Tóth Dániel

Önálló labor összefoglaló

A téma: modell alapú kódgenerálás J2EE környezetben. A félévre kitűzött feladat a J2EE környezet összetevőinek megismerése, majd a JBoss.org rendszer telepítése és használatának megismerése; ezután egy konkrét feladat, számomra egy dokumentum kezelő rendszer modellezése, a modellnek a J2EE környezetbe illesztése volt.

A konkrét probléma lényegében egy sokfelhasználós rendszer, melyben az egyes felhasználók rendelkeznek egy-egy saját dokumentumtárral, az egyes dokmentumokhoz hozzáférési jogokat rendelhetnek, így lehetővé téve más felhasználók számára a közös elérést. A feladatban megjelenő entitások: a személyek, a dokumentumok, a jogok, illetve a dokumentumhoz rendelt esemény feljegyzések. A modell osztálydiagramján jól láthatók az entitások közötti relációk. A cél az, hogy kihasználjam a J2EE azon lehetőségét, hogy ne kelljen a perzisztens adattároláshoz relációs adatbázist kézzel felépíteni. Ezt az EJB (Enterprise Java Beans) összetevő biztosítja a Session bean-ek (kiszolgáló business logika) és Entity bean-ek (adatároló egységek) szétválasztásával. A feladatban a kiszolgáló logikához az állapottal bíró (stateful) session bean-t találtam a legalkalmasabbnak, így a szerver követi, hogy a felhasználó éppen mit választott ki és az ahhoz kapcsolódó többi entitást kínálja fel listában. A J2EE környezet összetevői közül a feladat szempontjából még különösen hasznos a JNDI (Java Naming and Directory Service), amely az egyes objektumokhoz való név kötéssel teszi lehetővé a tárolást és visszatöltést; a JAAS (Java Authentication and Authorization Service), ami pedig a felhasználók beléptetését, illetve a jogosultságok kezelését nagyrészt készen tartalmazza. A megjelenítéshez a JSP (Java Servlet Pages) szolgáltat lehetőséget dinamikus HTML generálással. Azt, hogy a szolgáltatást web böngészővel, vagy dedikált kliens alkamazással lehet-e elérni, még nem döntöttem el. A későbbiekben elképzelhető, hogy mindkét változatot megvalósítom. Nem előírt része a J2EE szabványnak, a JBoss által alkalmazott XMBeans (XML ModelMBean) rendszer, mely új szolgáltatás modulok létrehozását teszi lehetővé XML formátumú modell leírás alapján. A dokumentumkezelő rendszer még nem igényli ilyen új szolgáltatások felvételét, ám ezt a lehetőséget bonyolultabb modelleknél hasznosítani lehet.

Jelenleg rendelkezem egy kész, de általános modellel, melyet a megismert lehetőségek alapján alakítok át konkrét, a J2EE környezetbe illeszkedő modellé. Ez jelenti az osztály struktúra módosítását, és az API hívás szekvenviák konkrét behelyettesítését. Az utóbbihoz kell először a feladatnak egy működő implementációját elkészíteni. Ez elvileg még ennek a félévnek lett volna a programja, ám idő hiányában (más tárgyak követelményei miatt) itt hagytam abba. A továbbiakban az eredeti és a konkrétizált modellek, illetve az implemetált kód alapján alapján lehet majd transzformációs szabályokat alkotni, ami végül alkalmas lesz a kitűzött kódgenerálásra. A transzformációk leírásához még meg kell ismerni a Viatra rendszert is.

My assignment is titled model based code generation in J2EE environments. The concrete tasks for this semester were to get familiar with the components of the J2EE system, setting up and learning to use the JBoss.ors application server, designing a model for a concrete application, in my case a document management system and modifying the model for integration into the J2EE framework.

My application is a multi-user system, where each user has an own document pool, and each document has a set off assigned rights, allowing or restrictig access by others. The entities are the personel, the documents, the rights, and some events associated with each document. The class diagrams clearly show the realtions between the entities. My goal is to take advantage of J2EE's facility that removes the need to specify the classical relational database scheme. The EJB component's session beans and entity beans separately contain the server business logic and the data structure respectively. I found the stateful session bean most useful for my application, since it allows easy tracking of the user's actions. For example the user's current object selection determines which realted objects need to be listed. Other features I found useful are the JNDI, which supports the object strorage and retrieval by binding them to names, and the JAAS which contains a fairly complete facility for user management, session login, and access control needed in the application. For the user interface the JSP dynamic html generation can be useful, alhough I have not yet decided, wether go for a web based client access or a dedicated client application. It is possible that both versions will be implemented later. A neat feature of the JBoss environment are the XMBeans service modules, which allow integration of new services into the framework by simple XML descriptions. These are an overkill for the current application, but I guess they are worth exploring for use in more complicated models.

I now have a complete, but general model for the application, which I gradualy modify to get a concrete detailed model integrated into the J2EE environment. This includes modification of the class sructure for the beans model as well as adding the concrete api calls to the sequences. For the later a by hand implementation is necessary. This originally was planned for this semester, but due to lack of time this is where I left it off. The old and new model with the implemented code will allow me to construct transformation rules which in turn will be useable as code generators and this is the goal of this course. For the transformations the Viatra framework will be used and therefore also need to be studied.