University recruitment system in J2EE using open source frameworks. Implementing features, needs and problems.

Andrzej Rostkowski

University of Gdańsk, Poland
Andrzej.Rostkowski@ug.edu.pl

Keywords
J2EE, portal, recruitment, open source, bean shell, Freemarker, Ajax

1. EXECUTIVE SUMMARY

1.1. Backgrounds

University of Gdańsk is one of the biggest high school in Poland. Only last year it recruited over 7000 of candidates on over 100 faculties. In the past every department has committee of its own. One had an excel sheet, other recruited in standard form. Few years ago, IT department decided to support this process. The task wasn't easy. Almost every faculty has its own recruitment rules and laws. In Poland, because of frequent changes in education's law, we have also many types of documents, which allow to take part in recruitment process. Of course almost every type of documents has its own grades scale and that's the reason why the recruitment rules have to be so flexible.

In 2005 first attempt was made. Written in .NET, standalone application was served. But needs of better process control and efficient maintenance created a need for a web application.

Following year the requirements for system were established. First of all, it should give the possibility of defining recruitment rules for all faculties in a simple and readable way. Moreover, this system should have:

- web engine which is fast to develop and refactor,
- simple views for each group of users,
- different panels for each group of users,
- small support and maintenance cost.

Because of the last point, every library and framework used to create the system was open souce.

1.2. Realization

The very beginning was hard. Finding appropriate frameworks was time-consuming. Then going through, sometimes very modest and incomplete, documentation. Unfortunately, no fully suitable web engine was found. So we decided to develop one of our own. It was risky because of short time we had for the system to complete. It of course slowed down the whole process, but at last we found it being a good decision.

1.3. Conclusions

Finally, we managed. Stable, attractive and reliable system was made in about six months using, as was said, only free of charge frameworks and libraries. In the given time the author will show all the met problems and share the best practices in many fields.

2. IRK SYSTEM

IRK system meets all our current needs. It consists of several separable views. Complex security and role system divides supervisor's panel in many smaller modules. For example supervisor's panels provide:

- defining rules,
- changing faculty limits and dates of certain recruitment steps,
- defining own messages that will be send automatically on certain stages,
- possibility of sorting, filtering the ranking lists,
- group operations on candidates,
- exporting data to XLS for same more complex actions,
- complex generator for document: protocols, lists, certificates.

Of course candidates has it's own view. They can register, generate documents, pay tickets. They can

- manage their own money on virtual account,
- receive online messages from department supervisors,
- observe the process of recruitment on many faculty by following the rankings,
- sign up for additional exams.

3. ARCHITECTURE

3.1. WebEngine

Because in 2006, when the decision of developing IRK system was made there weren't any mature web framework which stands for our needs. We wanted Ajax integration, rapid and easy development capabilities. So on we decided to develop our own engine. It's based on java servlets and integrated with Freemarker as a template engine. It also contains simple approach for security issues. In business layer it is similar to JSF. Every page has it's own "code behind page".

Thanks to certain usage of Freemarker as template engine we gained a full separation of business and presence tier.

3.2. Database manager

IRK system has a specific relation with database. It doesn't query db only when it's needed, but caches whole database in RAM.

This approach, while having appropriate server, increases the efficiency of application. Of course it also require much effort in maintenance. In order to keep a synchronization between cached structures and indexes, and the database image write operations are being done immediately.

Database manager operates on objects extending DbItem classes. Thank to this, its methods are entirely independent from specific data structures. DbMgr has a several features:

- it can save or update any instace of DbItem
- reload any object from DB to update corresponding object in cache
- store object instances only in cache if needed
- handles the logging of changes and some security issues.

3.3. Templates

Template is a text file, which has a structure that comply with Freemarker's language specification. It may have any content. In following system templates were always written in XHTML. Only flow of processing the template and its dynamical content were controlled by Freemarker's tags. They differ from the ordinary content with appropriate tags. The following example represents a piece of code responsible for message box:

```
<#if (messageOgolny?exists && messageOgolny != "") || (errorOgolny?exists && errorOgolny !=
"") || (messages?exists && messages.main?exists)>
<div id="messagesBar" style="display: none; position: <#if userAgent?index_of("IE 5") != -1 ||
userAgent?index_of("IE 6") != -1>absolute<#else>fixed</#if>">
  <#if errorOgolny?exists && errorOgolny != "">
          <div id="errorOgolny">
                  <div id="mCloseButton"></div>
                  <span id="errorHead">Błąd:</span>
        <br />${errorOgolny}
          </div>
  </#if>
  <#if messages?exists && messages.main?exists>
          <div id="messageOgolny">
                  <div id="mCloseButton"></div>
                  <#list messages.main as message>
                          <span id="messageHead">Komunikat:</span><br />${message}
                  </#list>
          </div>
  </#if>
  <#if messageOgolny?exists && messageOgolny != "">
          <div id="messageOgolny">
                  <div id="mCloseButton"></div>
                  <span id="messageHead">Komunikat:</span>
        <br />${messageOgolny}
          </div>
  </#if>
          </div>
</#if>
```

In above the one can see characteristic tags:
- <#...> for controlling the flow and running native macros
- ${var} for displaying variable content
- <@...> running predefined custom macros

Syntax of Freemarker language is easy and intuitive. Prepared in such a way, the template is being parsed by the engine after the injection of data source – constructed in code behind page. It's the last step of the process of preparing page after the request from a thin client. The finished XHTML site is the result of this action.

4. USEFUL OPEN SOURCE ADAPTATIONS

Below table describes open source frameworks and how they were used in out system.

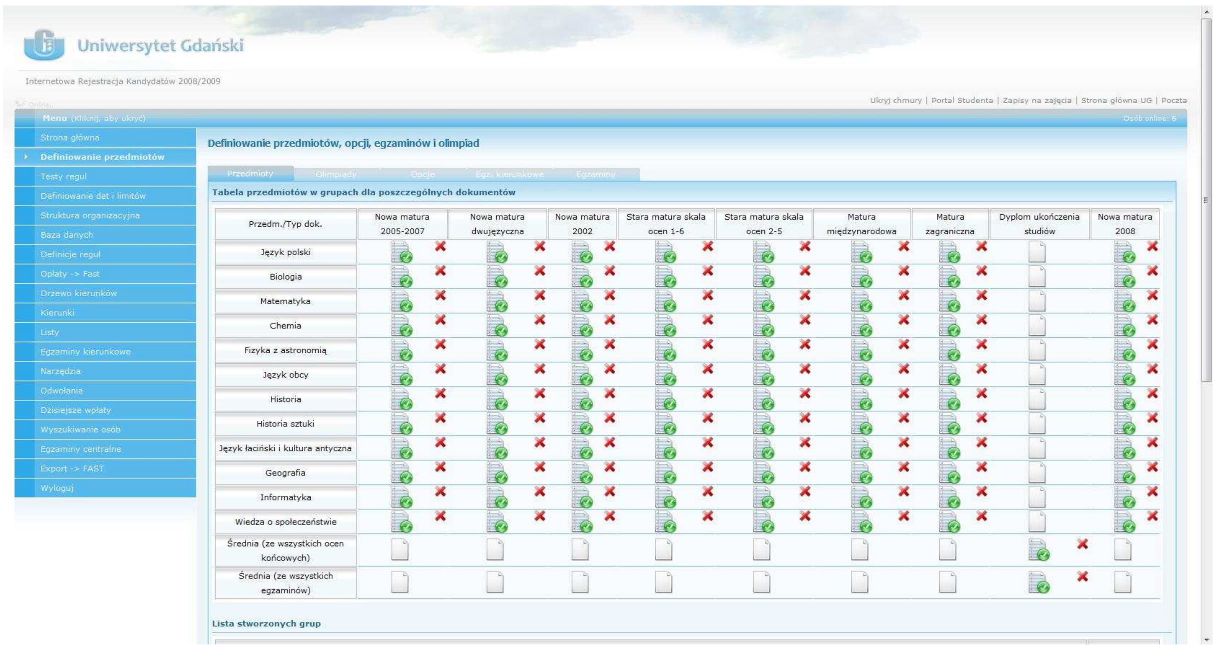
Bean shell	Scripting language based on Java – used for dynamic changes in recruitment rules
iText	Every document generated in IRK is in PDF form
Hibernate	Database's manager lowest layer is based on it
Prototypejs	Javascript framework providing a simple way of DOM manipulation and Ajax requests
Aculoscripts	Javascript framework which simplify the way of creating attractive interface by adding subtly effects
jackcess	Java library for manipulating Access mdb files – import of supervisors structure for every faculty by one click
Jxl	Java library for manipulating XLS files – many various exports
MailApi	Sending email messages to candidates
Freemarker	Java template engine

5. INTERFACE – ERGONOMICS, COMFORT AND ATTRACTIVENESS

To meet nowadays standards and needs we decided to design interface using few open source javascript frameworks.

First and foremost Prototype framework should be mention. We integrated it with WebEngine and implemented many Freemarker macros to make developing in Ajax easier. Thanks to this, IRK system reloads only that part of page, which the action is targeted to.

Success of the layout consists also of Aculoscripts library. It's still developing javascript framework, which includes many useful effects. We managed to create a layout which can interact with user making his work more comfortable. For example any time he want he can hide or show menu gaining the valuable space in more complex views.



Definiowanie przedmiotów, opcji, egzaminów i olimpiad

Tabela przedmiotów w grupach dla poszczególnych dokumentów

Przedm./Typ dok.	Nowa matura 2005-2007	Nowa matura dwujęzyczna	Nowa matura 2002	Stara matura skala ocen 1-6	Stara matura skala ocen 2-5	Matura międzynarodowa	Matura zagraniczna	Dyplom ukończenia studiów	Nowa matura 2008
Język polski	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Biologia	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Matematyka	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Chemia	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Fizyka z astronomią	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Język obcy	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Historia	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Historia sztuki	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Język łaciński i kultura antyczna	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Geografia	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Informatyka	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Wiedza o społeczeństwie	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	✓ ✗	□	✓ ✗
Średnia (ze wszystkich ocen końcowych)	□	□	□	□	□	□	□	✓ ✗	□
Średnia (ze wszystkich egzaminów)	□	□	□	□	□	□	□	✓ ✗	□

Lista stworzonych grup

6. PRODUCTION ENVIRONMENT

Application was deployed on SUN Fire V40 server powered by Solaris os in version 10. The machine has 20 GB of RAM. We used Tomcat in version 6 as a servlet container. Above configuration exceeds the application requirements. Despite loading whole DB image to RAM real used amount didn't overpass 2GB. However sometimes the processors were busy almost in 50%. The main causes of such a state were:

- generating dynamic rankings of candidates
- generating dynamic sorted and filtered lists for supervisor's panel.

The application doesn't keep any copy of lists in cache because of quick changes from many panels and on many conditions.

7. CONCLUSIONS

Developing own web engine turn our to be very comfortable. It is very flexible and adapted to our needs. It makes creating new pages quick and easy, while keeping fixed flow control. It's also easy in refactoring. It doesn't require creating enormous XML files for flow control and it has unique features like:

- getting list of files sended from page's form
- complex structures of session managed pages data

Database manager still need some attention. It could be improved in some advanced caching matters. Now, creating dynamic rankings of candidates even several times a second means sorting collections consisting of over 20k object – often with much complicated criteria. Creating middle tier with some indirect data structures with appropriate refresh engine could improve efficiency and therefore lower server consumption.

Using Freemarker framework as a template engine is very important and efficient feature. Apart from obvious advantage of simple and easy readable syntax it's worth to mention one another – possibility of changing templates also when application is in working state. We often encounter a situation, when a while after redeploying we found out, that presentation layer contains small, but glaring misprint, or even more annoying errors in templates. Shutting down the server couldn't be taken under attention because of to big traffic. Thanks to this approach such incidents could be solved without interruption in service.