Debreceni Egyetem
Informatikai kar

Oracle alapú rendezvényszervező portál

Témavezető:
Dr. Juhász István
Adjunktus

Készítette:
Fodor István Balázs
Programtervező informatikus (B.Sc.)

Debrecen
2007

Az szakdolgozat témájának rövid ismertetése

Szakdolgozatom témájaként egy rendezvényszervezési webes alkalmazás elkészítését és annak leírását tűztem ki célul.

Az alkalmazás egy rendezvényszervezőket, rendezvények iránt érdeklődőket támogató portál. Feladata, hogy a rendezvényszervezők és a rendezvények iránt érdeklődők közt a kommunikációt segítse, bő tájékoztatást biztosítson az oldalra látogatóknak. Könnyű kezelhetősége révén, mely nem igényel semmilyen informatikai tudást, széles körben elérhetőek vele az emberek. A portál - az egyszerű böngészés mellett - lehetőséget biztosít események információinak publikálására és a portál adminisztrációs feladatainak elvégzésére is, és mindezt könnyen kezelhető, webes felületen.

Tartalomjegyzék

I. Áttekintés I-1
I.1 A szoftver rövid leírás I-1
I.2 Fogalmak I-2
I.3 A portállal szemben előzetesen támasztott követelmények I-4
I.3.1 Felhasználói követelmények I-4
I.3.2 Rendszerkövetelmények I-4
II. Felhasználói dokumentáció II-5
II.1 Telepítési dokumentáció II-5
II.1.1 Telepítési előfeltételek II-5
II.1.2 Telepítési fileok II-5
II.1.3 Telepítési útmutató II-5
II.2 Bevezető kézikönyv II-7
II.2.1 Áttekintés – Általános leírás II-7
II.2.2 Látogatói funkciók II-9
II.2.3 Feltöltési funkciók II-10
II.2.4 Adminisztrátori funkciók II-12
III. Megvalósítás III-15
III.1 Konkrét példák III-15
III.1.1 Design III-15
III.1.2 Rendezvények megjelenítése III-15
III.1.3 Események laponként történő megjelenítése III-16
III.1.4 Adatok validálása III-17
IV. Alkalmazott Technológiák IV-19
IV.1 Áttekintés IV-19
IV.2 Oracle technológiák IV-19
IV.2.1 Áttekintés IV-19
IV.2.2 Az Oracle XML DB IV-21
IV.2.3 Az Oracle Text lehetőségei IV-22
IV.3 Java technológiák IV-24
IV.3.1 A Java EE áttekintése IV-24
IV.3.2 A Java EE 5 platform technológiái IV-25
IV.3.3 A Glassfish alkalmazás-szerver IV-28
V. Függelék V-29

VI. Irodalomjegyzék VI-34

I. Áttekintés

I.1 A szoftver rövid leírás

A szoftver egy **Oracle** adatbáziskezelő-rendszer és **Glassfish** alkalmazásszerver alapú, rendezvényszervező portál. A portálra bizonyos kitüntetett felhasználók rendezvények hirdetéseit tölthetik föl. A rendezvények adatait a legkisebb részletekig nyilvántartja a rendszer. A nyilvántartott adatok modellje dinamikusan változhat a szoftver későbbi evolúciójával. A megjelenítési alrendszert nem érinti az adatmodell bővülése, így könnyedén bővíthető a szoftver.

Egy tartalomszolgáltató oldal talán legfontosabb funkciója a keresés. A szoftver erre több módot is tartalmaz. A legalapvetőbb keresési lehetőség a „gyors keresés" vagy „quick search". Ez egy általános tartalomkeresési mód, minden komolyabb webes tartalomszolgáltató alkalmazásnál megjelenik. Ereje abban rejlik, hogy egy nagyon egyszerű keresési űrlapot használ. Az űrlap egyetlen szövegbeviteli eleme egy szövegmező, amiben a szavakat, karaktersorozatokat adunk meg. A kereső azon rekordokat adja vissza, amelyek tartalmazzák a megadott karaktersorozatokat. Jelen esetben, a találatok megjelenítése feltöltési idő alapján, csökkenő sorrendben történik. Egy másik keresési opció, hogy a rendezvények időpontja alapján keresünk. Ekkor egy időintervallumot kell megadni. Az adott időintervallumban megrendezésre kerülő eseményeket kapjuk keresési eredményként.

A szoftver a látogatóknak lehetőséget nyújt hírlevélre történő feliratkozásra. Ennek módja az oldalsávon megjelenő „feliratkozás hírlevélre" nevű űrlap. Ezután a feliratkozott látogató e-mailben értesítést kap minden, az oldalra frissen feltöltött eseménnyel kapcsolatban.

A portálra a regisztrált felhasználók a rendezvények adatait webes felületen keresztül tölthetik föl. A feltöltéseket, illetve egyéb tartalommódosító folyamatokat mindig felhasználói azonosítás előzi meg.

Körlevél küldésére az adminisztrátorok jogosultak. A körlevelet webes felületen kell összeállítani és elküldeni. Az üzenet küldhető a hírlevélre feliratkozóknak és a feltöltők csoportjának is. A felhasználói fiókok kezelését az adminisztrátorok webes felületen végezhetik.

I.2 Fogalmak

A fejezetben a rendezvényszervező portál körében megjelenő fogalmak szerepelnek.

1. Rendezvény: A portálban megjelenő rendezvények a valós életben megrendezésre kerülő rendezvényeket szimbolizálják, azokkal kapcsolatban az összes fontos információt tartalmazzák. Ezek az információk:
    - A rendezvény neve. Például: Debreceni Virágkarnevál
    - A rendezvény kezdetének időpontja.
    - A rendezvény helyszíne.
    - A rendezvény jellege. Ez lehet például előadás vagy koncert. Célja, hogy kulcsszavas keresést segítsék. Például, ha valaki a „koncert" szóra keres rá, akkor az összes olyan eseményt is megkapja, aminek a „jelleg" mezőjében szerepel a „koncert" szórészlet.
    - Rövid bemutató szöveg. Az esemény rövid leírása, csak a legfontosabb információkat tartalmazza. Szerepe akkor van, amikor több rendezvény információt jelenítünk meg.
    - Bemutató szöveg. Az esemény hosszú leírása. Minden fontos információt tartalmaz. Akkor látható, amikor a felhasználó egy adott esemény összes információját megjeleníti.
    - Jelentkezési információk:
        i. Szükséges-e jelentkezni?
        ii. Hol lehet jelentkezni? (weboldal, telefon, cím, e-mail)
    - Jegyvásárlás információk:
        i. Szükséges-e jegyet venni?
        ii. Hol lehet jegyet vásárolni? (weboldal, telefon, cím, e-mail)

2. Látogató: Olyan emberek, akik azért látogatják a portált, hogy onnan információt szerezzenek eseményekről és rendezvényekről, illetve azokra jelentkezzenek, jegyet vásároljanak. A látogatók böngészhetnek az események között, azokon kereséseket hajthatnak végre.

3. Regisztrált felhasználó: A feltöltők csoportja. Egy kisebb létszámú csoport, mint a látogatóké. Amellett, hogy a látogatók lehetőségeivel is rendelkeznek, lehetőségük van rendezvényekről információt feltölteni. Ehhez be kell jelentkezniük az oldalra. A bejelentkezéshez szükséges, hogy előtte az oldal regisztrált felhasználói legyenek. A regisztráció nem nyílt, és csak egy portáladminisztrátor regisztrálhat új felhasználókat.

4. Portáladminisztrátor: A regisztrált felhasználók egy szűkebb részhalmaza. Ez a kör tartalmazza a legkevesebb felhasználót. Feladatuk amellett, hogy a feltöltők feladatkörét is elláthatják, az, hogy a portál működtetését végzik. Ez jelenti az új felhasználók regisztrálását, régebbiek módosítását vagy törlését, illetve körlevelek küldését.

5. Regisztrálás: A leendő felhasználó értesíti az oldal egy adminisztrátorát regisztrálási szándékáról. Az adminisztrátor, ha az illető megfelel, létrehoz egy felhasználói azonosítót.

6. Bejelentkezés: Regisztrálás után a felhasználó rendelkezik egy felhasználói névvel, és egy jelszóval. A portál védett részeihez való első hozzáférést bejelentkezés előzi meg.

Bejelentkezéskor meg kell adni a felhasználói azonosítót és a jelszót.

7. Esemény feltöltés: A regisztrált felhasználóknak és a portáladminisztrátoroknak lehetőségük van rendezvényekkel kapcsolatos információk elhelyezésére a portálon. Ez egy űrlap segítségével történik, amelyben megadják az összes feltölteni kívánt információt. Ez egy védett része a portálnak.

8. Felhasználók menedzselése: Az adminisztrátorok jogkörébe tartozó tevékenységek. Felhasználó létrehozása, törlése és módosítása tartozik ebbe a feladatkörbe. Mindegyik feladat webes felületen hajtható végre. A felhasználók menedzseléséhez köthető funkciók védettek, első használatukat azonosítás előzi meg.

9. Körlevél: Olyan levél, melyet vagy automatikusan a szoftver vagy egy adminisztrátor küld. Címzettjei lehetnek azok, akik feliratkoznak a körlevél szolgáltatásra, vagy regisztrálva vannak az oldalon mint feltöltők vagy adminisztrátorok. Az adminisztrátor az általa küldött körlevél címzetti köreit meghatározhatja: vagy körlevélre feliratkozók vagy a regisztrált felhasználók vagy mindkét kör.

I.3 A portállal szemben előzetesen támasztott követelmények

I.3.1 Felhasználói követelmények

1. A portálnak képesnek kell lennie a rendezvények és események adatainak részletes tárolására és megjelenítésére.
2. A rendszer a telepítést követően teljes mértékben webes felületen kell, hogy kezelhető legyen.
3. A regisztrált felhasználók és alkalmazottak adatait biztonságosan kell őrizni.
4. A portál nem nyilvános komponenseit felhasználói azonosítással kell védeni.
5. Új felhasználót csak adminisztrátorok regisztrálhatnak.
6. A regisztrált felhasználók alapesetben rendezvényfeltöltési joggal rendelkeznek.
7. A regisztráltak egy szűkebb csoportja portáladminisztrátori jogkörrel is rendelkezik.
8. Az adminisztrátorok a feltöltés mellett egyéb feladatokat is ellátnak. Ezek: felhasználó regisztrálása, felhasználói adatok módosítása és felhasználó törlése.
9. A portál lehetőséget kell, hogy biztosítson keresésre az események közt. A keresés történhet kulcsszavak illetve időintervallum alapján.

A felhasználói követelményeket a függelék 1. ábrája mutatja.

I.3.2 Rendszerkövetelmények

1. A rendezvények adatait XML formátumban kell tárolni úgy, hogy azokon szöveges keresést lehessen végrehajtani.
2. Az alkalmazás Oracle alapú, tehát az SQL szabványon túl egyéb, Oracle-specifikus szolgáltatásokat is igénybe kell vennie.
3. Körlevél küldéséhez rendelkezni kell egy SMTP protokollt támogató levelező szerverrel.
4. Platform:
    - Operációs rendszer: Platform független
    - **Glassfish**, Java EE 5 certified alkalmazásszerver
    - **Oracle Database 10g, Release 2**
    - Tetszőleges SMTP szerver

II. Felhasználói dokumentáció

II.1 Telepítési dokumentáció

II.1.1 Telepítési előfeltételek

1. A kiválasztott gépen telepítve kell, hogy legyen egy **Glassfish** alkalmazásszerver példány, egy **Oracle Database 10g, Release 2** példány és egy tetszőleges SMTP kiszolgáló.
2. Telepítve kell, hogy legyen egy 6-os verziójú **Java Virtual Machine** példány.
3. Egy tetszőleges web-böngésző megléte.

II.1.2 Telepítési fileok

1. `RendezvenyPortal.ear`, Enterprise Application Archive fájl
2. `RendezvenyPortal.sql`, Oracle SQL fájl

II.1.3 Telepítési útmutató

1. A **Glassfish** szerver feltelepítése után indítsuk el a szervert. Ha már létezik domain-ünk, akkor indítsuk azt el, vagy hozzunk előbb létre egyet. Ezeket Windows alatt megtehetjük az `glassfish/bin/asadmin.exe` utility segítségével.
2. Indítsuk el az **Oracle Database 10g** –t, a **TNS listenert** és az **Oracle Enterprise Managert**.
3. Lépjünk be az **Enterprise Managerbe** SYS felhasználói névvel.
4. Hozzunk létre egy új felhasználót. A neve tetszőleges lehet, most legyen `User`.
5. A `User` nevű felhasználónak adjuk meg a következő jogosultságokat:
   - CREATE ANY TABLE
   - CREATE ANY VIEW
   - CREATE ANY PROCEDURE
   - CREATE ANY TYPE
   - CREATE ANY INDEX
6. SQL*Plus környezetben jelentkezzünk be a `User` accountra, és futtassuk le a `RendezvenyPortal.sql` fájlt, lépjünk ki az SQL*Plusból.
7. Lépjünk a **Glassfish** adminisztrációs felületére.
8. A „Resources” menüpontban válasszuk ki a „JDBC” fület, azon belül a „Connection Pools” fület. Ott kattintsunk a jobb felső sarokban lévő „New” feliratú gombra.
9. Az első megjelenő űrlapon a következő adatokat adjuk meg, majd kattintsunk a „Next”-re:
   - Name: oracle-thinPool
   - Resource Type: javax.sql.DataSource

1. ábra Glassfish, Resources menüpont

- Database Vendor: Oracle

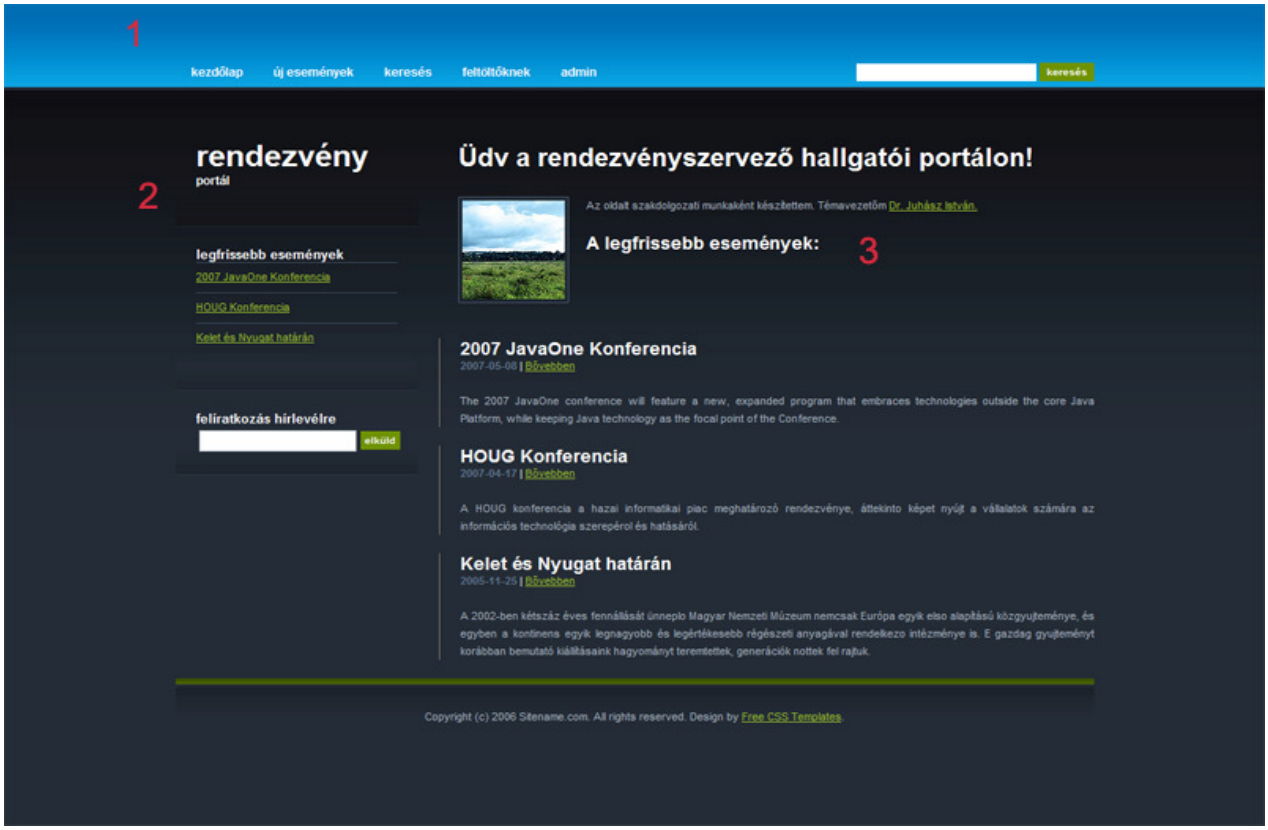
10. A megjelenő képernyő alsó részében az „Additional Properties” táblázat elemeit jelöljük ki, az URL, Pasword és User kivételével, majd kattintsunk a „Delete Property” feliratú gombra.
11. Az így kapott űrlap elemeit a következő módon töltsük ki, majd kattintsunk a „Finish” gombra:
    - User: Az adatbázis-felhasználónk neve. Most `User`.
    - Password: A `User` nevű felhasználó jelszava az adatbázishoz.
    - URL: Ide egy „JDBC Connection Stringet” kell megadni. jdbc:oracle:thin:@<HOST>:<PORT>:<SID>
        - HOST: ha azonos gépen fut a **Glassfish** és az **Oracle**, akkor „localhost”, különben az adatbázisszerver URL-je.
        - PORT: a **TNS Listener** által figyelt port. Alapesetben 1521.
        - SID: (System Identifier) Az **Oracle** példányunk azonosítója.
12. A most létrehozott Connection Pool tárolja az adatbázis kapcsolatainkat. A kapcsolatot leellenőrizhetjük, ha a megjelent képernyőn kiválasztjuk az „oracle-thinPool”-t, és a „Ping” gombra kattintunk.
13. A „Resources” menüpontban válasszuk ki a „JDBC” fület, azon belül a „JDBC Resources” fület.
14. Itt kattintsunk a „New” feliratú gombra, a megjelenő űrlapot a következőképpen töltsük ki, majd kattintsunk az „Ok” gombra:
    - JNDI Name: oracle
    - Pool Name: oracle-thinPool
15. A „Resources” menüpontban válasszuk ki a „JDBC” fület, azon belül a „JavaMail Sessions” fület.
16. Hozzunk létre egy új „JavaMail Sessiont” a „New” feliratú gombra kattintással.
17. A kapott űrlapot a következő módon töltsük ki, majd kattintsunk az „Ok” gombra:
    - JNDI Name: mail
    - Mail Host: localhost
    - Default User: a SMTP szerveren lévő felhasználói fiókunk
    - Default Return Address: Az e-mail címünk
18. A JavaMail alapértelmezésként az SMTP szolgáltatást a default SMTP porton, a 25-ösporton próbálja elérni. Ezért az SMTP kiszolgálót úgy kell konfigurálni, hogy az a 25-ös poron várja a kéréseket.
19. Ezzel az alkalmazásszerver beállítása kész.
20. A `RendezvenyPortal.ear` fájlt másoljuk be a glassfish\domains\<domainnév>\autodeploy könyvtárba. Egy pár másodperc elteltével az adott gépről elérhető lesz az alkalmazás a `http://localhost:<PORT>/RendezvenyPortal-war` URL-en. A PORT a **Glassfish** http kiszolgálója által figyelt portot jelöli.

II.2 Bevezető kézikönyv

II.2.1 Áttekintés – Általános leírás

Ebben a fejezetben a portál minden funkciója és azok használata van bemutatva. Az első pontban a portál azon tulajdonságai és funkciói kerülnek bemutatásra, amelyek általánosan, minden felhasználói körhöz kötődnek. Ezek közé tartozik a felhasználói felület elemeinek ismertetése.

Az oldalra való érkezéskor a felhasználó a portál főoldalára kerül. Innen lehet elérni a további funkciókat.



2. ábra Főoldal

Az oldal minden esetben 3 részre különíthető el.

1. A menüsáv
2. Az oldalsáv
3. Az oldal fő területe

A menüsáv két funkcióval rendelkezik. Az egyik a navigálás. Az oldal főbb funkciói innen érhetőek el, egy menüpont kiválasztásával. A menüpontok sorban:

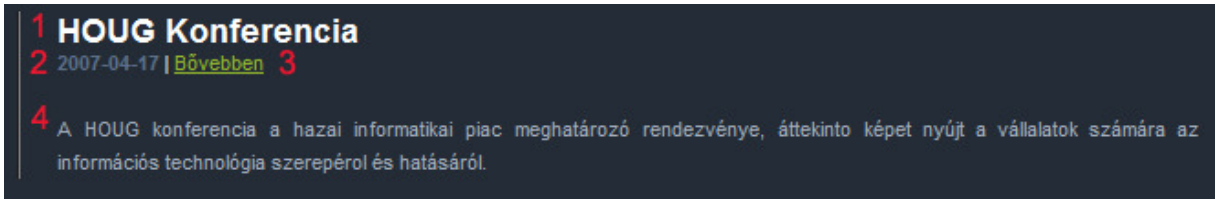
1. Kezdőlap: A képen látható oldalra visz vissza.
2. Új események: A legfrissebben feltöltött eseményekhez jutunk.
3. Keresés: A kereső oldalára jutunk.

4. Feltöltőknek: A bejelentkező oldalra, vagy ha már megtörtént, a feltöltési oldalra jutunk.
5. Admin: A bejelentkező oldalra, vagy ha már megtörtént, az adminisztrációs oldalra jutunk.

A másik funkciója a keresés. Itt található a „gyorskeresés” űrlapja. Ez a jobb felső sarokban látható. A fehér részre kell beírni a kulcsszavakat, és a zöld gomb lenyomása után megkapjuk az eredményeket.

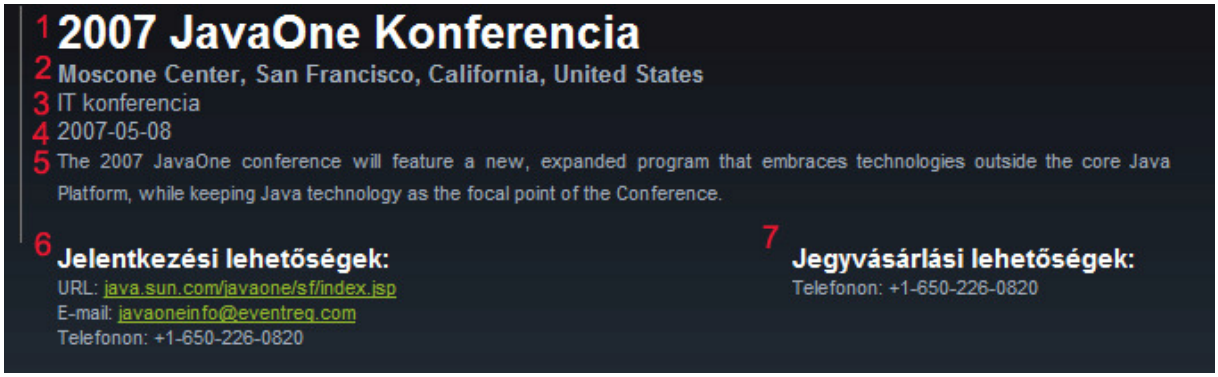
Az oldalsáv szintén két funkcióval bír. Az egyik egy olyan hiperlink gyűjtemény, amely a legutoljára feltöltött három esemény ismertetőjére mutat. Ez a szolgáltatás lényegében semmi plusz lehetőséget nem nyújt a felhasználónak, mivel itt mindig azok a rendezvények vannak felsorolva, amelyek a főoldalon is megtalálhatóak. Ez lehetőséget biztosít bármely felhasználónak, hogy az aktuális műveletet megszakítva a legfrissebb események információit megtekintse. Ez azért kényelmes, mert az oldalra történő feltöltés és böngésző oldalfrissítése után a felhasználó rögtön látja az új eseményt, nem kell külön böngésznie. A másik itt található funkció a hírlevélre történő feliratkozás. Ezzel az űrlappal lehet feliratkozni a körlevelekre. Körlevelet két esetben kap a feliratkozott: ha egy adminisztrátor körlevelet akar a feliratkozottaknak küldeni, vagy ha valaki új rendezvénnyel kapcsolatos információkat tölt fel az oldalra.

Az oldal előbbi két területe statikus abban az értelemben, hogy maguk vagy egyáltalán nem változnak, bármely funkcióját használjuk is a rendszernek, vagy ha az oldalsáv hiperlink gyűjteményét nézzük, akkor csak feltöltések esetén történik változás. Ezzel szemben a portál fő területe mindig az aktuálisan használt funkcióhoz kötődik. Emiatt a továbbiakban az oldalról készült képeknek csak ezt a részét közlöm. Az oldalon az események kétféleképpen jelenhetnek meg: rövid vagy hosszú formában.



3. ábra Rendezvény, rövid formátum

A rendezvények rövid formájának elemei:
1. A rendezvény neve.
2. A rendezvény kezdetének dátuma.
3. Egy olyan link, ami a rendezvény hosszú megjelenítési formájához vezet.
4. A rendezvény rövid leírása.



4. ábra Rendezvény, hosszú formátum

A rendezvények hosszú formájának elemei:

1. A rendezvény neve.
2. A rendezvény helyszíne.
3. A rendezvény jellege.
4. A rendezvény kezdetének időpontja.
5. A rendezvény hosszú leírása.
6. Ha kell előre jelentkezni, arról itt található információ, vagy itt lehet megtenni. A jelentkezési lehetőségek formái lehetnek: weboldal cím, e-mail cím, telefonszám és pontos cím.
7. Ha a rendezvényre csak jeggyel lehet belépni, arról itt lehet információt szerezni, vagy itt lehet azt megvásárolni. Ennek formái szintén lehetnek weboldal cím, e-mail cím, telefonszám, vagy pontos cím.

II.2.2 Látogatói funkciók

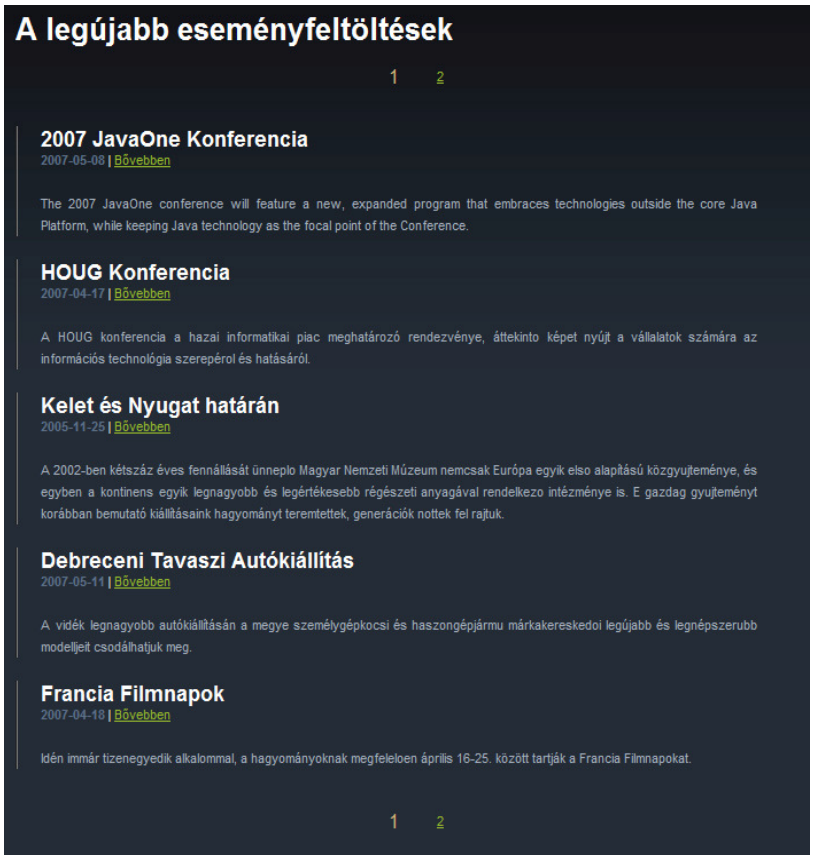
A látogató célja, hogy rendezvényekről információkat szerezzen. Ezt a már korábbiakban leírt módokon teheti meg: böngészés a főoldalon, böngészés az új események közt és keresés alapján történő böngészés. A látogató interakcióinak összefoglalása a függelék 1. ábrája. A következőkben ezek leírását adom meg.

Böngészés az események közt

A főoldalon történő böngészés során a kezdőlapon található rendezvények közül választ ki egyet a felhasználó, majd a „bővebben” linkre kattintva a többi információt is elolvassa. Ha a felhasználó vissza akar térni a főoldalra, azt megteheti az esemény bal felső sarkában megjelenő „vissza” gombbal vagy a menü „kezdőlap” feliratú gombjával.

Az új események közti böngészést a menü „új események” feliratú gombjára történő kattintással kezdheti meg a felhasználó. Az ekkor megjelenő oldalt az 5. ábrán láthatjuk:

Az oldalon a legutoljára feltöltött öt esemény rövid



5. ábra Új események

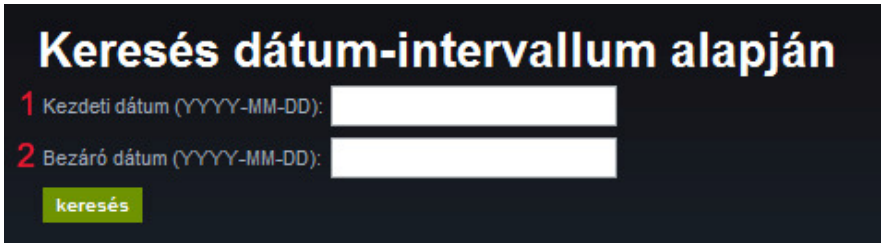
alakja látható. Az oldal tetején és alján lévő számok oldalakat jelölnek. A számok mindegyike egy hiperlink, amely az általa jelölt oldalra vezet. A második oldal a legutoljára feltöltött öt képet megelőzően legutoljára feltöltött öt képet jelöli. Így az n. oldal az ((n – 1) * 5) legutoljára feltöltött rendezvényt megelőzően feltöltött utolsó öt esemény jelöli. Az oldalak száma dinamikusan változik, annak megfelelően, hogy hány eseményről tartalmaz információt az portál. Így erről az oldalról elérhető a portál teljes tartalma.

Keresési funkciók

Az oldal két keresési funkcióval rendelkezik. A keresések találatait mindig rövid formában kapjuk meg.

A gyorskeresés használatakor a felhasználó megad kulcsszavakat, és a megadott szavakat vagy szótöredékeket tartalmazó rendezvényeket visszaadja. A kereső a rendezvény minden tulajdonságát lefedi, azaz a keresett kulcsszavak a teljes esemény minden részére illesztve lesznek.

A dátum-intervallum alapján történő kereséshez a menü „keresés” feliratú gombjára kell kattintani. Az ilyen jellegű keresés azt teszi lehetővé, hogy egy időintervallumban megrendezendő eseményekre keressünk rá. A „keresés” gombra kattintva a 6. ábrán látható képernyő jelenik meg. A képernyőn egy űrlapot találunk, amely két szövegbeviteli elemmel rendelkezik.



6. ábra Keresés dátum alapján

1. Itt az időintervallum alsó határát adhatjuk meg YYYY-MM-DD formában.
2. Az időintervallum felső határát adhatjuk meg, szintén YYYY-MM-DD formában.

Ha üresen hagyjuk valamelyik elemet, akkor az intervallum nem véges. Ekkor, ha a bezáró dátumot adtuk meg, akkor az összes azt megelőző rendezvényt, ha pedig a kezdeti dátumot, akkor az összes azt követő rendezvényt kapjuk meg. Ha üresen hagyjuk mindkettőt, akkor az összes rendezvényt visszakapjuk. A „keresés” feliratú gomb lenyomása után megkapjuk a keresés eredményét. Ha nem hagyjuk üresen valamelyik sort, de azt nem a megfelelő formátummal töltjük ki, akkor a keresés gomb lenyomása után erre az oldalra kerülünk visszairányításra.

A rendezvény hosszú információiról a látogató továbbléphet más oldalakra, vagy visszatérhet a kereséshez és a böngészéshez.

II.2.3 Feltöltési funkciók

A rendezvényfeltöltési funkciókat a regisztrált felhasználók illetve a portáladminisztrátorok jogosultak használni.

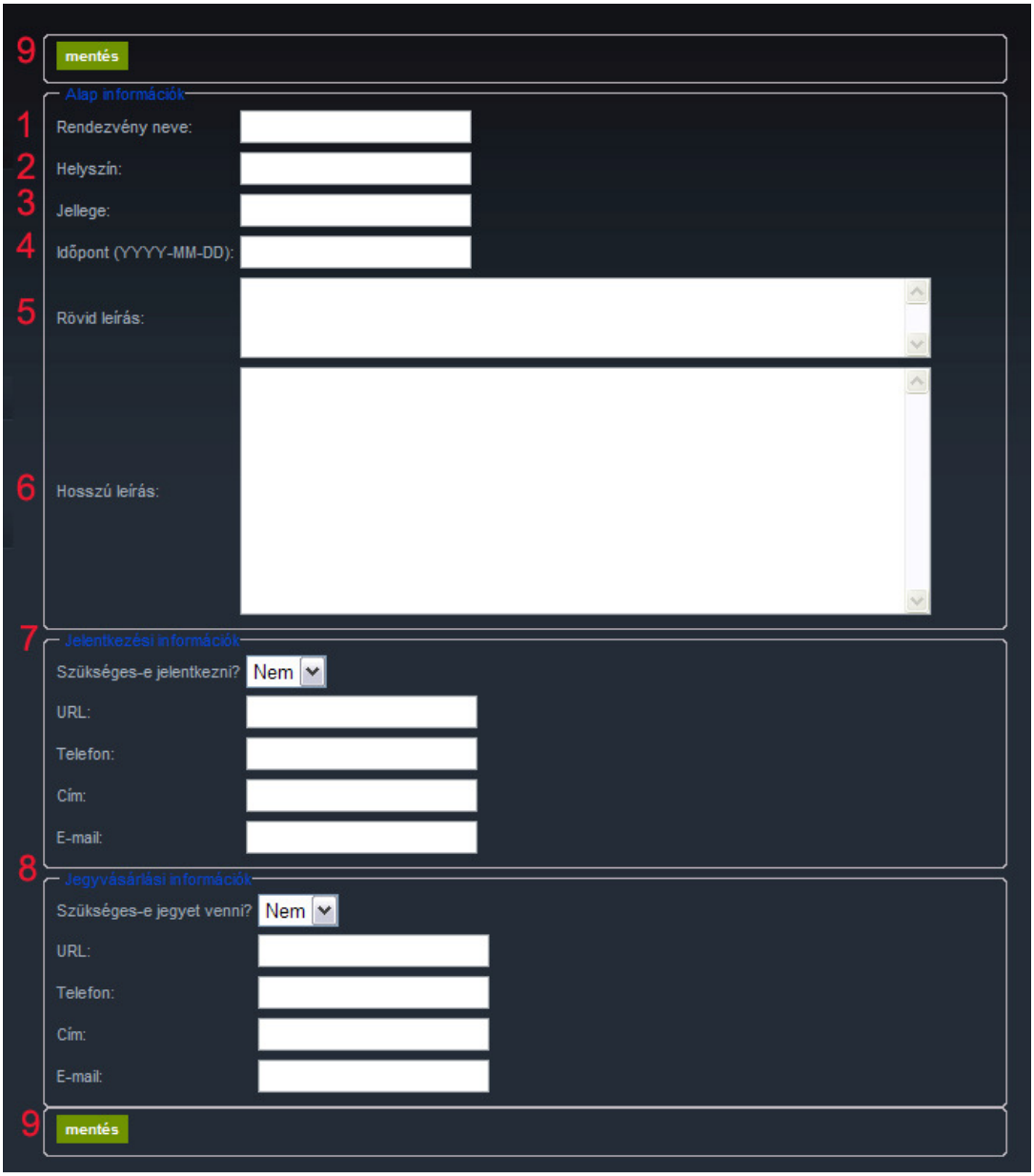
A feltöltés védett funkció, ezért

7. ábra Bejelentkezés

csak bejelentkezés után használható. A bejelentkezéshez nincs külön menüpont rendelve. Ha valaki a menü „feltöltőknek”, illetve „admin” feliratú gombjaira kattint és nincs bejelentkezve, akkor automatikusan a bejelentkezési oldalra kerül. A bejelentkezési oldal képe a 7. ábrán látható.

A felhasználói név és a jelszó megadása és a „bejelentkezés” gomb lenyomása után, ha helyesek a megadott adatok, a regisztrált felhasználók a feltöltési oldalra, az adminisztrátorok az adminisztrációs felületre kerülnek. Helytelen adatok esetén visszairányít a bejelentkezési oldalra. Sikeres bejelentkezés után a menü a „kilépés” elemmel bővül.

8. ábra Kibővült menü



9. ábra Feltöltési oldal

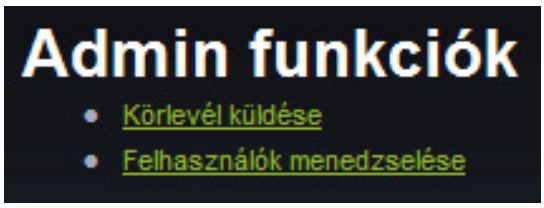
A feltöltési oldalon egy űrlap található. Ez szolgál a rendezvény adatainak megadására. Az űrlap elemei sorrendben:

1. A feltöltendő rendezvény neve.
2. A rendezvény helyszíne.
3. A rendezvény jellege.
4. A rendezvény kezdetének időpontja.
5. A rendezvény rövid leírása.
6. A rendezvény hosszú leírása.
7. A jelentkezési információk.
8. Jegyvásárlási információk.
9. Mentés gomb. Erre kattintva föltöltheтjük a megadott adatokat.

Ha a rendezvény kezdeti időpontját nem a megfelelő YYYY-MM-DD formátumban adjuk meg, akkor visszakerülünk a feltöltési oldalra, egyébként pedig a főoldalra. Ha nem akar a felhasználó több rendezvényt feltölteni, akkor a „kilépés” gombbal kilép, és elhagyja a portált.

II.2.4 Adminisztrátori funkciók

Az adminisztrátori funkciók az „admin” gombra kattintás után érhetőek el. Ha nem vagyunk bejelentkezve, akkor az előbbiekben ismertetett módon megtörténik a bejelentkezés, és az adminisztrációs felületre érkezünk. Ennek képe a 10. ábrán látható.



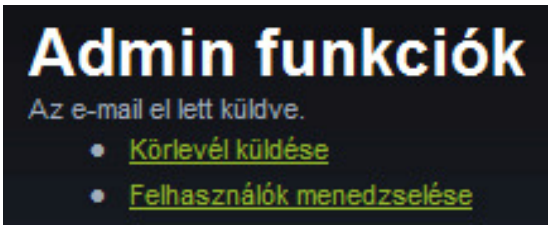
10. ábra Admin funkciók

Körlevél küldése

Az adminisztrációs felület két feladatkörhöz nyújt hozzáférést, körlevél küldéséhez és a felhasználók menedzseléséhez. A „körlevél küldése” hiperlinkre kattintva a „körlevél küldése” oldalra jutunk. Az oldal tartalma egy űrlap, amellyel megszerkeszthetjük a körlevelet és kijelölhetjük a címzetteket. A funkció használatához szükséges, hogy az SMTP kiszolgáló be legyen konfigurálva a **Glassfishben.** Az űrlap elemei sorrendben:

11. ábra Körlevél küldése

1. A szövegbeviteli terület, ahova a levelet írhatjuk.
2. Ha a checkboxot bejelöljük, a levelet megkapja mindenki, aki az oldalsávon lévő „feliratkozás hírlevélre” űrlappal feliratkozott erre a szolgáltatásra.
3. Ha a checkboxot bejelöljük, minden feltöltő és portáladminisztrátor megkapja a körlevelet.
4. Az „elküld” feliratú gomb lenyomása után a portál elküldi a leveleket. Ha mindkét checkbox be van jelölve, mindkét kör megkapja a levelet. Ezután az adminisztrációs felületre térünk vissza. A felületen üzenetet kapunk arról, hogy sikeres volt-e a levélküldés. Ha nem adtunk meg címzetteket, akkor a portál ezt jelzi, ha az SMTP kiszolgálóval volt gond, arról is értesítést ad. Sikeres levélküldés esetén a 12. ábrán látható képet kell, hogy kapjuk.

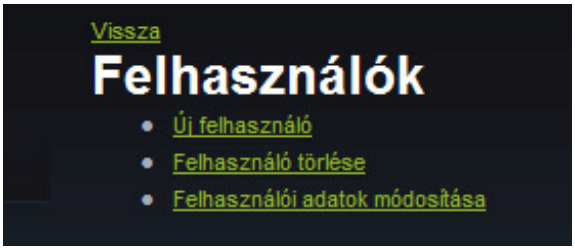


12. ábra Sikeres e-mail küldés

Új felhasználó lérehozása

A „felhasználók menedzselése” hiperlinkre kattintva a 13. ábrán látható menüt kapjuk.

Az „új felhasználó” hiperlinkre kattintva a 14. ábrán látható űrlapot kapjuk. Az űrlapon az alábbi adatokat kell megadnunk:



13. ábra Felhasználók menedzselése

1. Felhasználó vezetékneve.
2. Felhasználó keresztneve.
3. Felhasználói azonosító. A felhasználó ezzel a névvel léphet be az oldalra.
4. Jelszó. A felhasználói azonosítóhoz tartozó jelszó.
5. A felhasználó e-mail címe. Az e-mail címnek szabályosnak kell lennie.
6. A felhasználó szerepköre. Lehet „Felhasználó”, vagy „Adminisztrátor”.
7. Az adatok elküldésére szolgáló gomb.

Mindegyik mezőt ki kell tölteni. Ha már létezik ilyen felhasználói azonosítóval rendelkező felhasználó, akkor az „elküld” feliratú gomb megnyomása után erről kapunk üzenetet. Ha az e-mail helytelen, visszakerülünk a „Felhasználó létrehozása” űrlaphoz.

14. ábra Felhasználó létrehozása

Felhasználó törlése

A 13. ábrán látható menü következő pontja a „Felhasználó törlése”. A megadott hiperlinkre kattintva a 15. ábrán látható legördülő listát kapjuk. A listából ki kell választani a törlendő személy nevét. Ezután a „megmutat” feliratú gombra kattintva megjelennek a kiválasztott felhasználóhoz tartozó felhasználói adatok és a „töröl” feliratú gomb. Erre kattintva a megjelenített felhasználó törlődik. Törlés után az adminisztrációs főoldalra kerül vissza a felhasználó.

15. ábra Felhasználó törlése I.

16. ábra Felhasználó törlése II.

Felhasználói adatok módosítása

A 13. ábrán látható menü utolsó pontja a „Felhasználói adatok módosítása”. Erre kattintva a 15. ábrán látható oldallal megegyező kinézetű és működési elvű oldalra jutunk. A név kiválasztása és a „megmutat” feliratú gombra kattintás után megjelenő felhasználói adatok alatt a „módosít” feliratú gomb áll. Erre kattintva a 17. ábrán látható űrlaphoz jutunk, amely elemeit tekintve megegyezik a 14. ábrán látottal. Az űrlap automatikusan kitöltődik. Itt is érvényes, hogy minden adatot meg kell adni. A jelszó mező ez alól kivételt képez. Ha oda nem adunk meg semmit, akkor nem változik meg a jelszó. Az egyetlen adat, amin nem lehet változtatni, az a felhasználói azonosító. Az is megjelenik az űrlapon, de nem módosítható. Hibás adatok esetén visszakerülünk a módosítási oldalra.

17. ábra Felhasználói adatok módosítása

III. Megvalósítás

A fejezetben a szoftver bizonyos szolgáltatásainak konkrét megvalósítását elemzem. A fejezet célja, hogy betekintést engedjen a portál „belsejébe”, és a felhasznált technológiák alkalmazását bemutassa.

III.1 Konkrét példák

III.1.1 Design

A portál egyetlen eleme, amelyet nem magam készítettem. A design a Free Css Templates oldalról származik, a neve Differential. Az design a Creative Commons Attribution 2.5 licenc szerzői jogi szabályozása alá esik. Ez alá a licenc alá eső dolgok akkor felhasználhatóak, ha visszalinkelünk a készítő oldalára. Ezt megtettem az oldal láblécében.

III.1.2 Rendezvények megjelenítése

Webes alkalmazások készítésnél nehezítheti a dolgunkat, ha több alakban kell megjeleníteni az ugyanazokat az adatokat. A portálon rendezvények háromféleképpen jelennek meg: rövid és hosszú formában, illetve az oldalsávban nevükkel. Ha relációs formában lennének a rendezvény adatok tárolva, három külön programrész lenne felelős a három különböző megjelenítésért. Ezek elkészítése időigényes. Ráadásul ez algoritmikus megoldása lenne az alapvetően deklaratív „Hogyan nézzen ki?” problémának.

A korábbiakban említve volt, hogy a szoftver XML formában tárolja a rendezvények adatait. A megjelenítéshez viszont HTML formába át kell alakítani. Az XML egyik nagy előnye, hogy mára rengeteg hozzá kapcsolódó, szintén XML alapú technológia alakult ki. Ezek közül az egyik első volt az XSLT. Ennek lényege, hogy az XML dokumentumunkat egy szintén XML formátumú, XSLT dokumentum segítségével megformázzuk. Az átalakítást egy XSLT motor végzi, melynek implementációját a Java Standard Edition API rendelkezésünkre bocsátja.

A használt osztályok teljes ismertetését mellőzöm, jelen esetben csak részleteikben érdekesek.

1. `EventXML`: Az adatbázisból kivett XML dokumentumok csomagolóosztálya.
2. `EventRetrierver`: Az adatbázisból eseményeket ezen a session beanen keresztül lehet kivenni.
3. `XMLTransformer`: Az adatbázisból kivett és becsomagolt `EventXML` példányokon tud végrehajtani XSL transzformációt.

A működést a függelék 5. ábráján láthatjuk. Az ábrán az az eset van modellezve, mikor egy rendezvényt hosszú formájában nézünk meg.

1. A szervlet a megjelenítendő rendezvény hosszú alakját kéri az `EventRetriever`-től.
2. Az `EventRetriever` kiveszi az adatbázisból a megfelelő rendezvényt.
3. Ha kértünk formázást, akkor továbblépünk 4.-re. Egyébként továbblépünk a 6.-ra. Csak a

portál belövéséhez volt arra szükség, hogy ki lehessen kapcsolni az XMLTransformert.

4. Az EventRetriever elküldi a rendezvényt a XMLTransformernek .
5. Az XMLTransformer kiveszi az adatbázisból a megfelelő XSL stíluslapot, megformázza a kapott EventXML példányt, és visszaadja azt a hívó EventRetrievernek.
6. Az EventRetriever visszaadja a hívó szervletnek a rendezvényt. A szervlet kiírja azt a kimeneti csatornájára, és visszakerül a vezérlés a hívó JSP oldalra.

III.1.3 Események laponként történő megjelenítése

Az alfejezetben használt EVENT_DATA tábla két oszloppal rendelkezik. Az első neve data és XMLType típusú, a második neve date és timestamp típusú

Az „új események" menüpont alatt az eseményeket lapokra bontva kapjuk meg. Az n. oldalon az ((n – 1) * 5) legutoljára feltöltött rendezvényt megelőzően feltöltött utolsó öt eseményt akarjuk mutatni. Ehhez az kell, hogy a rendezvényeken feltöltési idő szerinti csökkenő sorrendben tudjunk végigmenni. Ez a select kifejezésünkben egy order by rész jelzi. Ezután elkezdhetjük leválogatni a sorokat, és ha az (((n-1)*5)+5). –hez értünk vagy elfogytak a sorok, abbahagyjuk a leválogatást. Ez nagyon rossz teljesítményhez vezethet nagyobb alkalmazásoknál, mivel a lekérdezés végrehajtásához a tábla tartalmát rendezni kell adatbázis oldalon, majd végig kell rajta lépkedni, ami nagyon nagy és felesleges hálózati forgalmat eredményezhet. Egy ilyen lekérdezés a következőképpen nézne ki:

```
SELECT * FROM EVENT_DATA
ORDER BY DATE DESC
```

A hatékonyságot növelheti a rownum nevű pszeudo-oszlop használata. Ez az oszlop nem egy ténylegesen létező oszlop. A lekérdezésben használva az adott rekordról tartalmaz információt. Ez az információ az, hogy a lekérdezési feltételnek megfelelő sorok közül hányadiknak értük el ezt a rekordot. Ha „felső n darab" jellegű lekérdezést akarunk végrehajtani, akkor a rownum<=n jellegű korlátozó feltételt kell használni. Viszont a rownum=n vagy rownum>n jellegű feltételeknek nincs értelmük, mivel rownum csak a where feltételnek megfelelő rekordokhoz rendelődik és csak a hozzárendelődés után nő az értéke, tehát a lekérdezés egyetlen sort sem adna vissza. A rownum értéke az order by utasításrész végrehajtása előtt kerül kiosztásra, így egy sima lekérdezéssel számunkra nem használható. Allekérdezést használva viszont a következőképpen működik:

```
SELECT * FROM
  (SELECT FOO1.*, ROWNUM R FROM
    (SELECT * FROM EVENT_DATA
    ORDER BY DATE DESC) FOO1
   WHERE ROWNUM<:FELSŐ_HATÁR) FOO2
WHERE R>:ALSÓ_HATÁR
```

Így csak a szükséges rekordokat küldjük át a hálózaton.

A lekérdezésünk még mindig rendelkezik hátrányokkal. A legbelső lekérdezés a teljes táblát érinti. Az order by miatt minden rekordot meg kell vizsgálni, de azok legnagyobb része később el lesz dobva. A megoldás egy index használata. Az indexekről a későbbiekben

bővebben írok. Számunkra most a legfontosabb tulajdonságuk, hogy az index tartalmazza a sorok azonosítóját (ROWID), az indexelt oszlop értékeinek növekvő vagy csökkenő sorrendjében. Ezt a tulajdonságot ki tudjuk használni. Ha a date oszlopon csökkenő sorrendű indexet hozunk létre, akkor a legkésőbbi dátum fog a legelső helyen szerepelni, és a legkorábbi a legvégén. Számunkra ez pont a kívánt sorrendet eredményezi, ezért az order by rész használata esetén az SQL motor minden esetben használja az indexet.

A legbelső lekérdezés eddig minden oszlop értékét visszaadta. Ez táblában történő keresést, vagyis felesleges I/O műveleteket eredményez. Gazdaságosabb, ha a legbelső lekérdezésben a ROWID oszlopot kérdezzük le, mert annak értékeit az index is tartalmazza. Mivel nagy mennyiségű szöveges adatot tárolunk a táblában, ezért biztos, hogy az index jóval kisebb struktúra, mint a tábla. A költségesebb oszlopot csak a megfelelő oldalhoz tartozó ROWID-k megkeresése után vesszük ki az adatbázisból. Mivel a rekordok sorrendje megfelelő, már a legbelső szinten alkalmazhatjuk a ROWNUM oszlopra vonatkozó felső határt. Egy ilyen jellegű lekérdezés a következő:

```
SELECT DATA FROM
  (SELECT FOO1.RID FROM
     (SELECT ROWID RID, ROWNUM RN FROM EVENT_DATA
     WHERE ROWNUM<=:FELSŐ_HATÁR
     ORDER BY DATE DESC) FOO1
   WHERE RN>=:ALSÓ_HATÁR) FOO2,
   EVENT_DATA
WHERE FOO2.RID = EVENT_DATA.ROWID
```

III.1.4 Adatok validálása

Az oldalra feltöltendő adatok egy részének bizonyos formai követelményeknek eleget kell tenniük. Ilyen adat a dátum és az e-mail cím. A dátumnak YYYY-MM-DD formátumban kell szerepelnie, amely a „<4 számjegy év>-<2 számjegy hónap>-<2 számjegy nap>” formát jelöli. Az e-mail címet a megszokott, szabályos e-mail formában kell megadni.

Az adatok feltöltésének menete a következő:

1. A felhasználó kitölt egy űrlapot.
2. A felhasználó rákattint a küldésre szolgáló gombra.
3. A böngésző a HTTP szabályainak megfelelő formában elküldi a szervernek az adatokat.
4. A szerver oldal az adatokon validálást végez.
5. Ha a validálás sikeres, feldolgozza az adatokat, és a felhasználó erről értesül.
6. Ha a validálás sikertelen, a felhasználó hibaüzenetet kap.

A portál, az MVC tervezési mintának megfelelően, a küldött adatokat szervletek segítségével dolgozza fel. Legegyszerűbb megvalósítása a feltöltésnek az, ha szervletek végzik a validálást és - EJB komponensek használatával - a feltöltést is. Ez viszont kódismétléshez vezethet: Tekintsük azt a két esetet, amikor felhasználót hozunk létre, és amikor rendezvényt töltünk fel. Mindkét esetben adhatunk meg e-mail címet, mindkét feltöltést végző szervletnek végeznie kell e-mail cím validálást. Az egyik megoldás az lehet, ha a közös funkciókat kiemeljük egy ősosztályba. De mi a helyzet, ha az adatmodellünk változik, és nincs szükségünk továbbra a felhasználók e-mail címeire? Akkor a felhasználókat feltöltő szervlet számára felesleges, valódi működéséhez nem kötődő metódusokkal

rendelkezik. Egy ilyen megoldás kényelmes lehet, de a szoftver architektúrája leromolhat az evolúcióval.

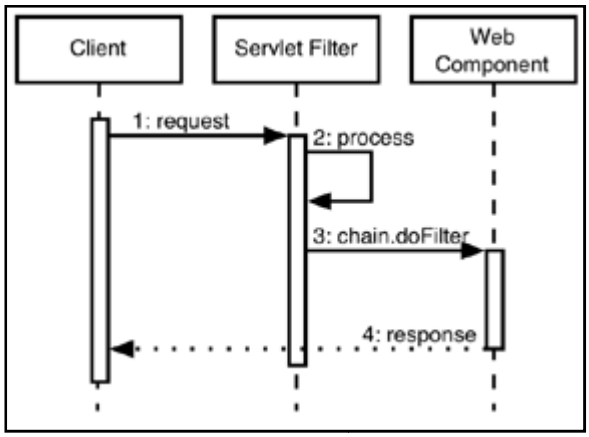
A probléma legjobb megoldása az aspektus-orientált, Interceptor nevű tervezési minta implementálása. Ez a tervezési minta szétválasztja a szoftver modul fő alkalmazáslogikai tevékenységét és az egyéb, adminisztratív illetve technikai műveleteket. A minta működési elve a következő:

1. A szoftver egy elemének (például egy szervletnek) kérést küldünk.
2. Az erre bekonfigurált, az interceptor szerepét betöltő osztály egy példánya a kérést „elkapja”.
3. A kérést az interceptor feldolgozza, majd továbbküldi az interceptorok hívási láncán. Az interceptor lánc utolsó tagja az eredeti címzett.
4. Az interceptor az elkapott kérést más irányba „terelheti”, például egy hibaüzenetet generáló oldalra.
5. Az interceptor lánc utolsó elemének válasza visszafelé végighalad az interceptorok hívási láncán. A választ a legelső interceptor a kérést küldő szoftverelemnek adja tovább.

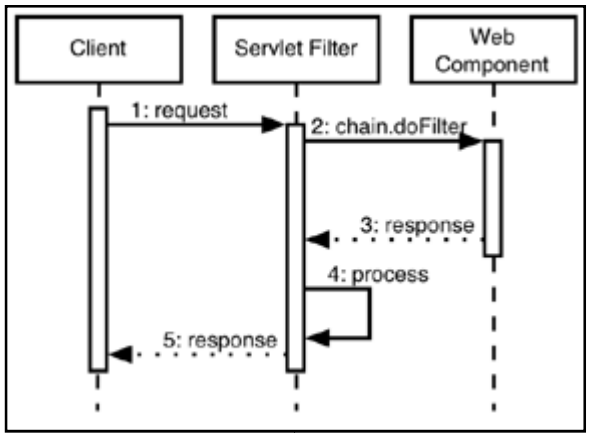
Ennek előnye, hogy az adminisztratív tevékenységek függetlenek a felhasználói követelményeket megvalósító tevékenységtől. Kényelmes a használata, mert egy interceptor kikapcsolható, illetve az interceptor által figyelt elemek listája változtatható.

Az Interceptor tervezési minta a Java EE web rétegében a Filter interfészt implementáló osztályokkal valósítható meg. Egy ilyen osztályhoz rendelhetünk az alkalmazás leíró állományban URL-eket és szervleteket. A hozzárendelt URL-ekre és szervletekre küldött kéréseket „elkapja”. A kéréseket feldolgozó filterek működést a 18. ábra szemlélteti.

A filterek nem csak a kérést, hanem a választ is fel tudják dolgozni. Ez azután történik, hogy a web komponens előállította a választ. Az ilyen filereket leggyakrabban tömörítésre használják, hogy a hálózaton kevesebb legyen az adatforgalom. A válaszokat feldolgozó filterek működését a 19. ábra szemlélteti.



18. ábra Előre feldolgozó filter



19. ábra Utólag feldolgozó filter

A portálon a validáláson kívül filterekkel történik a bejelentkezés és a jogosultságok ellenőrzése is.

IV. Alkalmazott Technológiák

IV.1 Áttekintés

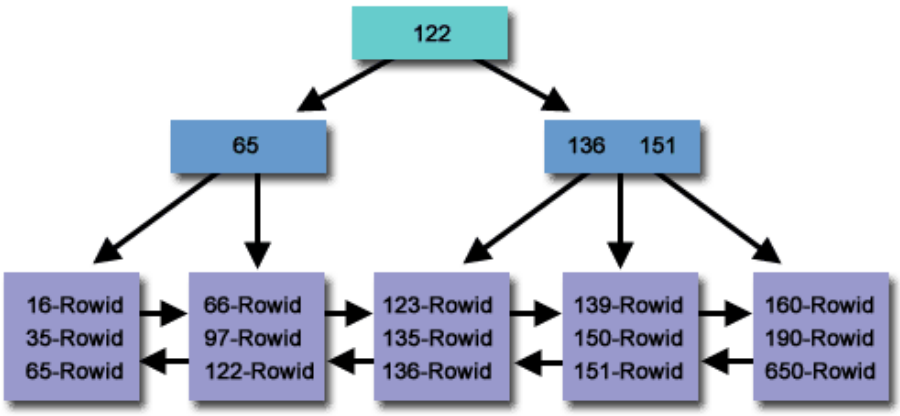
A fejezetben a portál elkészítéséhez alkalmazott legfontosabb technológiai elemeket elemzem. A fejezet célja, hogy bemutassa ezen technológiák használatának egyszerűségét és fontosságát.

IV.2 Oracle technológiák

IV.2.1 Áttekintés

Az Oracle 10g napjaink méltán legismertebb és legelterjedtebb relációs adatbáziskezelő-rendszere, mely minden eszközt biztosít adatbázis alapú szoftverek készítéséhez. Néhány a rendelkezésre álló eszközök közül, melyek igen jelentős részét fölhasználtam az alkalmazás fejlesztésénél:

1. **PL/SQL –** A PL/SQL az Oracle procedurális programozási nyelve, melynek elődjének tekinthető az Ada nyelv. PL/SQL-ben létrehozhatunk tárolt alprogramokat, triggereket, objektumok metódusait, melyek akár a szoftver teljes üzleti logikáját lefedhetik, illetve az SQL nyelven nem vagy csak körülményesen megfogalmazható megszorításokat valósíthatják meg.
2. **Indexek –** Az Oracle a táblákhoz való hozzáférések hatékonyabbá tételére több indextípust is támogat. Az indexek alapvetően a rekordok fizikai címeinek tárolását és azok közötti gyors keresést szolgálják. Az Oracle indextípusai:
    - **B*-fa index –** Mint neve is mutatja, a B*-fa indexek B*-fa adatszerkezetet használnak a rekordok fizikai címének tárolására, ahol a levélelemek nem mutatókat, hanem az adatbázisrekordok fizikai címeit tartalmazzák. Az 20. ábrán egy numerikus adattípusú kulcs alapján felépített B*fa indexet láthatunk.



20. ábra B*-fa index

- **Bitmap index –** Egy bitmátrixot használ, melynek minden sora egy adatbázis

rekordnak, minden oszlop az indexelt oszlopnak illetve oszlophalmaznak a lehetséges értékét szimbolizálja. Ha az indexelt oszlop kevés értéket vesz föl, az index nagyon kis helyet foglal el. A lekérdezéseknél nagyon gyors válaszidőt tesz lehetővé egy bitmap index, mivel a bitmátrix logikai operátorokkal kezelhető. Ennek a technikának a hátránya az, hogy UPDATE és INSERT utasításoknál a tábla nagy részét zárolni kell. Főleg adattárházaknál használják.

- **Fordított kulcsú index –** Hasonló a B*-fa indexekhez, csak megfordítja az indexelt értékeket. Nem tesz lehetővé range scant, így kisebb annak az esélye, hogy I/O versenyhelyzet alakuljon ki.
- **Függvényalapú index –** Egy másik indexelési technikán alapul (pl B*-fa index), nem oszlophalmazt, hanem kifejezést, SQL függvényt, vagy az Oracle valamely natív (pl: PL/SQL, C) nyelvén létrehozott függvényt indexelünk. Ha lekérdezésünkben szerepel az adott kifejezés vagy függvény, akkor az Oracle képes az index alapján megtalálni a rekordot.
- **Partícionált index –** Partícionált táblákon hozhatjuk létre.
- **Szövegindex –** Szöveges adatokat indexelhetünk ezzel a technikával. Segítségével a szövegalapú keresésekhez számos egyéb lehetőség áll rendelkezésre az egyszerű mintaillesztésen kívül, mint például „fuzzy illesztés”, ami a hasonló szavakat tartalmazó rekordokat is visszaadhatja, hasonló hangzású szavak keresése, strukturált szöveges típus (pl: `XMLType`) indexelése és lekérdezése egy XPath-hoz hasonló nyelv segítségével.

3. **Fejlesztő környezetek –** Az Oracle alapvetően két fejlesztő környezetet biztosít a fejlesztéshez: az SQL*Plus-t és annak webes változatát, az iSQL*Plus-t. Ezek segítségével riportokat készíthetünk, illetve scripteket hajthatunk végre. Az SQLDeveloper fejlesztőkörnyezet egy IDE, az SQL*Plus szinte minden szolgáltatását tudja, illetve sok egyébbel bővíti azt ki. Az SQL*Plus-szal ellentétben egy grafikus környezetet biztosít, melyben hatékonyan és gyorsan hozhatunk létre táblákat és egyéb adatszerkezeteket illetve belőhetünk tárolt alprogramokat. Az Oracle a Java alapú alkalmazás-fejlesztésre a JDevelopert biztosítja, mely ígéretes, de egyelőre nem tud a Netbeans és az Eclipse árnyékából kilépni.
4. **Java –** Az Oracle támogatja a JDBC (Java Database Connectivity) protokollt, saját JDBC változata számos egyéb szolgáltatás is nyújt a standard JDBC API-ban definiáltakon felül. A szakdolgozat írásakor az Oracle által támogatott JDBC változat a 3.0-s verzió. Emellett az Oracle támogatja Java nyelven megírt tárolt eljárások létrehozását, Java programok futtatását.
5. **XML –** Az Oracle XML alrendszere támogatja XML adatok kezelését, azokban történő keresést, módosítását, PL/SQL csomagban rendelkezik DOM illetve SAX implementációval.
6. **OO –** Az Oracle 10g egy objektum-relációs (OR) adatbázis, az OO-t minden szinten támogatja. Az Oracle-ben objektumtípusok hozhatóak létre, örölkődés van, bezárás nincs. Az Oracle adatbáziskezelő-rendszer fejlesztése OR alapon történik. Az OO-ra épülve az Oracle támogatja a komponenselvű fejlesztést, továbbfejlesztést. Az Oracle Extensibility Architecture keretrendszerben adatkazettákat hozhatunk létre (Data Cartridge), melyek beépülnek az adatbázisszerverbe. Az adatkazetták tetszőleges funkciót tölthetnek be, például saját indextípust definiálhatunk bennük.

IV.2.2 Az Oracle XML DB

Az Oracle XML DB egy olyan gyűjtőfogalom, amely az **Oracle Database 10g** XML-hez kötődő technológiáit tartalmazza. Az Oracle XML DB segítségével az XML natív módon, SQL be beágyazva kezelhető.
Az XML DB a következő szolgáltatásokat nyújtja:

1. Támogatja a World Wide Web Consortium által létrehozott XML és XML Schema szabványokat.
2. Lehetőséget biztosít XML adatok tárolására, lekérdezésére, frissítésére és transzformálására.
3. SQL rekordhalmazokon XML műveletek végezhetőek.
4. Tárolás-, tartalom- és programozási nyelv-független XML adatkezelés. Ezáltal egyszerű módon elérhetőek az XML adatok.
5. XML specifikus memóriakezelés és optimalizálások.
6. Megbízhatóság, magas fokú rendelkezésre állás, skálázhatóság és biztonság.

Az XML adatok tárolására **`XMLType`** adattípus szolgál. Az **`XMLType`** objektum típus felhasználható SQL-ben mint oszloptípus, PL/SQL-ben mint paraméter, visszatérési érték és változó típusa, valamint Java-ban. Meg kell jegyeznünk azonban, hogy a Java programokban történő alkalmazása platformfüggővé teszi alkalmazásunkat, mert az Oracle specifikus JDBC osztályokat használ. Helyette ajánlott a **JDBC 4.0**–ba bekerült java.sql.SQLXML adattípus, bár ez a dolgozat írásakor az Oracle 10g által még nem támogatott. Az SQL és az XML világ közt átjárást biztosít az Oracle 10g azzal, hogy lekérdezések eredményeit XML formában megkaphatjuk, illetve egyszerű, szöveges adatokból létrehozhatunk XML adatokat.

Az `XMLType` előnyei

1. Az **`XMLType`** segítségével XML műveleteket végezhetünk SQL adatokon, SQL műveleteket végezhetünk XML adatokon, navigációs eszközök állnak rendelkezésünkre, valamint számos XML-en operáló SQL és PL/SQL függvény illetve ezeket csoportosító csomag létezik.
2. **`XMLType`** típusú adatok bárhol megjelenhetnek egy SQL kifejezésben. Például egy **`XMLType`** oszlop lekérdezésének eredményét összekapcsolhatjuk más, relációs oszlopokkal.
3. Az **`XMLType`** csak akkor jelenik meg a memóriában, faformában, ha az szükséges. Ha egy **`XMLType`** oszlopon végrehajtunk egy SQL lekérdezést, a dokumentum szerializált formában van jelen mindaddig, amíg nincs feltétlenül a faszerkezetre szükség. Ilyen eset lehet például, amikor az `extract()` metódust használjuk. Ez a felhasználó számára transzparens, a szerializációt és a deszerializációt az Oracle végzi implicit módon.
4. Az **`XMLType`** oszlopok a gyorsabb keresések érdekében indexelhetőek. Az Oracle Text szövegindexelési technika nyújt erre lehetőséget. Függvényalapú index is létrehozható az `existsNode()` illetve `extract()` metódusok segítségével.

Mikor használjuk az `XMLType`-ot?

1. Ha XML-dokumentumokat egészben kell az adatbázisban tárolni.
2. Ha szükség van a dokumentum egészének vagy részének az SQL-lekérdezhetőségére. Az `existsNode()` és az `extract()` metódusok biztosítanak erre lehetőséget.
3. Ha szigorú típusosságra van szükség az SQL-utasításokban és PL/SQL-programokban. Így az XML adatok meg vannak különböztetve a hagyományos szöveges adatoktól.
4. Ha XPath funkcionalitásra van szükségünk. XPath kifejezések az `existsNode()` és az `extract()` függvények paramétereiként adhatóak meg.
5. Ha dokumentumok XPath-kereséseit kell indexelnünk függvényalapú indexelési technikával.
6. Ha nincs szükség a dokumentum részenként történő frissítésével. Megoldható feladat, például DOM-mal, de a dokumentumrészletek külön történő zárolása még nem megoldott, így komoly performancia-problémák léphetnek fel.

Hogyan használjuk az `XMLType`-ot?

XML-adatok `XMLType` típusú oszlopban történő tárolásakor a következő alapelveket kell figyelembe venni:

1. Az `XMLType` típusú oszlop definiálása. Az oszlopdefinícióhoz bizonyos tárolási jellemzők megadhatóak.
2. Az `XMLType` példány létrehozása. Használhatjuk az `XMLType` konstruktorát, a `SYS_XMLGEN` és `SYS_XMLAGG` függvényeket. Ezen kívül a `JDBC 4.0` verziótól kezdve a `java.sql.SQLXML` példányok megfeleltethetőek `XMLType` példányoknak.
3. `XMLType` példányok lekérdezése, azokból adatok kinyerése. A fentebb említett extract() és existsNode() metódusokkal történik.
4. Oracle Text `CONTEXT` index definiálása, majd lekérdezés a megfelelő operátorok segítségével.

IV.2.3 Az Oracle Text lehetőségei

Az Oracle Text egy kényelmes eszközrendszert biztosít szövegalapú keresések végrehajtásához. A szöveglekérdezések különösen bonyolultabbak lehetnek, ha csak a szokásos relációs műveletek állnak rendelkezésünkre. Az Oracle Text fő alkalmazási területei elsősorban a tartalomszolgáltató webes alkalmazások. Számos keresőmotor illetve tartalomszolgáltató portál használja világszerte.

Az Oracle Text lehetőséget biztosít szöveges keresések és egyszerű relációs műveletek kombinálására egy SQL kifejezésen belül. Segítségével szöveges tartalmak kereshetővé válnak tartalmuk és tulajdonságaik alapján. Az Oracle Text SQL API egyszerű interfészt biztosít ilyen jellegű feladatok megoldására, szövegindexek létrehozására, karbantartására.

Az Oracle Text teljesen integrált az Oracle 10g adatbázissal, ami további hatékonysági garanciát biztosít. A szövegindex az adatbázisban van, a szöveges kereséseket az adatbázis

folyamata futtatja. A lekérdezés-optimalizáló befolyásolhatja a szöveges lekérdezés végrehajtási tervét.

Az Oracle Text szolgáltatásait igénybevevő alkalmazások családját három kategóriába lehet sorolni a szövegindex felhasználási területe szerint:

1. Dokumentumok egy csoportjában történő szöveges keresés
2. Katalógusban történő keresés
3. Dokumentumok osztályozása

1. Az ilyen alkalmazások lehetőséget biztosítanak a felhasználóknak szöveges adatbázisokban történő keresésre. Az alkalmazás alatt lévő adatbázis viszonylag ritkán módosul, az indexet ritkán kell frissíteni. A lekérdezések általában szavakat, vagy kifejezéseket tartalmaznak. Az ilyen jellegű alkalmazásoknál fontos, hogy minél több olyan találatot adjon vissza a kulcsszavak alapján, ami „érdekes" a felhasználónak, és minél kevesebbet, ami nem az. Erre a feladatkörre az Oracle Text a `CONTEXT` indexet biztosítja. Egy ilyen lekérdezésben a `CONTAINS` SQL operátort használjuk. Egy ilyen interakció tipikus forgatókönyve a következő:

- A felhasználó megadja a keresett kifejezést, szavakat.
- Az alkalmazás végrehajt egy `CONTAINS` operátort tartalmazó SQL kifejezést.
- Az alkalmazás a felhasználónak egy találati listát mutat.
- A felhasználó kiválasztja a találatok közül a számára legmegfelelőbbet.
- Az alkalmazás megmutatja a teljes dokumentumot.

2. A katalógusban történő keresés valamilyen termék készlethez kapcsolódik. Mint például egy online kereskedő portál. A keresési feltételek kombináltan vonatkoznak strukturált és nem strukturált adatokra, mint például egy könyv esetében: a keresési feltétel szerint azokra a könyvekre vagyunk kíváncsiak, amelyek címe tartalmazza az „Oracle" és „Java" szavakat, illetve az áruk kevesebb, mint 5000 Ft. Az ilyen alkalmazásoknál elengedhetetlen a gyors válaszidő. Erre a feladatkörre az Oracle Text a CTXCAT indexet biztosítja. Az ilyen lekérdezésekben a CATSEARCH SQL operátort alkalmazzuk. Használati forgatókönyve, az operátorhasználatot leszámítva, megegyezik az 1.-vel.

3. A dokumentumok osztályozását megvalósító alkalmazás transzformációt végez a beérkező adatokon. Előre megadott feltételek alapján vizsgálja a beérkező dokumentumokat, és a feltételek alapján egy osztályozást hajt végre. Ilyen lehet például egy híroldalakat figyelő alkalmazás, amely gazdasággal kapcsolatos hírek érkezésekor az alkalmazást futtató személynek értesítést küld. Erre a feladatkörre az Oracle Text a CTXRULE indexet biztosítja.

Az Oracle Text lehetőséget nyújt strukturált adatok indexelésére is. Az Oracle natív XML típusa, az `XMLType` is ezek közé tartozik. Az ilyen kereséseknél az XML struktúráját fölhasználva leszűkíthetjük, korlátozhatjuk a keresést.

Egy `XMLType` típusú oszlopokon létrehozhatunk `CONTEXT` indexet. Az index létrehozása után az `CONTAINS` operátor kombinálható egyéb, az XML-t támogató operátorokkal. Ezek az `INPATH, HASPATH, WITHIN` operátorok.

Az Oracle Text ezeken túl még számos lehetőséget nyújt, melyeket a dolgozatban nem ismertetek.

IV.3 Java technológiák

IV.3.1 A Java EE áttekintése

A Java EE alkalmazásmodell alapja a Java programozási nyelv és a Java Virtuális Gép. Az évek során bizonyított hordozhatóság, biztonság, valamint a magas fejlesztői produktivitás képezi az alkalmazásmodell alapját. A Java EE olyan alkalmazások támogatására és kialakítására jött létre, amelyek vásárlók, alkalmazottak, beszállítók és partnerek, valamint más, döntéshozó egyéneknek nyújtanak vállalati szolgáltatásokat. Az ilyen alkalmazások jellegüknél fogva komplexek, nagy valószínűséggel több forrásból használnak föl adatokat, és változatos, többféle felhasználói igényeket kell, hogy kielégítsenek.

Ezen alkalmazások a jobb irányíthatóság és karbantarthatóság érdekében az üzleti funkciókat, melyek a különféle felhasználói csoportokat támogatnak, a köztes rétegben, vagy „Middle tier”-ben valósítják meg. A köztes réteg egy olyan környezetet alkot a vállalat szoftverrendszerén belül, mely folyamatos és szoros kontroll alatt áll. Ezt a feladatkört a vállalat IT részlege látja el. A köztes réteg legtöbbször dedikált szervergépen vagy gépeken fut, és a vállalat teljes szoftveres szolgáltatáspalettájához rendelkezik hozzáféréssel.

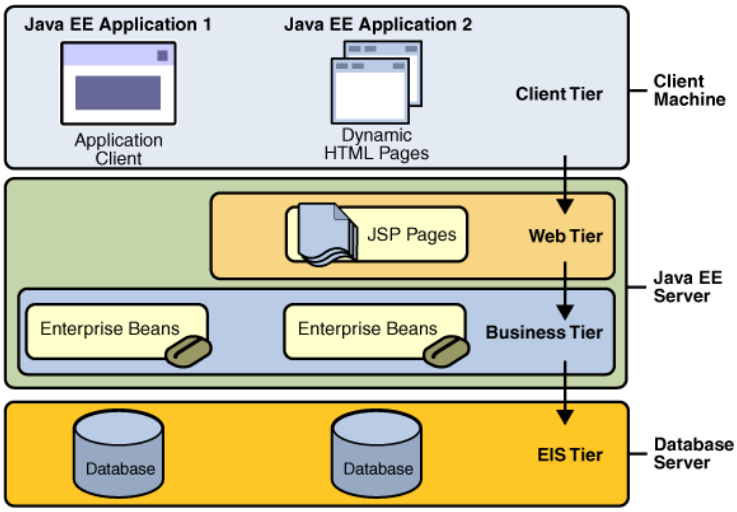
A Java EE alkalmazásmodell egy bizonyos architektúrát szab meg a többrétegű vállalati szintű alkalmazások készítéséhez, mely biztosítja azok skálázhatóságát, rendelkezésre állását, karbantarthatóságát. Ez a modell a többrétegű alkalmazások implementálását két részre választja: az üzleti és a megjelenítési logikát a fejlesztő készíti, az általános rendszer-szolgáltatásokat pedig a Java EE platform biztosítja. A fejlesztő bármely Java EE platformot megvalósító rendszerben számíthat a rendszerszintű problémák egyszerű kezelésére.

Elosztott modell

A Java EE platform elosztott, többrétegű modellt használ a vállalati-szintű alkalmazások fejlesztéséhez. Az alkalmazás-logika több komponensre van osztva a megvalósított funkciók alapján, és ezek a komponensek akár különböző gépekre lehetnek telepítve attól függően, hogy melyik rétegbe tartoznak a többrétegű Java EE környezeten belül. Az 21. ábra mutatja a Java EE rétegeit.

A kliens réteg (client tier) a felhasználó gépén fut. Leggyakrabban webes böngésző, de Java EE támogatja Java alkalmazáskliensek és Appletek használatát is. Feladata lehet input adatok küldése, megjelenítés, esetleg validálás (JavaScript, VBScript).

A köztes réteg két alrétegre osztható. A megjelenítési réteg (web tier, presentation tier) az adatok megjelenítéséért felelős. Feladata a grafikus felület



21. ábra A Java EE alkalmazásmodellje

összeállítása, adatok továbbítása a business tier felé és az onnan kapott adatok megjelenítése. Egy jó architektúrájú Java EE alkalmazásban a megjelenítés réteg semmilyen üzleti funkciót nem valósít meg. A megjelenítési réteg implementálásához leggyakrabban a JSP, JSF, Struts és Servlet technológiákat alkalmazzák.

Az üzleti réteg (business tier) az alkalmazáslogikát implementálja. Feladata, hogy a specifikációban megadott üzleti funkciókat megvalósítsa. Az üzleti réteg kommunikál az adatbázis rétegével.

A köztes réteg két alrétege eloszthető akár több csomópont között is (pl: Apache Tomcat és JBoss kombináció), de ez nem jellemző: a legjelentősebb alkalmazásszerver-fejlesztő cégek termékei mind a két alréteget tartalmazzák.

A Java EE alkalmazásmodell harmadik rétege az adatbázis réteg (database tier, Enterprise Information System tier). Ez a réteg egy adatbáziskezelő-rendszert és egy vagy több adatbázist tartalmaz. Ez a réteg biztosítja a párhuzamos hozzáférést, tranzakciók kezelését és az adatok biztonságát. Az alkalmazás miden perzisztens adatot ebben a rétegben tárol.

Konténerek

A Java EE modellben futási időben a megjelenítési réteg elemeit egy web konténer, az üzleti réteg elemeit egy EJB konténer tartalmazza. A konténerek szerepe egy futtató környezetéhez hasonlít. Nagyon sok adminisztratív feladatot elrejt alkalmazásfejlesztők elől, így kényelmes fejlesztést tesz lehetővé: A konténerek végzik a szervlet és EJB osztályok példányosítását és megszüntetését, az éppen nem használt, de későbbiekben használandó objektumok szerializálását és újbóli memóriába helyezését.

IV.3.2 A Java EE 5 platform technológiái

A Java EE 5 többrétegű, elosztott architektúrán felül számos egyéb szolgáltatást, funkciót specifikál. Ezek közül a legfontosabbak: Servlet 2.5, JavaServer Pages(JSP) 2.1, JavaServer Faces(JSF) 1.2, Enterprise JavaBeans 3.0, Java Naming and Directory Interface, Java Persistence Api. Egy alkalmazásszerver csak akkor felel meg a Java EE specifikációknak, ha ezeket és számos egyéb API-t implementál. A következőkben a legfontosabb, a portál készítésénél is fölhasznált API-król írok.

Servlet

A Servlet (szervlet) API lehetőséget biztosít a fejlesztőknek, hogy dinamikus tartalommal lássanak el weboldalakat. A Servletek Java EE platform megjelenítési rétegének alapjait képezi. Az ehhez a réteghez kötődő API-k nagy része a szervletekre támaszkodik, vagy futási időben arra képződik le.

Egy szervlet egy, a Servlet interfészt implementáló osztály példánya, amely egy kérést - `ServletRequest` példányt – fogad és egy választ - `ServletResponse` példányt - generál. A Java EE szabvány nem szabja meg, hogy a szervlet milyen protokollhoz tartozó kérést kap, de mivel a Java EE platform alkalmazási környezetében a HTTP protokoll a

leginkább használatos, ezért a szabvány definiálja a `HttpServlet` osztályt és a hozzá kötődő összes egyéb, a `Servlet` osztályhoz is kötődő osztályok HTTP specifikus változatát. Például: `HttpServletRequest, HttpServletResponse`.

A szervletek képesek felhasználói munkamenetek nyilvántartására, munkamenethez tartozó objektumok kezelésére. A munkameneteket a `HttpSession` osztály példányai kezelik. A munkamenet nyilvántartása vagy HTTP Cookie-kal vagy az URL-ben történő kódolással történik. Az alkalmazás fejlesztőjének nem kell a nyilvántartás módjával törődnie, azt a web-konténer kezeli.

A szervletek kimeneteként a válaszként kezelt `ServletResponse` példány `getOutputStream()` vagy `getWiter()` által visszaadott kimeneti csatornákat használhatja. Ezekre leggyakrabban HTML szöveget íratunk ki.

JavaServer Pages

A JSP egy olyan, a szervletekre épülő Java technológia, amely lehetőséget teremt a fejlesztőknek HTML-be ágyazott szerveroldali szkriptek készítésére. Egy JSP oldal első futása előtt egy vele ekvivalens szervletre képeződik le, és azt fordítja le a Java fordító.

A JSP-nek két változata létezik. Az egyik a scriptlet. Ez HTML oldalba ágyazott, <% és %> jelek közé írt Java kódrészletet jelent. Ehhez nagyon hasonló a JSP kifejezés szintaxisa, amely <%= nyitó és %> záró jelek közé írt Java kifejezést jelent. A JSP oldal végrehajtása előtt a scriptletek végrehajtódnak illetve a JSP kifejezések kiértékelődnek. A JSP kifejezés értéke a JSP kifejezés helyén jelenik meg a kliens rétegben.

A JSP másik fajtája, amikor XML szintaxist használunk. A scriptletek helyett ilyenkor JSP-specifikus tag-ekkel (ejtsd: „teg”) dolgozunk. Az egyes tag-ek egy XML leíró állományban vannak megfeleltetve bizonyos metódusoknak. Az ilyen kód átláthatóbb, és jobban is illeszkedik a szkriptnyelvekkel szembeni elvárásokhoz.

Az összetartozó tag-eket tag könyvtárakba lehet szervezni, amivel komplett keretrendszerek hozhatóak létre. Mára a Java EE 5 specifikáció az alap JSP tagkönyvtáron túl kiegészült a JSTL tagkönyvtárral és a JavaServer Faces keretrendszerrel.

Enterprise JavaBeans

Egy Enterprise JavaBean egy olyan Java alapú szerveroldali komponens, amely alkalmazáslogikát implementál. Erre épül az Enterprise JavaBeans komponens architektúra. Az Enterprise JavaBean-ek segítségével modulárias felépítésű vállalati alkalmazásokat építhetünk. Az Enterprise JavaBeans ipari szabvány készítőinek legfontosabb szempontjai a következők voltak:

1. Az alkalmazás írójának nem kell olyan általános alkalmazásszerverbeli szolgáltatásokkal foglalkoznia, mint például a tranzakciók kezelése, perzisztencia kezelése, erőforrás-gazdálkodás és gyorsítótár-karbantartás. A fejlesztőnek elég csak az üzleti logikára koncentrálnia a fejlesztés során.
2. Az egyszer megírt EJB komponensek újrafordítás és forráskód-módosítás nélkül telepíthető az EJB-t támogató platformokra.

A többrétegű, elosztott üzleti alkalmazások logikáját megvizsgálva arra juthatunk, hogy habár az üzleti objektumok felépítése és viselkedése szorosan kötődik a megoldandó üzleti problémához, mégis, az üzleti objektumok néhány alapvető szolgáltatása meglehetősen általános jellegű. Ezek a következőek:

1. Perzisztencia kezelés: Egy üzleti objektum rendelkezhet olyan attribútumokkal, melyek állapotát a program két indítása, vagy két felhasználói bejelentkezés között meg kell őrizni. Perzisztens tárként manapság relációs és objektum-relációs adatbázisok vannak használatban.
2. Tranzakció kezelés: Ugyanazon perzisztens adatok egyszerre több felhasználó által történő elérése, adatbázis-problémák kezelése tartozik ezen szolgáltatások közé.
3. Terheléseloszlás kezelése: Nagy mennyiségű kliens egyidejű kiszolgálásakor a kliens számára dedikált üzleti objektumok elérhetőségét biztosítani kell.
4. Transzparens hibatűrés: Ha egy szerver összeomlik, amennyiben mód van rá, egy másik szervernek kell a kliens kiszolgálását átvennie, mindezt a kliens számára észrevétlen módon.
5. Távoli elérés: A kliens a konkrét szerver példányok fizikai elhelyezkedésétől függetlenül tud elérni bizonyos erőforrásokat.
6. Biztonsági követelmények: Az üzleti objektumok által nyújtott szolgáltatásokat védeni kell kliensazonosítási és jogosultságkezelési mechanizmusokkal.
7. Újrafelhasználhatóság: Az üzleti objektum mint komponens integrálható más rendszerekbe. Hogy ezt kényelmesen tehessük, szabványos módon kell fejleszteni, és szabványos környezetben kell futtatni.

Az EJB architektúra definiálja a komponensek és az alkalmazásszerver együttműködéséhez szükséges interfészeket. Ezen interfészek implementálásával tudatjuk az alkalmazásszerverrel, hogy az üzleti objektumunk – a 7. pont kivételével – igényt tart a felsorolt szolgáltatásokra. Ezen szolgáltatások implementációja és működése az EJB fejlesztő elől rejtve van, így csak az üzleti komponensek funkcióinak fejlesztésével kell foglalkoznia. Ezek a komponensek futási időben az EJB konténerben jönnek létre. A felsorolt szolgáltatásokat az EJB konténer nyújtja.

Az Enterprise JavaBeanek több típusba sorolhatóak. Mindegyik típusnak megvan a maga szerepköre.

1. Session bean: Üzleti tevékenységeket, folyamatokat megvalósító komponens. Az alkalmazás szerver készenlétben tar többet is, ha kérés érkezik egy klienstől, használatra odaadja azt. A kliensek az üzleti interfészükön keresztül kommunikálhatnak a session bean-ekkel. Amikor egy kliens meghív egy EJB metódust, azt garantáltan egyedül csak ő hívja meg. Ha nincsenek használatban, a szerver megszüntetheti őket, ha kevés van, újakat hozhat létre belőlük. A session bean-eknek két fajtájuk van:
    - Stateful: Attribútumainak állapotát megőrzi két metódushívás között. A session beanek költségesebb fajtája, mivel az első metódushívás után csak a hívó kliens érheti el a példányt. Ha egy felhasználó meghívja valamely metódusát, onnantól kezdve a session bean példány csak ezzel a felhasználóval kommunikál. Az EJB konténer passziválhatja, azaz lemezre írhatja, ha elérte a maximális bean példány mennyiséget. Ha a felhasználó újra megszólítja, visszahelyezi a memóriába, és egy másik, éppen nem használt példányt passzivál.
    - Stateless: Állapotmentes session bean. Két hívás közt nem őrzi meg attribútumainak állapotát. Létrejötte után több felhasználót is kiszolgálhat, de

egy időpontban csak egyet. Tipikusan egy lépésben elvégezhető feladatokat végeztetnek ilyen session beanekkel, mint például e-mail-küldés.

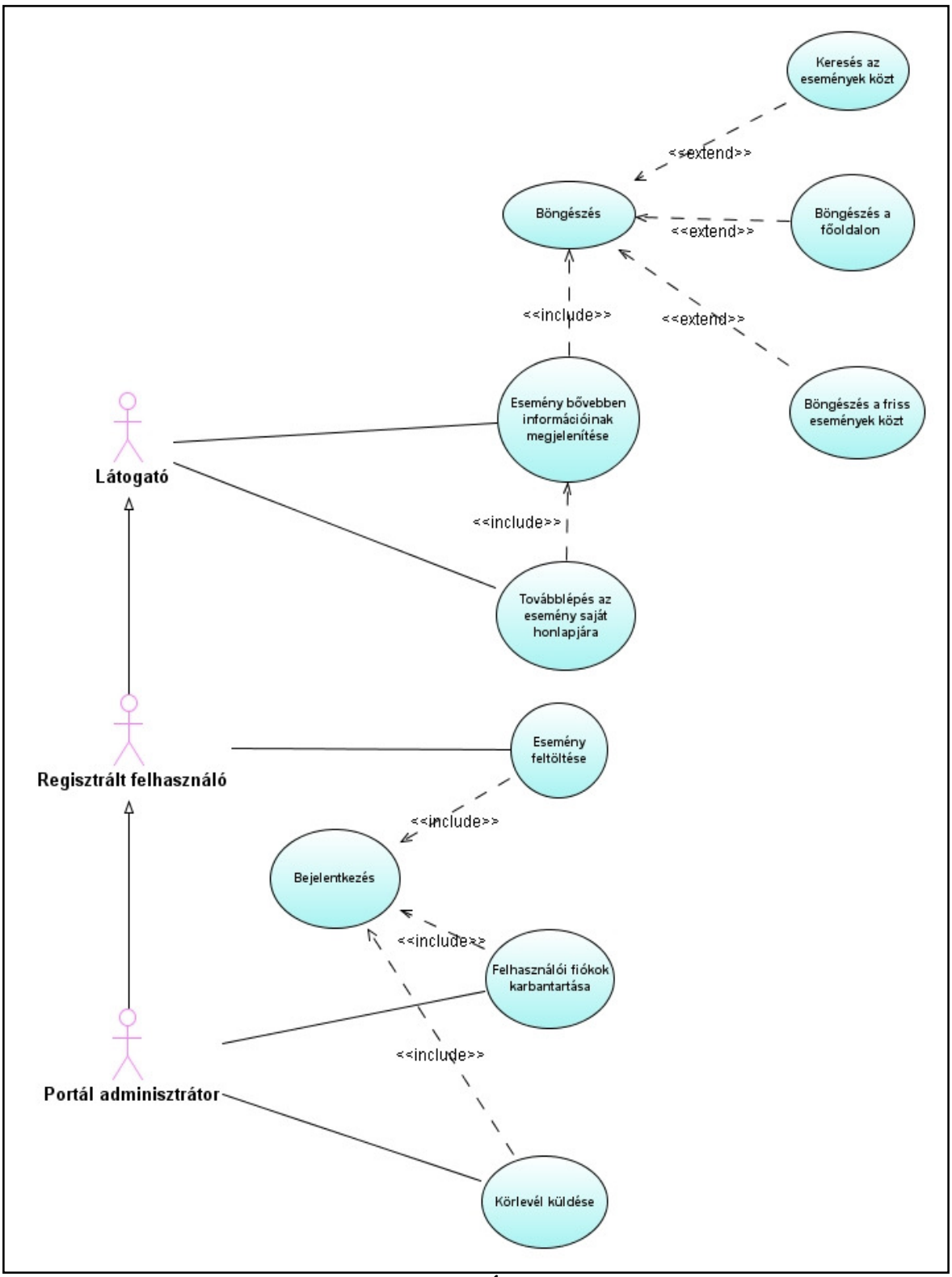
2. Entity bean: Olyan bean, amely perzisztens állapottal rendelkezik. Állapotuk adatbázisban tárolódik. Kezelésük a Java Persistence API-n keresztül történik.
3. Message-driven bean: Olyan bean, amely aszinkron hívások fogadására képes. Nem rendelkezik olyan interfésszel, amelyen keresztül a felhasználó elérhetné. Használata csak JMS API-n keresztül lehetséges. A JMS API üzenetek küldésére szolgál, amelyeket az üzenetsorokra vagy témákra feliratkozott üzenetfogyasztók kapnak meg.

IV.3.3 A Glassfish alkalmazás-szerver

A Glassfish valójában az ingyenes és nyílt forráskódú **Sun Java System Application Server 9.x.** neve. A Glassfish alkalmazásszerver teljes mértékben megfelel a Java EE 5 specifikációnak. Forráskódja a korábbi, zárt forráskódú Sun alkalmazásszerver verziókból, illetve a korábban szintén zárt forráskódú Oracle TopLink-ből származik. Web-konténere az Apache Tomcatból származik, amely a Grizzly nevű Java NIO API implementációval lett kiegészítve.

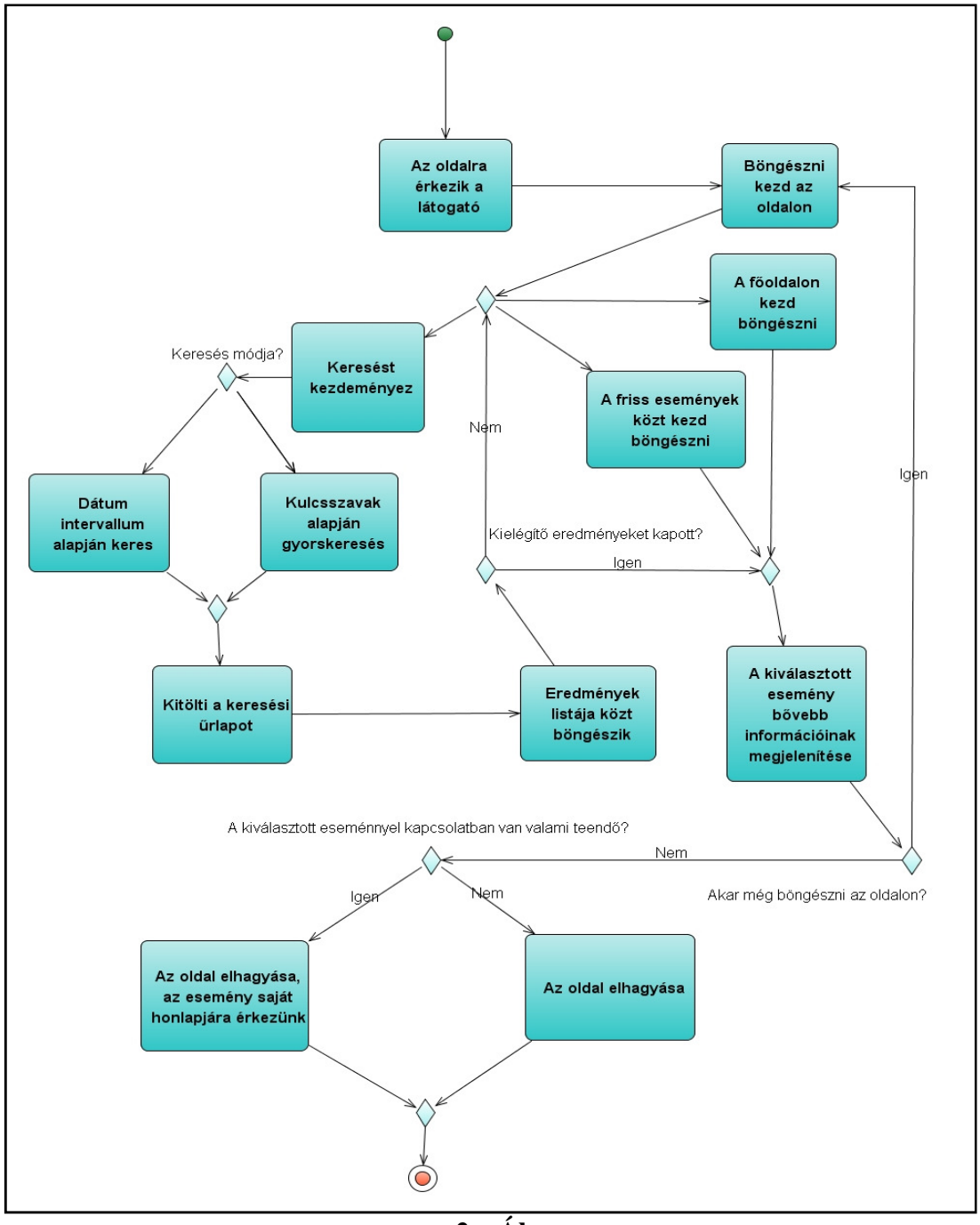
Széles körben alkalmazzák, mivel ingyenes, skálázható, egyszerű az üzemeltetése, legújabb verziója támogatja a gridet és a Java EE szabvány változásait gyorsan követi.

V. Függelék

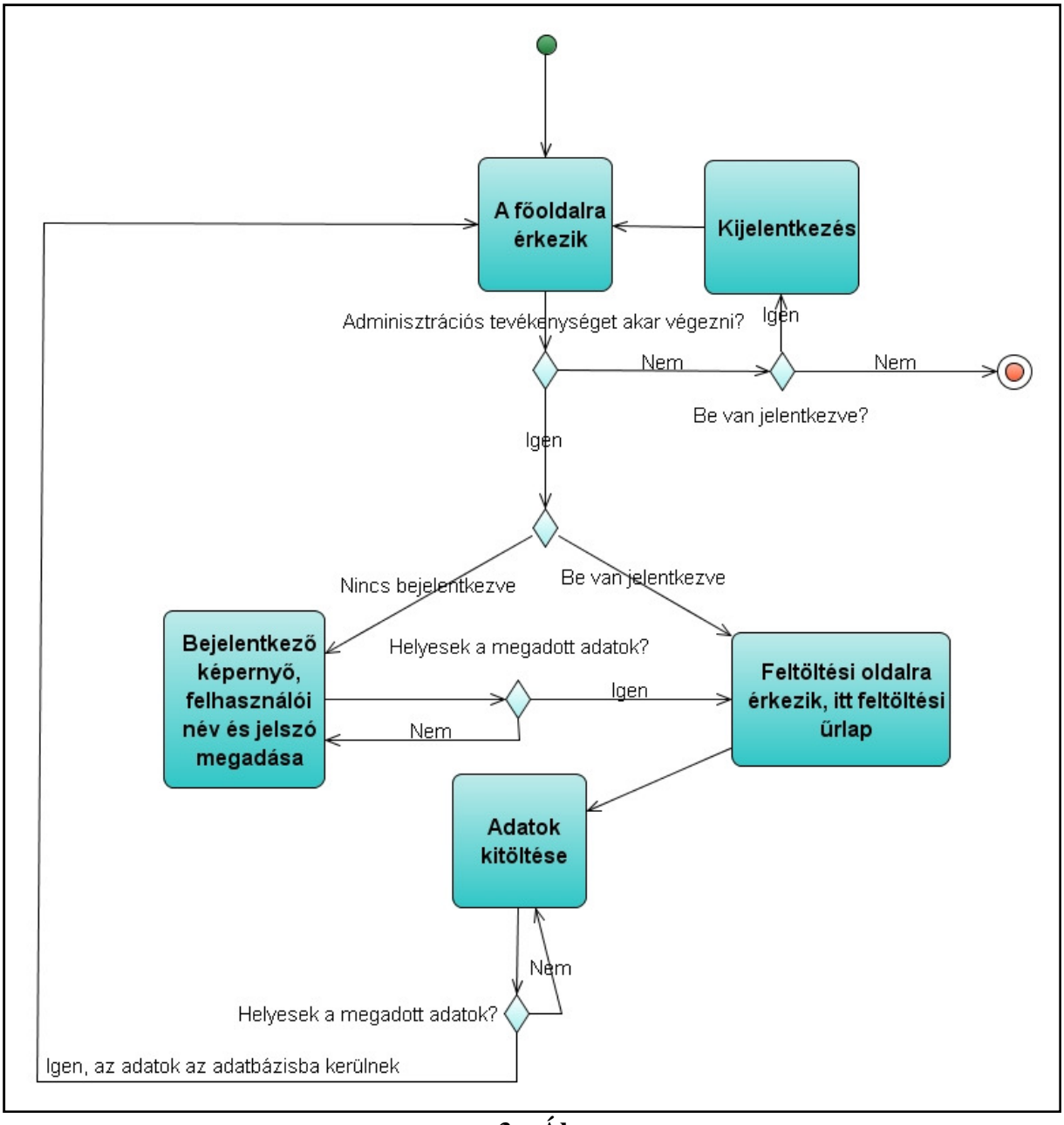


1. Ábra

A felhasználó követelményeket ábrázoló use case diagram

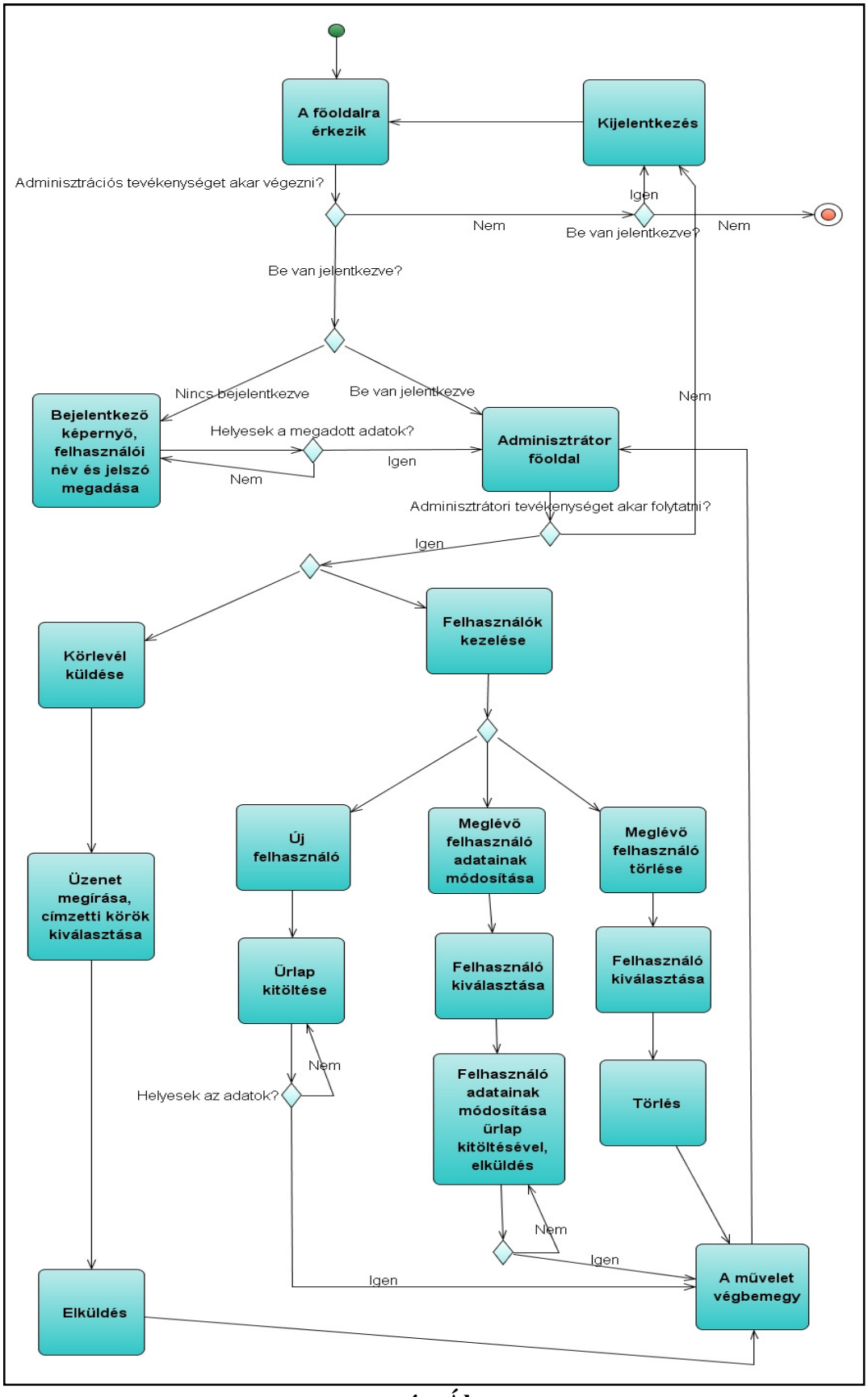


2. Ábra.
A látogató felhasználók tevékenységi sorozatát szemléltető activity diagram



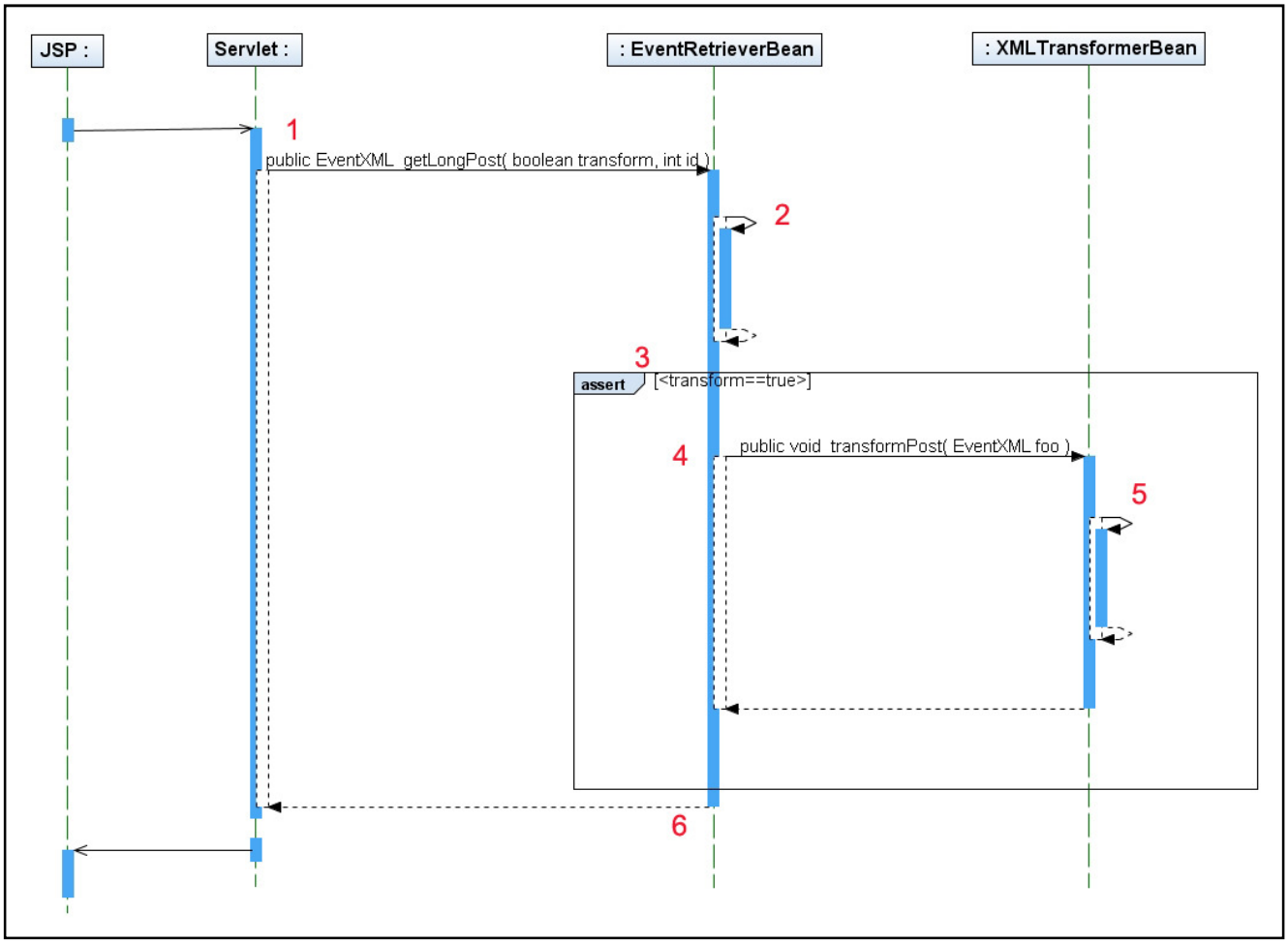
3. Ábra

A regisztrált felhasználók tevékenységi sorozatát szemléltető activity diagram



4. Ábra

A portál adminisztrátorok tevékenységi sorozatát szemléltető activity diagram



5. Ábra

A rendezvények HTML kódjának előállítását bemutató sequence diagram

VI. Irodalomjegyzék

A szakdolgozathoz felhasznált irodalmak listája

1. Kevin Loney: Oracle Database 10g Teljes Referencia
2. Gábor András, Juhász István: PL/SQL programozás, Alkalmazásfejlesztés Oracle 10g-ben
3. Gábor András, Gunda Lénárd, Juhász István, Kollár Lajos, Mohai Gábor, Vágner Anikó: Az ORACLE és a web
4. Nyékiné Gaizler Judit: J2EE útikalauz programozóknak
5. Oracle Text Developer's Guide: http://download-uk.oracle.com/docs/cd/B19306_01/text.102/b14217/toc.htm
6. The Java EE 5 Tutorial: http://java.sun.com/javaee/5/docs/tutorial/doc/index.html

Egyéb források

1. 18. és 19. ábra: www.donews.net/stevenjiang/
2. 20. ábra: http://www.builder.com.cn/2002/0402/46286.shtml
3. 21. ábra: The Java EE 5 Tutorial, http://java.sun.com/javaee/5/docs/tutorial/doc/Overview3.html#wp89164