

From academic project to production software based on Java web-tier CMS application

J. WOJCIECHOWSKI

COMPUTER CENTER - LODZ UNIVERSITY OF TECHNOLOGY, POLAND

Abstract. Main target of this article is to show what is important in tierd applications from point of view of production ready software on example of custom techniques functional solutions like multi hierarchy, multi-domain operability in app and performance gain practices applied in web-tier CMS application of Lodz University of Technology (TUL) at www.p.lodz.pl working in years 2007-2015 at the beginning as academic project.

Keywords. web system, web-tier application, functional solutions, performance practices, J2EE, Java , Lodz University of Technology (TUL)

Introduction

The Content Management System (CMS) described in this article is a content management system based on Java and HTML technology, and is tailor designed for requirements of Lodz University of Technology (TUL) dealing with heavy load as a production software, content creation, editorial workflow and publishing for TUL. First version of the system was designed in 2005-2006 at doctoral studies and later on the author had to treat it as legacy Struts application and continue improvement to obtain measurable performance effects not using state-of-the-art Java adds on.

Web CMS system consists of frontend and backend. The front is what we see and the backend is hidden within its architecture and logic to obtain functional and performance purposes. This article shows a little behind the curtain of authors original CMS project which emerged within 2005-2013. This CMS is multi-tierd application based on web tier of Java Enterprise Edition (JEE) platform with Model View Controller (MVC) framework, Java Server Pages (JSP), Java Standard Tag Library (JSTL), ExpressionLanguage (EL), Struts 1.2, Object Relational Mapping (ORM) Hibernate and MySql.

As far as the design rationale is concerned author has choosen this solution because in 2005 it was hardly to find open source mature CMS solution which would fulfil requirement of all Steakholdes. Author concentrates on functional solutions of multi hierarchy, multi-domain operability in app, etc. and examples of performance gain practices applied in web-tier CMS application of TUL changing the academic project into production web application.

Some technical solutions cases are shown as examples to explain ways on how the web app was improved from academic project to production software, scaled based on

own experiences on the research [1][2][3][4][5][6][7][8][9][10][11][12][13] and engineering projects, and making at the same time practice, science and algorithms.

This web system was custom designed for demands of administration of Lodz University of Technology and was refactored to service the emerging increasing quantity of incoming http traffic year by year becoming production software. In year 2006 the CMS introduced decentralization of responsibility for the information which was put to the web by administrative departments of Lodz University of Technology. Each administrative department started operating its web content. There were multi hierarchy, multi-domain operability, multi lang versions features of the system implemented in one application context.

In 2010 quantity of visitors increased significantly and improvement of performance was demanded. In on – peak traffic periods the http sessions were from 1k to 10k per day. Rewriting code for decend performance and ability for scalability took consistently till 2013. The refresh of front end was done simultaneously (front end was delivered by other vendor) and in may 2013 the new production and scalability ready web app was deployed at www.p.lodz.pl. The system is ready for operation of 1 to 2k http session at the same time with one server. Nowadays the monthly traffic for p.lodz.pl domain in peak season is 300k http sessions per month what gives circa 1M clics on the web per month.

1. Functional solutions

Functional solutions like multi hierarchy, multi-domain operability in app was implemented because of the requirements of organization. For the first stage the development of CMS project, the aspect of functionality was first priority. Thus the author concentrated on functionality required to operate the administration of Lodz Univeristy of Technology. The so important aspect of performance was an add-on code refactor later on in time.

1.1. To be smarter than infrastructure - multi-domain operability in app

Web-programming model for enterprises called the Java 2 Enterprise Edition (J2EE) extened with architectural framework allowed to build multitierd, here 3-tierd e-business applications.

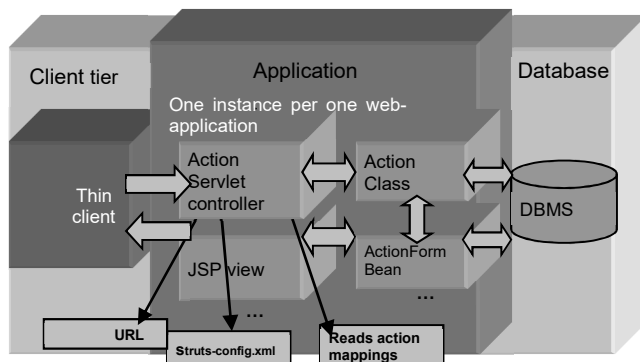


Figure 1. Struts MVC flow of control schema.

Author used one of the MVC frameworks called Struts to operate http request [18].

HTTP requests from thin client are view events, fig. 1, a logic operates the HTTP request and responses through ActionClass and a controller, here ActionServlet directs control to proper views (JSP pages). Detailed analysis of J2EE architecture and code generation from model are described in authors doctoral dissertation [18].

Struts 1.x which is used in mentioned CMS do not support multi-domain operability that might operate many domains in one web context. Usually one instance per one web-application.

Author implemented own logic for multi-domain operability in one web context by adding additional flag “main_context” based on url decomposition in ActionServlet container of Struts framework. The code listed below shows the idea.

```

if(request.getAttribute("main_context")==null ){
    //----- next domain
    String domena=request.getServerName();
    ...
    if( (domena.endsWith("www.studyinlodz.edu.pl") ||
        domena.endsWith("studyinlodz.edu.pl")) ){
        ...//data context
        Menu m=impl.getDefaultLeafForDomainName(domena);
        if(m!=null){
            int numer=m.getId_kat();
            url="/studyinlodz,menu"+numer+"_index.htm";
        }else{
            try{
                request.getRequestDispatcher("/"+"domena_not_operable.htm"
                ).forward(request,response);
            }return;
        }catch(Exception ex){ log.debug(ex.toString());}}
    ...
    try { super.process(request, response);} catch ...
}
}
}

```

Code list 1. Code example of multi-domain operation in one web context block in controller

If server name contains domain checked then the data are being fetched to show in the context of this domain. This might be implemented as Struts extension since most Web-tier application frameworks lack this design pattern.

1.2. Multi hierarchy and multi language support

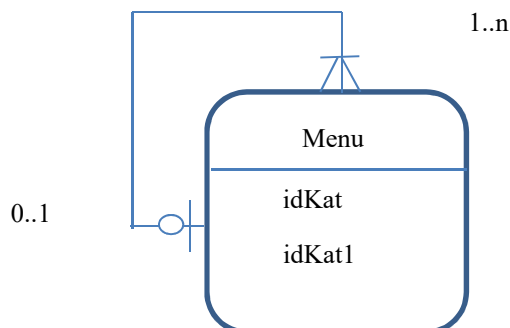


Figure 2. ERD schema of self referencing menu table

This solution is very flexible and usefull for multi hierarchy support for menu items. The data is encoded with UTF-8 standard thus allowing for multilanguage content for all domain context to be presented in one web context. Thus presenting polish, english version, and Chinese, and russian, and Ukrainian.

The relational schema implies risk on how we collect the hierarchical data. One mistake in the algorithm and it may casue the jam problem. Then only helps the memory dump of the thread with “kill -3 javapid”, detailed analysis of the dump, debug and fixation of code.

1.3. Decentralization of operation of CMS by administrative staff of TUL

The Access Control Lists (ACL) for the CMS allowed for decentralization of operation of CMS. Using a combination of ACLs, permissions, and roles, CMS provides methods for setting and restricting the access available to CMS users.

That means that each organizational unit is able to operate its content by themselves. This feature was deployed in 2007 at the Lodz University of Technology. The rest functionality are the following:

- Browser-based interface
- WYSIWYG editing tool
- Role-based workflow
- Permissions model

1.4. Friendly urls

The system has been based on url generation with keywords coming from title of the article put to the CMS by the editor. The url schema was modelled on the basis of the “Google secrets...” [20]. The link structure is plain and wide.

Thanks to the Search Eengine Optimization (SEO) up till now the web page is easily found on top 10 position in Google. For example it keeps top Search Engine Results Page (SERP) position for keyword “Politechniki”. There are circa 30k content urls.

2. Performance solutions

When quantity of visitors of web in 2010 increases the improvement of performance was demanded. Many software designers and developers take the functionality as the most important issue in a product while thinking of performance and scalability as add-on features. Most of them believe that expensive hardware can fix the performance issue. The same was with the authors CMS thus the web application must have evolved from academic project and become production application.

To scale vertically (or scale up) means to add additional CPUs or memory to a single computer [14]. To scale horizontally (or scale out) means to add more nodes to a system, such as adding a new computer to a distributed software application. The author concentrates on architectural approach which touches mostly the aspect of performance and at last the vertical scalability.

2.1. Avoid the database - make cache

In order to improve performance of web-application we have to take into account many aspects of web-application i.e. server side consists of many aspects in the topic of performance. By analyzing the results obtained during this phase it is possible find bottlenecks, memory leaks or performance problems related to database layer.

In this case the re-architecture and re-code the whole solution is demanded but it cost money and time since obtaining performance is a time consuming work and error prone.

The architecture of the system assumed in the academic project in 2006 (static data) that all files are put to the database (because of the ease of db migration) and when http requests comes they are taken again and again out of the database through all 3-tier layers of the web app. This caused big bottleneck when the http traffic increased. The re-architecture and re-coding the whole solution was done in a way that the uploaded files were not only saved to database but also into directory of web server (as static data) as well. This solution concerning static data improved significantly performance.

In the web apps where every request processing action needs much data to process the memory-caching comes into play. The method applied by the author in the CMS is based on caching objects in ApplicationContext and readdressing them to HttpRequest for every request in the session for the presentation layer of guest user-agent - code list 3 illustrates that. Avoiding the database to reduce the database reads is sometimes not possible because of the dynamics of the system and e.g. some-point-critical e.g. financial data but in the author's CMS caching and reloading the cache is appropriate solution for servicing data.

In the CMS the http data is cached where there are more frequent reads operation than updates. The cached object is *PrePrezenter*.

Of course there are algorithms for invalidating and remaking the cache whenever the update from the backend system is done. This solution improved the performance significantly. Caching objects in memory when the system is initialized to avoid creating and fetching from persistent tier too many objects when running improves performance.

```
PrePrezenter pp1=null;
...
pp1=(PrePrezenter)context.getAttribute("preprezenter_spec_pl");
if(pp1!=null&&pp1.getList().size()>0&&breloadnewsstronaglowna==false){
    Helper.setToRequest(request,pp1,"preprezenter_spec");
}else{
    pp1=Helper.getDocumentsMainPage("stronaglownamain",...params );
    Helper.setToRequest(request,pp1,"preprezenter_spec");
context.setAttribute("preprezenter_spec_pl", pp1);
context.removeAttribute("reloadnewsstronaglowna")
/--
}
```

Code list 3. In app cache making schema

2.2. Switch from Hibernate to JDBC on front end requests

At the beginning of working on CMS the front end as well as back office were designed with a Hibernate - an object-relational mapping framework for

the Java language. This framework mapped from Java classes to database tables (and from Java data types to SQL data types) excellent but there was a little performance overhead. Author decided to rewrite the code for front-end http actions to jdbc instead of hibernate leaving the previous framework for a backend.

JDBC (Java DataBase Connectivity) access a database in much quicker time. Of course there is source code overhead instead when writing JDBC logics.

2.2.1. Not leak resources

Close any jdbc instances that weren't explicitly closed during normal code path, not 'leaking' resources. The code listing 4 shows the details of explicit releasing resurces in whatever path of execution of the code.

```

public List getRodzajeMenu(String lang) throws DAOSystemException {
    Connection c = null;
    Statement stmt=null;
    ResultSet rs = null,rs2 = null;
    List ret = new ArrayList();
    javax.sql.DataSource ds;
    String f="select identyfikator_menu from dmenu m where
    m.id_jezyka='"+lang+"'
    group by m.identyfikator_menu order by m.kolejnosc";
    try {
        ds = DBAFactory.getDs();
        c = ds.getConnection();
        stmt = c.createStatement();
        rs=stmt.executeQuery(f);
        while(rs.next()){
            ret.add(rs.getString("identyfikator_menu"));
        }
        if(rs!=null)rs.close();
        if(rs2!=null)rs2.close();
        stmt.close();
        c.close();
    } catch (SQLException se) {
        log.error("Error List "+se.toString());
        throw new DAOSystemException("SQLException: " + se.getMessage());
    }finally {
        if (stmt != null) { try {
            stmt.close();
        } catch (SQLException sqllex) {}
            stmt = null;}
        if (c != null) {
            try {
                c.close();
            } catch (SQLException sqllex) {}
            c = null;
        }
    }
    return ret;
}

```

Code list 4. JDBC closing connection

2.3. Coordination beetwen threads – “synchronized” keyword

“The primary tool for managing coordination between threads in Java programs is the synchronized keyword. The synchronized keyword will force the scheduler to serialize operations on the synchronized block. If many threads compete for the contended

synchronizations, and only one thread is executing a synchronized block, then any other threads waiting to enter that block are stalled. If no other threads are available for execution, then processors may sit idle. In such situations, more CPUs can help little on performance. The JVM has to maintain a queue of threads waiting for that block (and this queue must be synchronized across processors), which means more time spent in the JVM or OS code and less time spent in your program code” [14][15]. To avoid the hot lock problem, the author made synchronized blocks as short as possible – code listing 5 - moving the thread safe code outside of the synchronized block.

```
package config;
...
public class SessionCounter implements HttpSessionListener {
...
    if(se.getSession().isNew()==true){
        synchronized(this){
            activeSessions++;
        }
    }
...
}
```

Code list 5. Synchronized block

Paying attention to lock granularity is recommended. When we put the "synchronized" keyword on a method, we are locking on "this" object implicitly making lesser granularity. The entire object is locked when calling its method thus we decrease performance and ability to scale. The same is the lock on static methods which means lock on all instances of this class [15]. Programmer may choose the attitude from vast choice of wait-free methods like compare and swap CAS or from java.util.concurrent.atomic package.

2.4. Non-Blocking IO in Tomcat 6

Upgrade of server Tomcat 5 to Tomcat 6 which has embraced non-blocking IO was key factor of better performance of the web application of Lodz University of Technology.

In non-blocking IO, a working thread will not binding to a dedicated request [15]. If one request is blocking due to any reasons, this thread will reuse by other requests, In such way, Glassfish can handle thousands of concurrent users by only tens of working threads.

2.5. Adding more Memory to the Server

Memory is an important resource for your applications. Enough memory is critical to performance especially for database systems. More memory means larger shared memory space and larger data buffers, to enable applications read more data from the memory instead of disks.

“Too little memory will cause garbage collection to happened too frequently. Enough memory will keep the JVM processing your business logic most of time, instead of collecting garbage. Java garbage collection relieves programmers from the burden of freeing allocated memory, in doing so making programmers more productive. The disadvantage of a garbage-collected heap is that it will halt almost all working threads when garbage is collecting. In addition, programmers in a garbage-collected environment have less control over the scheduling of CPU time devoted to freeing

objects that are no longer needed. If one adds Java applications are NOT scalable by given too much memory. In most cases, 3GB memory assigned to Java heap (through "-Xmx" option) is enough. " Cited [14].

This scenario gives the conclusion that Java applications must be well prepared for the scalability, from the system design phase to the implementation phase of the products' life cycle. The scalability is really based on ones programmer vision.

Conclusion

In a Web environment concurrent use is measured as simply the number of users making requests at the same time. When the application has decent response time then this aspect is called good performance. Performance refers to the capability of a system to provide a certain response time. It is also software quality metric.

It became crucial for the author of CMS when number of visitors of web page of Lodz University of Technology increased in 2010.

As we see in this paper the system become production application from academic, focusing in the later stage on the performance increasing technique rather then functional. The author realizes that the systems are NOT scalable out-of-the-box and in nearly all cases this is architectural problems.

The system reached decent response time ~1s for 0-2000 http request at the same time. The statistics shows nearly 300.000 http sessions per month in a peak period.

Author suggest premature optimization should be done with performance optimization during designing and implantation phase.

Lifecycle APM (Lifecycle Performance Management) and Continuous Performance Management [19], suggest to get all information to know about the scalability and performance characteristics of your application any time. This serves as a basis for deciding when and where to optimize.

"Concluding we can say that if we want our systems to be scalable we have to take this into consideration right from the beginning of development and also monitor throuhout the lifecycle. If we have to ensure it, we have to monitor it. This means that performance management must then treated equally relevant than the management of functional requirements". [16]

References

- [1] Wojciechowski J., Napieralski A., „Zastosowanie Platformy J2EE w Projekcie Serwisu Internetowego DWZ P.L.” XI Konferencja „, Sieci i Systemy Informatyczne, Łódź, październik 2003, pp. 131-135, ISBN 83-88742-91-4
- [2] [Wojciechowski J., Napieralski A. „System wspomagający wykładowcę i studenta przez WWW ” MIKROELEKTRONIKA I INFORMATYKA, maj 2004, KTMiI P.Ł. pp. 235-238, ISBN 83-919289-5-0](#)
- [3] [Wojciechowski J., Sakowicz B., Dura K., Napieralski A.,” MVC model struts framework and file upload issues in web applications based on J2EE platform” TCSET’2004, 24-28 Feb. 2004, Lviv, Ukraine, pp., 342-345 , ISBN 966-553-380-0](#)
- [4] SZYMAŃSKI G., Wojciechowski J.A., CIOTA Z. “Design of Web-Based Tutor-Supporting System on The Basis of JAVA Platform” 11th International Conference MIXDES 2004, Szczecin , Poland 24-26 June, pp. 607-610, ISBN 83-919289-7-7
- [5] [Wojciechowski J.A., OWCZAREK M., Napieralski A. “Java Web Services Application in University Web System” 11th International Conference MIXDES 2004, Szczecin , Poland 24-26 June, pp. 611-614, ISBN 83-919289-7-7](#)

- [6] [Wojciechowski J., Kozłowski M., Napieralski A. "Security Aspects of Web Applications Implemented within J2EE Platform" 11th International Conference MIXDES 2004, Szczecin, Poland 24-26 June, pp. 619-622, ISBN 83-919289-7-7](#)
- [7] [Wojciechowski J., Napieralski A., „System jednolitej autoryzacji w środowisku heterogenicznym opartym o www z zastosowaniem architektury klucza publicznego PKI oraz bazy danych LDAP” International Workshop for Candidates for a Doctor’s Degree, 16-19 October, Wisła, pp. 461-464, ISBN 83-915991-8-3](#)
- [8] [Wojciechowski J., Murlewski J., Napieralski A.: POZYCJONOWANIE STRON INTERNETOWYCH W SERWISACH WYSZUKIWAWCZYCH NA PRZYKŁADZIE GOOGLE, KMiTI Mikorzyn 23-25.09.2005, Mikroelektronika i Informatyka, Prace Naukowe, Łódź 2005, str. 89-94, ISBN 83-922632-0-0](#)
- [9] [Wojciechowski J., Murlewski J., Sakowicz B., Napieralski A., "Object-relational mapping application in web-based tutor-supporting system", CADSM, Lviv-Polyana, Ukraine, Feb. 23-26, 2005, pp. 307-310, ISBN 966-553-431-9](#)
- [10] [Owczarek D., Wojciechowski J., Murlewski J., Sakowicz B., Napieralski A.: „Electronic Document Management System”, 13th International Conference Mixed Design of Integrated Circuits and Systems MIXDES 2006, 22-24 czerwca 2006, Gdynia, wyd. KMiTI, str. 791-792, ISBN 83-922632-9-1, str. 808, A4](#)
- [11] [Wojciechowski „New methodology in designing reactive systems with formal methods based on authorization for hierarchical, component based system with time dependencies”, International PhD Workshop for Candidates for a Doctor's Degree OWD 2006, 21-24 X 2006, Wisła, Polska](#)
- [12] [Wojciechowski J., „Mapping of Petri net formal model of concurrent system to class model with aspect of polymorphism in object oriented paradigm”, Zeszyty Naukowe Katedry Mikroelektroniki i Techniki Informatycznych : Mikroelektronika i Informatyka, zeszyt nr 7, Łódź 2007, ISBN 83-922632-5-1, ss.167-170](#)
- [13] [Zięba B., Wojciechowski J., Jabłoński G., Zabierowski W., Napieralski A., :Web-Based Distributed Physic-Based Simulation System of Semiconductor Diode Structure” 10th International Conference Mixed Design of Integrated Circuits and Systems MIXDES 2003, 26-28 June 2003, Łódź, Poland, pp. 690-693, ISBN 83-7283-095-9](#)
- [14] [Scaling Your Java EE Applications, By Wang Yu, 01 Jul 2008, TheServerSide.com <http://www.theserverside.com/news/1363681/Scaling-Your-Java-EE-Applications>](#)
- [15] [The Top 10 Ways to Botch Enterprise Java Application Scalability and Reliability, Cameron Purdy on Jul 23, 2008 <http://www.infoq.com/presentations/10-ways-botch-scalability-reliability>](#)
- [16] [Performance vs. Scalability September 11, 2008, Alois Reitbauer <http://apmblog.dynatrace.com/2008/09/11/performance-vs-scalability/>](#)
- [17] [Wojciechowski J., “From formal methods to implementation based on Petri Nets model of concurrent systems”, Pomiary, Automatyka, Kontrola, Vol. 53, No. 5/2007, Maj 2007, pp.132-134, ISSN 0032-4140](#)
- [18] [Wojciechowski J. ”Translation method of Coloured Petri Nets models towards Java Web application schema based on multi-tier distributed authorization system” Praca doktorska, 2009, Biblioteka Politechniki Łódzkiej](#)
- [19] [Compuware APM application lifecycle performance management \[http://www.compuware.com/en_us/application-performance-management/products/lifecycle-performance-management.html\]\(http://www.compuware.com/en_us/application-performance-management/products/lifecycle-performance-management.html\)](#)
- [20] [“Google secrets How to get a top 10 Ranking on the most important search engine in the world” Blue Moose Webworks inc. 2003 ISBN 0-9728588-0-6](#)