználni az Ajax által támogatott funkcionalitást, de a későbbiekben belevették a technológia támogatását. A JSF 2.0-ás specifikációja már beépített támogatást biztosít az Ajax technológia használatához, amellyel könnyedén kezelhetjük az Ajax kérések életútját, támogatást kapunk a az Ajax események kezeléséhez, és egyéb Ajaxhoz kapcsolódó funkcionalitást használhatunk egyszerűen.

AJAX

Egy másik megközelítés

Manapság egyes weboldalak teljesen úgy néznek ki és úgy viselkednek, mint az asztali alkalmazások anélkül, hogy bármilyen böngésző specifikus jellemzőt kihasználnának, vagy egyéb böngészőbe integrálható kiegészítő szükséges lenne a futtatásukhoz. A webes alkalmazások hagyományosan HTML oldalak egy csoportja, melyeket újra kell tölteni, ha bármely részen változtatást akarunk eszközölni. Azonban a JavaScript és a CSS használatával már olyan dinamikus webes alkalmazásokat tudunk létrehozni, amelyek minden böngészőn megfelelően működnek. Ezeket a lehetőségeket használja ki az AJAX technológia.

Az AJAX (Aszinkon JavaScript technológia és XML)

A JavaScript technológia használatával képesek lehetünk arra, hogy a HTML oldalunk aszinkron hívásokat intézzen a szerver felé, és a visszajövő adatot feldolgozza, majd ezt az adatot arra használja, hogy megváltoztassa a HTML oldal tartalmát. Bár ez a technika már régóta része volt a JavaScript technológiának, a nagy áttörést az XMLHttpRequest objektum hozta, ami bár nem része a JavaScript specifikációnak, mégis szinte minden böngésző támogatja a használatát.

Ami az AJAX alapú webalkalmazásokat egyedivé tesz az az, hogy a kliens tartalmazza az oldal specifikus folyamatot JavaScript-ként beágyazva. Az oldal a különböző JavaScript események hatására, mint például az oldal betöltődése, az egér vagy kurzor mozgása esetén, különböző műveleteket végez el. Az AJAX események lehetővé teszik a megjelenítés elválasztását a mögötte lévő üzleti logikától. Ahhoz, hogy technológia működjön, a más szerver oldali architektúra szükséges, mint a hagyományos webalkalmazások esetén, hiszen a azok minden egyes kérés esetén újragenerálják a HTML dokumentumokat, melyeket a felhasználó a válaszban visszakap, és így új oldal jelenik meg. Az úgynevezett Rich Internet alkalmazások esetén viszont a kliens egy HTML dokumentumot kap a szervertől a kérés válaszaként, amelyet egy konténerbe helyezve azonnal meg is jeleníthetünk.

Néhány példa, amire az AJAX technológiát használhatjuk:

- Valós időben validáhatjuk az adatok formai helyességét.
- Az adatok betöltését végezhetjük a háttérben, és csak akkor jelenítjük meg, ha már minden adat a rendelkezésünkre áll, vagy részletekben töltjük be az adatokat, így a felhasználó gyorsabban kap választ a kérésére.
- Az UI komponensek megváltoztatásához nem kell újratöltenünk az oldalt, így kényelmesebben kezelhetővé válik a felület a felhasználó számára.
- Bizonyos időközönként frissíthetjük az oldalon található adatokat anélkül, hogy a teljes oldalt újratöltenénk.

Az AJAX interakció tipikus folyamata

1. Egy esemény történik a kliens oldalon
2. Létrehozzuk az XMLHttpRequest objektumot, és felkonfiguráljuk
3. Az XMLHttpRequest objektum egy kérést intéz a szerver felé
4. A szerver feldolgozza a kérést, és elvégzi a szükséges módosításokat. Java alkalmazás esetén egy szervlet dolgozza fel a kérést.
5. A szervlet visszaküldi a kliensnek az eredményt tartalmazó választ.
6. Az XMLHttpRequest objektum meghívja a callback függvényt, és feldolgozza a választ.
7. A HTML DOM fáját frissítjük

Egyéb gondolatok

Bár a technológia sok hasznos tulajdonsággal bír, a felhasználók felé történő gyors interakció miatt már-már asztali alkalmazás-szerű programot kapunk, azért sok dolog jelent kihívást a folyamatok során.

- *Komplexitás*

  A szerver oldali fejlesztőknek meg kell tervezni, hogy milyen logika szükséges a megjelenítéshez, és össze kell tudniuk állítani a válaszhoz szükséges XML tartalmat. A HTML fejlesztőknek pedig meg kell ismerniük a JavaScript technológiát, amivel az AJAX hívásokat létre tudják hozni.
- *Az XMLHttpRequest objektum nem része a JavaScript standardnak*

  Így előfordulhat, hogy egyes böngészők nem támogatják a technológiát, vagy a működése eltérhet böngészőnként.
- *Biztonság*

  A kliens felé visszaadott adatokat biztonságos formában kell átadni, hiszen az oldalak tartalmát bárki megtekintheti, így vissza lehet élni a tartalommal.
- *Nyomkövetés*

  A fejlesztés során előálló hibákat nehezebb nyomonkövetni, hiszen mind a kliens oldalon, mind a szerver oldalon történnek változások.

Mint láthatjuk az AJAX technológia sok problémára nyújt megoldást, és a kliens felé történő interakció miatt egy emberbarátabb megjelenítést lehet a felhasználó felé prezentálni. A hátránya, hogy a fejlesztéséhez összetett ismeretekkel kell rendelkezni, nagy rendszerek esetén átláthatatlan lehet a kód, és mivel a technológia nem része a JavaScript szabványnak, egyes böngészőkön lehet nincs rá mód, hogy futtassuk az alkalmazásunkat.

GWT

A GWT

A Google Web Toolikt (GWT) egy fejlesztői eszközrendszer, melyet az összetett web alapú alkalmazások fejlesztéséhez, és optimalizálásához hoztak létre. A célja, hogy lehetővé tegye a nagy teljesítményű webalkalmazások hatékony fejlesztését anélkül, hogy a fejlesztőknek ismernie kellene a böngészők eltéréseit, vagy értenie kellene az XMLHttpRequest-ekhez és a JavaScripthez. A GWT nyílt forráskódú, mindenki számára elérhető eszközrendszer, amely teljesen ingyenes, és manapság már sok fejlesztő használja.

Fejlesztés GWT-vel

- *A kód megírása*

A GWT fejlesztői kertrendszer egy alap Java API-ból, és a hozzá tartozó prezentációs objektumokból, úgynevezett widgetekből áll. Ezek segítségével fejleszthetünk olyan AJAX technológián alapuló alkalmazásokat, melyeket miután a GWT keretrendszer által biztosított eszközzel magasfokon optimalizált JavaScript kódra fordítunk, minden olyan böngészőn futtathatunk, amelyek támogatják az AJAX technológiát. A GWT segítségével így gyorsabban és könnyebben készíthetjük el az alkalmazásunkat, hiszen a DOM fa manipulálása és az XHR kommunikáció felett egy magasabb absztrakciós szinten folyik a fejlesztés. Ráadásul a keretrendszer által biztosított komponenseken túl, a saját egyedi komponenseinket is létrehozhatjuk.

- *Hibakeresés*

Bár azt hinnénk, hogy az alkalmazás fejlesztés közben sok idő telhet el a hibák javításával, mint egy egyszerű AJAX alapú alkalmazás esetén, ez nem így van, mert a keretrendszer olyan eszközöket biztosít a számunkra, amelyekkel egyszerűen monitorozhatjuk az alkalmazásunkat, mintha egy asztali alkalmazást futtatnánk.

- *Optimalizálás*

A GWT keretrendszer több hatékony eszközt biztosít az optimalizált webalkalmazások létrehozásához. Az egyik a GWT fordító, ami átfogó optimalizálást végez a fejlesztő által előállított kódon, mint például eltávolítja a felesleges kódrészleteket, optimalizálja a szöveges részeket és még sok minden mást. A lefordítandó kódot részekre bonthatjuk, hogy ne egy nagy JavaScript kód forduljon az általunk megírt kódból, hanem több kisebb rész, így csökkenthetjük az alkalmazás indításához szükséges időt.

- *Futtatás*

Ha készen van az alkalmazás, a GWT fordító lefordítja az elkészült Java forráskódot optimalizált JavaScript kóddá, melyeket egyszerűen futtathatunk a legtöbb böngészőben.

JavaFX

A RIA (Rich Intenet Application)

A Rich Internet Alkalmazások (a továbbiakban RIA) olyan webalkalmazások, melyek szinte teljesen ugyanolyan tulajdonságokkal rendelkeznek, mint az asztali alkalmazások. Ezeket az alkalmazásokat általában egy böngésző önmagában nem képes futtatni, a működéshez szükség van egy kiegészítőre, melyet a böngésző kiegészítéseként telepíthetünk fel. Egy RIA indításakor, a futtatáshoz szükséges szoftver letölti az alkalmazást, ha már nem kell letölteni, akkor megnézi, hogy van-e újabb verzió az alkalmazásból, és ha található frissebb verzió, akkor frissíti azt, majd ellenőrzi az alkalmazást, hogy futtatható-e, és ha igen, akkor elindítja azt. Ez a legnagyobb különbség az ugyanilyen funkcionalitással bíró JavaScript alapú alkalmazásokhoz képest, mivel azok a böngésző beépített szolgáltatásait használják a megfelelő funkcionalitás eléréséhez.

A JavaFX

A JavaFX egy olyan platform, amellyel Rich Internet Alkalmazásokat fejleszthetünk, melyeket nem csak böngészők segítségével futtathatunk, hanem számos más eszközön is, mint például mobil telefonokon, vagy akár asztali alkalmazásként is használhatjuk. Az egyetlen követelmény ahhoz, hogy a JavaFX segítségével létrehozott alkalmazást futtassuk, hogy az eszközön, melyen az alkalmazást használni akarjuk, legyen Java futtató környezet. Mivel a legtöbb weboldal futtatásához szükség van a Java futtató környezetre, ezért általában nincs akadálya, hogy az alkalmazás böngésző alatt is fusson. A JavaFX technológia nyelve a JavaFX Script.

JavaFX Script

A JavaFX Script egy script nyelv, amely a Java platform része. A JavaFX Script örökölte a script nyelvek tulajdonságait, így gyorsan és könnyen lehet a segítségével

alkalmazásokat fejleszteni, és örökölte a Java nyelv által támogatott objektum orientált programozás tulajdonságait is, mely segítségével megbízható és újrahasznosítható komponenseket hozhatunk létre. A JavaFX Script használata könnyen megtanulható, és úgy hozták létre, hogy egyszerűen használhassunk benne Java kódban megírt komponenseket, így segítségével könnyedén hozhatunk létre Rich Internet Alkalmazásokat. A nyelv nagymértékben támogatja a grafikus komponensek fejlesztését, például a Java platformon elérhető Swing komponenseket egyszerűen integrálhatjuk a JavaFX Script kódunkba, és ezen túl további fejlett grafikus módszerek használatát is támogatja a nyelv.

Konklúzió

Bár a Rich Internetes Alkalmazások még nem túlságosan elterjedtek, elsősorban az online játékok terén használják őket, de egy teljesen új megközelítésbe helyezik a webes alkalmazásokat. A hátrányuk, hogy nem elég a futtatásukhoz egy böngésző, hanem egyéb kiegészítőkre is szükség van, és az erőforrás igényük a hagyományos webalkalmazásokhoz képest elég nagy, de cserébe a megjelenítés terén egy teljesen új élmény nyújtanak a felhasználók felé.

Összefoglalás

A Java alapú webalkalmazások fejlesztéséhez egy stabil alapot ad a szervlet technológia, amely egyéb eszközökkel kiegészítve lehetőséget teremt nagy és hatékony alkalmazások fejlesztéséhez. Szinte minden későbbi technológia a szervlet technológiára épül.

A JSP technológia ugyan megpróbálta megfordítani a fejlesztés menetét, és a megjelenítésért felelős JSP oldalakba elhelyezhettünk olyan kódot, ami az üzleti logikáért felelős, de a későbbiekben ezt a részt már nem használták. A technológia mégis megmaradt, hiszen segítségével egyedi tag-eket használhatunk az oldalon belül, amelyek funkcionalitását a szerver oldalon határozhatjuk meg. A JSP oldalak így megmaradtak a megjelenítési rész előállításához.

A későbbiekben a technológiák olyan eszközöket biztosítottak, melyekkel el lehetett választani a fejlesztés egyes részeit, ezzel könnyebbé és egyszerűbbé tették a webalkalmazások létrehozását. Továbbá beépített komponenseket biztosítanak, melyeket kiterjeszthetünk, létrehozva a saját alkalmazásunkhoz szükséges egyedi komponenseket. Támogatják az olyan funkciókat, mint a többnyelvűség kezelése, a felhasználó által bevitt adatok validálása, és egyéb olyan részeket tartalmaznak, amelyekkel egyszerűen fejleszthetünk webalkalmazásokat anélkül, hogy minden egyes nehézséget magunknak kelljen megoldani.

Az újabb technológiák inkább a felhasználó felé való interakciót korszerűsítik, és plusz lehetőségeket biztosítanak a felhasználó felé mutatott megjelenítés átláthatóbbá tételére. Egyre inkább elviszik a webalkalmazások funkcionalitását az asztali alkalmazások funkcionalitása felé. Bár a fejlesztés nem annyira egyszerű, mint asztali alkalmazásoknál, de a megjelenésükben és a kezelhetőségükben nagyban hasonlítanak az asztali alkalmazásokra. Valószínűleg a jövőben még több ilyen alkalmazás készül majd, hiszen a gépek teljesítménye növekszik, így egyre több mindent lehet a kliensre bízni, és csak a szükséges dolgokat kell a szerveren végezni.

Minden technológia támogatja az absztrakciót és az újrafelhasználhatóságot, ezzel csökkentve a fejlesztési időt. Jól használható komponenseket definiál, melyek használata egyszerű, és belőlük újabb, saját részre jól használható komponenseket lehet létrehozni, ezzel is gyorsítva a fejlesztést. Mindegyik törekszik rá, hogy a megjelenítés ne függjön az üzleti logikától, és hogy az alkalmazásban lévő folyamatok átláthatóak legyenek.

A felsorolt technológiák közül bármelyik alkalmas arra, hogy segítségével egy jól használható webalkalmazást készítsünk, elsősorban az alapján érdemes választani, hogy milyen funkciókat kell megvalósítani a felhasználó felé.

Irodalomjegyzék

[1] Brian Basham, Kathy Sierra, Bert Bates - Szervletek és JSP, O'Reily Media 2008

[2] Hans Bergsten - JavaServer Pages, O'Reily Media, 2003

[3] Hans Bergsten - JavaServer Faces, O'Reily Media, 2004

[4] Nyékiné Gaizler Judit - J2EE útikalauz Java programozóknak, ELTE TTK, 2002

[5] http://www.wikipedia.org/

[6] http://java.sun.com/products/jsp/

[7] http://struts.apache.org/