

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Prof. Dr. M.Akif Eyler. Throughout my graduate and post-graduate life and my thesis, he provided me with valuable guidance, morale and help. Without his directing and patience, I cannot imagine completing my thesis.

I would like to thank Mehmet İsmail Yeyinmen for his generosity in providing me a nice and convenient place to work.

I would also like to thank Tayfun İşbilen and Özgür Oktay for their theoretical help and support in learning new technologies.

Finally, I would like to thank my wife Fatma Ergin for her patience and moral support throughout my studies and my family for providing me the encouragement and motivation.

October 2005

Lemi Orhan Ergin

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	i
TABLE OF CONTENTS.....	ii
ÖZET.....	v
ABSTRACT	vi
LIST OF SYMBOLS / ABBREVIATIONS.....	vii
LIST OF FIGURES	viii
LIST OF TABLES	xi
PART I INTRODUCTION	1
PART II SEPARATION OF CONTENT FROM LAYOUT	4
II.1. Layout – Content Correlation.....	4
II.2. Web Page Layout Technologies.....	6
II.2.1. First Generation: JSP Pages With Embedded HTML Layout Tags .	7
II.2.2. Second Generation: JSP Pages With Includes	7
II.2.3. Third Generation: JSP Templates	8
II.2.4. Fourth Generation: Configurable Page Templates	10
II.2.5. Web Page Layout Frameworks	11
II.2.5.1. The Tiles Document Assembly Framework.....	12

II.2.5.2. SiteMesh Web Page Layout and Decoration Framework	12
II.3. Content Fetching Mechanism	13
PART III eXaMpLe PROJECT: THE CONTENT MANAGEMENT	
ENGINE	15
III.1. Key Motivations and the Aim of Develop	16
III.2. Development in Enterprise	16
III.3. History of Development	17
III.3.1. Development Branch for Pre-Releases	17
III.3.2. Development Branch for Releases	18
III.4. Birds-Eye-View to the Whole System	18
PART IV JConfiX: THE CONFIGURATION MANAGEMENT	
FRAMEWORK	20
IV.1. Key Purposes and Objectives	20
IV.2. Managing the Configuration Files	20
IV.3. The Structure of JConfiX	21
PART V JEWEL SYSTEM: WEB LAYOUT MANAGEMENT	
FRAMEWORK	24
V.1. Key Purposes and Objectives	25
V.2. Multi-Layered Presentation Layer	25
V.3. Terminology	26
V.4. Integrating a Web Page	26
V.5. Role Based Customization Mechanism	28
V.6. Navigation Management Module	29
V.7. Configuring Layouts	31
V.8. Embedded Cache Management	32
V.9. Managers: Business Layer of JEWEL	33
V.10. Workflow of JEWEL	34
V.10.1. Generating a Layout from an Ordinary Http Request	35
V.10.2. Generating a Layout from an Http Request After Form Submission	
.....	37
PART VI QUARRY: MANAGER FOR DYNAMIC AND MULTI-	
LINGUAL CONTENT	39
VI.1. Key Purposes and Objectives	39
VI.2. JClassifier: Category Management Engine	40
VI.2.1. Architectural Overview	40

VI.2.2. Configuring JClassifier	41
VI.3. The Structure of JClassifier and QUARRY in MVC Architecture...	43
PART VII CONCLUSIONS AND FUTURE WORK	46
REFERENCES	48
APPENDIX A A SAMPLE WEB APPLICATION	50
APPENDIX B PSEUDO CODED WORKFLOW	56
B.1. Coming Parameters.....	56
B.2. Rights Checking.....	56
B.3. Defining the View	58
APPENDIX C TECHNOLOGIES / PLATFORMS USED	61
C.1. Application Framework.....	61
C.2. Ant	62
C.3. Castor	62
C.4. Display Tag Library.....	62
C.5. J2EE - Java 2 Platform, Enterprise Edition	62
C.6. JDOM	63
C.7. JSP - Java Server Pages.....	63
C.8. JSTL	63
C.9. Spring Framework: Layered Java/J2EE Application Framework.....	64
C.10. XML – Extensible Mark-up Language	64
APPENDIX D FILE HIERARCHY IN JEWEL SYSTEM.....	65
D.1. Files for Rights Management	67
D.2. Files for Layout Management	67
D.3. Files for Layout Template Management.....	67
D.4. Files for Right-Layout Interaction:	68
RESUME.....	69

ÖZET

Web sitesi yada portal geliştirilirken değişik amaçlar için hazırlanmış birçok yazılım geliştirme çatısı kullanılıyor. Bunlardan birkaçına örnek olarak arabirim yönetim çatısı ve uygulama çatıları verilebilir.

Bu tezde, özellikle yoğun içerik içeren web portallarını geliştirmek için tasarlanmış birçok çatı geliştirilmiştir. Bu çatılar, içerik yönetim motoru adı altında gruplandırılabilirler. eXaMpLe Projesi, bir arabirim yönetim çatısı, bir içerik yönetim çatısı, bir sınıflandırma çatısı ve bir yapılandırma yönetim çatısı içerir. Bu çatılar, web geliştirme alanına birkaç yenilik getiriyor. Örnek vermek gerekirse, JEWEL Sistem, arabirim yönetim çatısı, 11 adet XML tabanlı yapılandırma dosyası kullanıyor. Böylelikle, web sitesi içi yönlendirme yönetimi, kullanıcı yönetimi, rol tabanlı güvenlik yönetimi ve dinamik arabirim yönetimi özelliklerini de destekleniyor. Başka bir örnek olarak, içerik yönetim çatısı olan Quarry, her tür bilgiyi sınıflandırarak geliştiriciye düzenlenmiş ve yönetilebilir bir ortam sunuyor. Bu araçlar sadece birbirleriyle uyum içinde çalışabilmelerinin yanısıra herbiri ayrı ayrı başka projelerde de kullanılabilirler. eXaMpLe Projesi tamamıyla Java'da ve açık kaynak teknolojileri kullanılarak, yazılım geliştirme sürecinin her evresinde standartlar ve önerilen teknikler takip edilerek geliştirilmiştir.

Ekim 2005

Lemi Orhan Ergin

ABSTRACT

Many frameworks are being used for different purposes for developing web applications, such as web sites and portals. Some are responsible for layout management and some are used as an application server. However, there exists a lack of frameworks for developing content intensive web sites.

In this thesis, a set of frameworks for developing the backbone of web portals, especially for content intensive web applications is introduced. Totally, these tools can be grouped as content management engine. eXaMpLe Project contains a web page layout framework, a content management framework, a category management framework and a configuration management framework. These frameworks offer some original enhancements to web development area. For instance, JEWEL System which is the layout management framework of eXaMpLe Project, uses 11 XML files in order to support navigation management, user management, role-based security and dynamic layout management. Also Quarry, which is the content management framework, categorizes all kind of data and let it be accessed as an organized form of data. Not only these frameworks can work together seamlessly, but also each one is designed to work as a component. All of them have completely been written in Java and by open-source technologies and by following the standards and best practices of web application development.

October 2005

Lemi Orhan Ergin

LIST OF SYMBOLS / ABBREVIATIONS

HTML : Hypertext Mark-up Language

XML : Extensible Mark-up Language

JSP : Java Server Pages

LIST OF FIGURES

Figure II.1. Presentation simply consists of layout and content	5
Figure II.2. A web page having many different logical parts	5
Figure II.3. A JSP page with HTML layout tags embedded in it	7
Figure II.4. A JSP page having include directives with hardcoded references to pages.....	8
Figure II.5. A more generic template page getting parameters from display page.....	9
Figure II.6. Sample XML portion storing references the pages as parameters	10
Figure II.7. Workflow diagram in fourth layout generation technology	11
Figure II.8. Different content fetching mechanisms in several architectural layers .	13
Figure III.1. Developing a single web page by using JEWEL System in a development team	17
Figure III.2. eXaMpLe Project and its sub-projects in one picture	19
Figure IV.1. Sample workflow of configuration value generation process	21
Figure IV.2. The main and the only configuration file of JConfIX	22
Figure IV.3. Workflow of a any project built on JConfIX.....	23
Figure V.1. Presentation layer spitted into two sub-layers	25
Figure V.2. Automaton-like flow of web page integration.....	26
Figure V.3. The whole picture of web page integration with filtering processes	27
Figure V.4. Sample action and the responses of JEWEL System.....	28

Figure V.5. Configuration files for rights management process	29
Figure V.6. Configuration file for navigation management.....	30
Figure V.7. Parameter flow between managers in JEWEL System.....	31
Figure V.8. The precedence of developing configuration files.....	32
Figure V.9. Caching mechanism of JEWEL System	33
Figure V.10. Interactions between managers in JEWEL System	34
Figure V.11. Handling a simple http request	36
Figure V.12. Handling an http request after a form submission	38
Figure VI.1. Build structure of QUARRY and JClassifier	41
Figure VI.2. A configuration file of JClassifier storing item types	42
Figure VI.3. A configuration file of JClassifier storing category types	42
Figure VI.4. A configuration file of JClassifier storing the matchings between items and categories	43
Figure VI.5. The Structure of JClassifier and QUARRY in MVC Architecture...	44
Figure A-6. Homepage of the reference web application	52
Figure A-7. User management console of the reference web application	53
Figure A-8. Content category console of the reference web application	54
Figure A-9. Content update page of the reference web application	55
Figure B-1. Http Request variables coming to JEWEL System	56
Figure B-2. Determining the view of an action.....	56
Figure B-3. Defining an action.....	57
Figure B-4. Determining the default layout for a role.....	57
Figure B-5. Defining a role	57
Figure B-6. Permission Denied action and its corresponding view	58
Figure B-7. Guest (Abbr: g) is determined to access and use login action	58

Figure B-8. Checking if a role has right to execute an action.....	58
Figure B-9. View definition configuration.....	59
Figure B-10. Integrating the pagelets defined in a view	59
Figure B-11. Processing of pagelets and templates.....	60
Figure D.1. Configuration files of JEWEL System	65
Figure D.2. Mapping files of the configuration files of JEWEL System.....	66
Figure D.3. Templates in JEWEL System	66
Figure D.4. Properties files of JEWEL System.....	66

LIST OF TABLES

Table A-1. Environmental information about the reference application.....	51
Table A-2. Source code measurements as lines of codes.....	51

PART I

INTRODUCTION

Internet is expanding in time and the number of portals, which are simply major starting sites for users, is more than we can imagine. According to the research of International Telecommunication Union [ITU], 580.78 million of people were online worldwide and keeping on clicking as of May 2002. There exist millions of web portals containing billions of pages being clicked all around the world. Google has already indexed more than 8 billion web pages [GOG] from approximately 40.3 million active domains [WHOS]. As seen in the whole picture, the number of web sites and pages are increasing in each day.

Hundreds of tools are used to generate web pages. But developing a web portal with various facilities is today's great challenge since a portal should introduce not only a set of web pages, but also a number of services to the users.

In this thesis, we focused on possible solutions to the basic challenges of software developers while developing content intensive web applications, namely web portals. Content management systems [BRA00] are out of scope of this thesis since it is not a complete system. Instead, this thesis is about a set of frameworks that helps to construct content intensive web applications. It is possible that these frameworks could be used to create a content management system.

This thesis report is mainly build on the answers to the questions stated as follows: What is the challenge that the developers are facing during the development process? And how can the frameworks that you wrote solve these challenges?

An important issue software developers are experiencing during the regular development process is the improper separation of stages in an enterprise web application. That is the situation when the layers of an enterprise web application

cannot be developed individually one by one. That causes to work on more complex and uncomfortable development environment.

In order to dispose this problem, 4 generic frameworks were developed for different purposes. A configuration management framework was developed in order to provide a generic API for managing and governing configuration values in the system. A generic category management framework was developed for providing an API for managing any organizable and categorizable data. Our content management framework built on the category management framework in order to provide an organized, categorized and manageable content data. And finally, a web page layout framework was developed to govern the presentation layer in all stages and propose a new, multi-layered approach to the presentation data. All these frameworks can be used as a modular component and be integrated to other software projects individually.

All of them have completely been written in Java and by open-source technologies. Tens of web technologies were used during development, deployment and testing stages of this thesis. In this thesis, the development stages of the frameworks will not be discussed. Instead, the architectural and structural features of the frameworks are discussed. The proposed system has release 7 major releases and 3 of these have tested on real production environments. The last version of the system has already used on developing a commercial web portal and this portal has been working well for more than one year.

In Part II, the correlation between the concepts content and layout is discussed in details. In this part, you will find background information and literature review also. In Part III, eXaMpLe Project is introduced. Since this project has 4 different sub-projects, Part III will give you the information about the whole project. History of development and used technologies and platforms are specified. In Part IV, we will introduce JConfIX – Configuration Management Framework with all aspects. In Part V, our web page layout framework, JEWEL System, is introduced. Its structural design and its way of working are discussed in details. Our content management framework, QUARRY, and its backbone the category management framework, JCategorizer are introduced in Part VI. Tens of figures are put to the thesis to support the concepts introduced. In Part VII, you will find the conclusion discussion.

In Appendix A, you will see a concrete example in all perspectives. User management module and category management module are also illustrated by

screenshots. Pseudo coded workflow of JEWEL System is introduced in Appendix B. And finally, we added the technologies and platforms used in our development with brief information for each to Appendix C.

PART II

SEPARATION OF CONTENT FROM LAYOUT

One of the important stages of the web application development is the layout management process. Since web interfaces are the showcases of web applications, these have to be treated carefully and intently for supporting web design standards. Fortunately, layout management process concerns only a part of the whole project.

The other stage is the content management process. Since content is the feed of a web page, it has to be treated in all layers by all development stages. A developer spends most of his time for dealing with the content management issue, either by coding or designing.

II.1. Layout – Content Correlation

There are many questions need to be answered in order to clarify the stages of web application development. These are stated as follows: What is layout and what is content? Or in other words, what is the relation between layout and content?

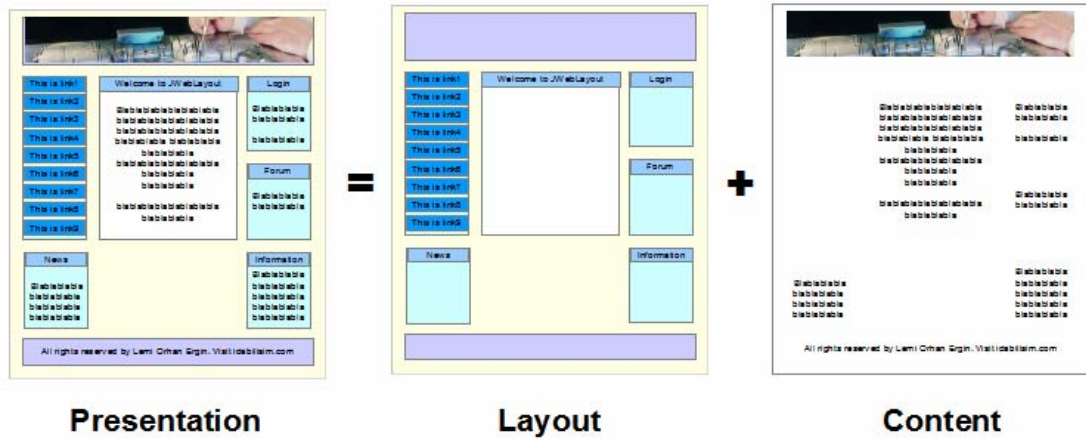


Figure II.1. Presentation simply consists of layout and content

Content of a web page is any information we want to express to the visitors. The ingredients can be files in any format, like texts, pictures, videos. Layout of a web page determines how a page is visually structured. The layout deals with consistent look and feel for all pages. Layout is the container for content. The same content can be displayed to the visitors by different layouts in different web pages. Also most of the time, single web application shares the same layout for many different contents.



Figure II.2. A web page having many different logical parts

Figure II.2 shows the home page of the official web site of Marmara University Faculty of Engineering. Here the page has a header and a footer, two menus, and several different body contents like login, search and banner parts. The layout defines where the header goes and where the menu goes and where the body content is displayed and so on.

II.2. Web Page Layout Technologies

In this subsection, we will discuss the presentation or display needs of typical web applications.

Web applications need to present common look and feel for all pages it displays. By “common look and feel”, we say a common layout structure with common header, footer, menu's, forms, dynamic and static content. Although we can encounter some unusual and extreme page layouts in Internet, we can say that there exists a common look and feel in general.

Also, these display pages need to be easily maintained. If the layouts have less hard-coded material, it will be easier to maintain the common look and feel. For example, it would be highly desirable to go to single place to change the common look and feel (instead of changing each and every page of a Web application). The popularity of today's famous content management systems are coming from providing a single place for managing the layouts.

And last, the layout and contents need to be separated. This means that layout and contents should be able to be changed without affecting the other.

At this point, the evolution of page layout technology can be given. This information is based on Sang Shin's categorization [SHIN]. Layout technology can be categorized them into 4 generations as follows.

II.2.1. First Generation: JSP Pages with Embedded HTML Layout Tags



Figure II.3. A JSP page with HTML layout tags embedded in it

As in Figure II.3, the first generation technology is plain vanilla JSP page with HTML layout tags embedded in it. So here there is really no separation between the layout and contents.

II.2.2. Second Generation: JSP Pages With Includes

In the 2nd generation technology, the JSP pages use JSP include directives or include actions. Here there is some separation between layout and contents. And JSP pages that contain contents are reusable in several display pages. However, the layout page is not reusable because every page requires different layout pages.

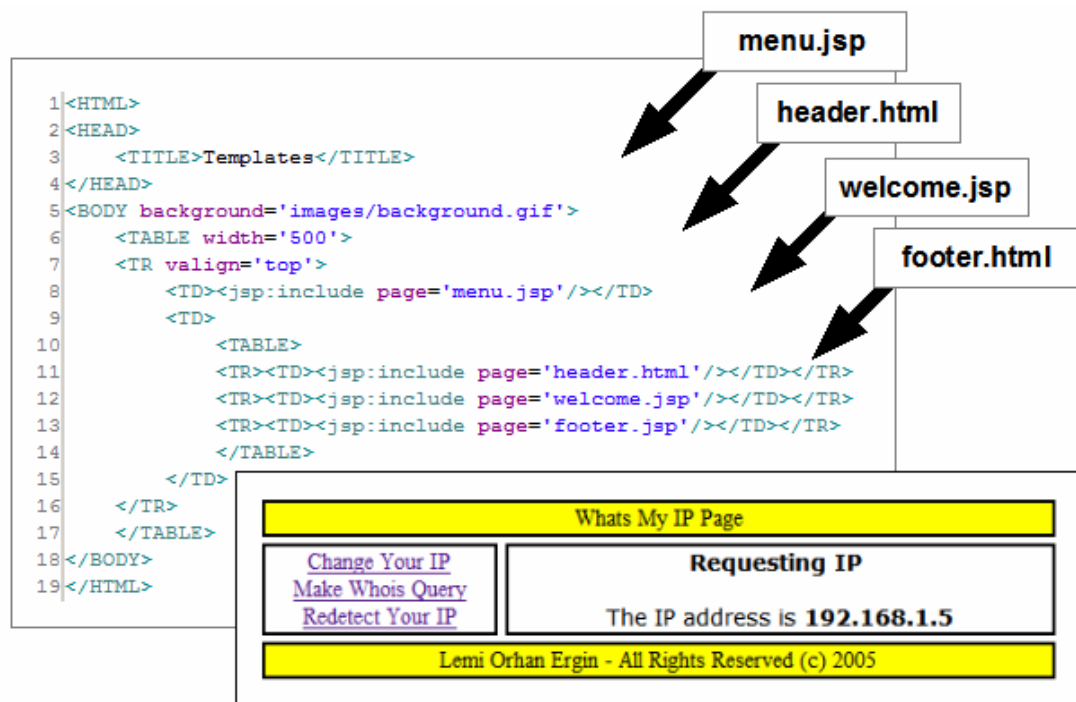


Figure II.4. A JSP page having *include* directives with hard coded references to pages

In Figure II.4, the content is included with `<jsp:include>` directives, which allow content to vary without modifying display page itself; however, because the layout is hard-coded, layout changes require modification to the display page.

The higher the number of pages needs a change in their layout, the harder the modification can take place. In other words, if layouts scheme needs to be changed, every display page has to be changed, which is an inconvenience.

II.2.3. Third Generation: JSP Templates

In the 3rd generation, the concept of JSP template was introduced. Here there is relatively clean separation between the layout and contents. And both JSP pages and layout pages can be reusable.

A template is a JSP page that uses JSP template custom tag library to describe the layout of a page without specifying contents. So there is a separation of contents from the layout. And the content is inserted into the template page during runtime.

Because a common template is shared by many display pages, template provides a single place to go and change when layout change is required. Also, the template provides a consistent look and feel without having to hard-code it in every display page.

Display Page

```
1<jsp:forward page="template.jsp" >
2  <jsp:param name="title" value="Web Templates" />
3  <jsp:param name="header" value="header.html" />
4  <jsp:param name="menu" value="menu.jsp" />
5  <jsp:param name="mainContent" value="welcome.jsp" />
6  <jsp:param name="footer" value="footer.html" />
7</jsp:forward>
```

Template Page

```
1<HTML>
2<HEAD>
3  <TITLE><%=title%></TITLE>
4</HEAD>
5<BODY>
6  <TABLE width='500'>
7    <TR valign='top'>
8      <TD><jsp:include page='<%=menu%>' /></TD>
9      <TD>
10         <TABLE>
11           <TR><TD><jsp:include page='<%=header%>' /></TD></TR>
12           <TR><TD><jsp:include page='<%=mainContent%>' /></TD></TR>
13           <TR><TD><jsp:include page='<%=footer%>' /></TD></TR>
14         </TABLE>
15      </TD>
16    </TR>
17  </TABLE>
18</BODY>
19</HTML>
```

Page Names

menu.jsp

header.html

welcome.jsp

footer.html

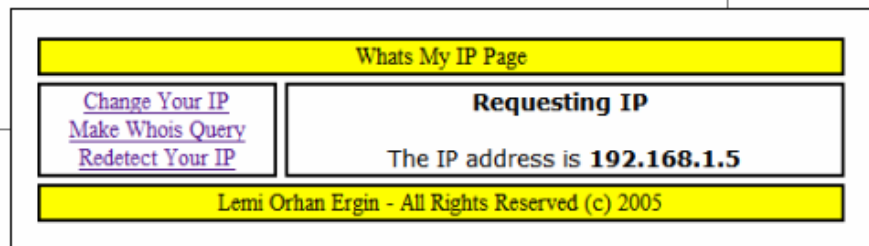


Figure II.5. A more generic template page getting parameters from display page

In Figure II.5, there is a generic template called as template.jsp which provides a layout in which contents can be inserted during runtime. Here it means that the display page is using template.jsp as a template and then contents, header.html, menu.jsp, footer.html etc. are then inserted into the template during runtime.

Preparing templates needs to define where the contents should be inserted. If the layout of a page needs to be changed, then developers should modify the hard coded page names from templates. Developers should be very careful when configuring templates, since hard coded material will most probably consume too much time for any kind of modification in the application.

II.2.4. Fourth Generation: Configurable Page Templates

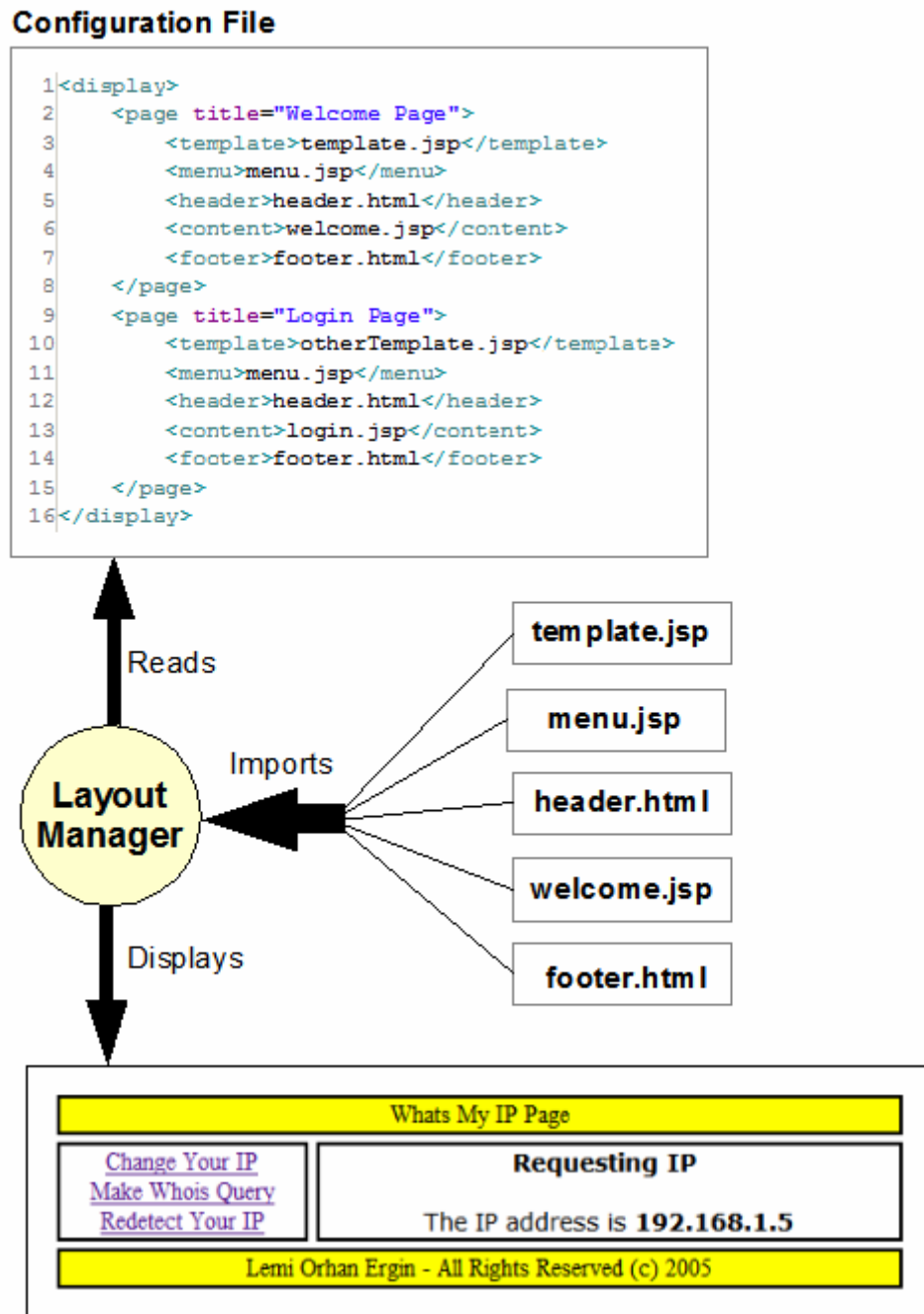


Figure II.6. Sample XML portion storing references the pages as parameters

In the fourth generation, the concept of JEWEL System is introduced (Figure II.6). As we will see later on, JEWEL System is basically a superset of template technology and in fact it extends the concept of JSP template with “parameterized components” or “pagelets”. It uses enhanced version of the web page generation workflow as in Part V.10, Workflow of JEWEL.

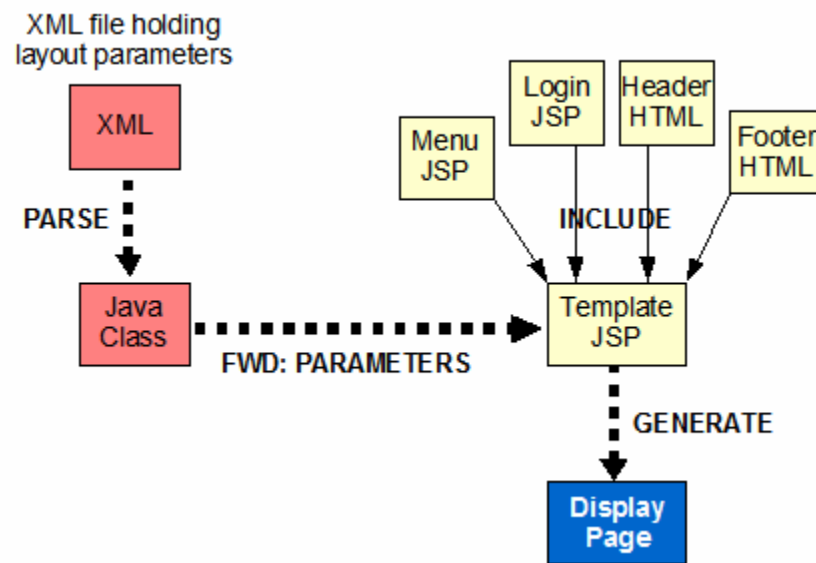


Figure II.7. Workflow diagram in fourth layout generation technology

Although the sample XML in Figure II.6 is an XML portion that holds parameters of a page layout, it is not related with JEWEL System. However, by the help of the graphic above, it describes how a page is generated by fourth generation web page layout frameworks.

In Figure II.7, a servlet (in 2-layered web architecture) or a business class (in three layered web architecture) fetches the XML file holding the parameters. Then it forwards the parameters by response to the template file. Template file includes all the content files and generates a single page having both layout and content together.

II.2.5. Web Page Layout Frameworks

Open-source Java provides two powerful frameworks that help you manage the user interface complexity: Tiles and Sitemesh. Tiles composes web pages from discrete regions of your user interface known as tiles [CAV04]. A tile contains a JSP page for layout and one or more JSP pages for content. Sitemesh decorates web pages with decorators that can be associated with URL patterns [SMH]. Once you set up your decorators, you can decorate pages that match a decorator's URL pattern.

II.2.5.1. The Tiles Document Assembly Framework

Tiles project was born as a sub-project of Struts (i.e. an MVC framework) [SIG05], but it has evolved to become a standalone, Struts independent web page layout framework. Tiles is introduced in official Tiles Guide [TILE] as below:

“Tiles builds on the "include" feature provided by the Java Server Pages specification to provide a full-featured, robust framework for assembling presentation pages from component parts. Each part ("Tile") can be reused as often as needed throughout your application. This reduces the amount of markup that needs to be maintained and makes it easier to change the look and feel of a website.”

All display page definitions are collected in tiles-defs.xml. Including the content pages and generating a display page processes are the same as Figure II.7.

II.2.5.2. SiteMesh Web Page Layout and Decoration Framework

SiteMesh project is developed for the same purpose: Managing web page layout as Tiles and others. The developers are introducing SiteMesh in their official web site [SMH] as follows:

“SiteMesh is a web-page layout and decoration framework and web application integration framework to aid in creating large sites consisting of many pages for which a consistent look/feel, navigation and layout scheme is required. SiteMesh intercepts requests to any static or dynamically generated HTML page requested through the web-server, parses the page, obtains properties and data from the content and generates an appropriate final page with modifications to the original. This is based upon the well-known GangOfFour Decorator design pattern.”

Although Tiles, SiteMesh and other web page layout frameworks are aimed to generate maintainable and configurable display pages, the ways of doing things are very different. B. Siggelkow describes it best [SIG05] as follows:

“When you use Tiles, you adopt the Tiles-way of doing things by constructing pages using Tiles custom JSP tags, creating definitions in a Tiles configuration file, and linking actions to Tiles using the TilesRequestProcessor. Tiles becomes a pervasive part of your web application.

SiteMesh takes a different approach. It uses a servlet filter to modify the HTTP responses generated by your application, with responses typically generated by JSP pages. SiteMesh amends the response based on settings in a configuration file.”

II.3. Content Fetching Mechanism

Content can be created in two ways. For the first way, the system processes raw data and generates useful information. Then the content could be either stored in a database or displayed as a web page. And for the second way, content is created by the users or editors and stored in a database. When content is requested, it is fetched from the database and displayed as a web page.

Content fetching mechanism is implemented and used by the frameworks in the thesis. That’s why we will discuss more about content storage and fetching mechanisms in this document.

Data can be stored in various formats, like as a text file, as a properties or an XML file, or as stored in a database. Most of the times, content data is stored in databases, configuration data is stored as XML files and as properties files.

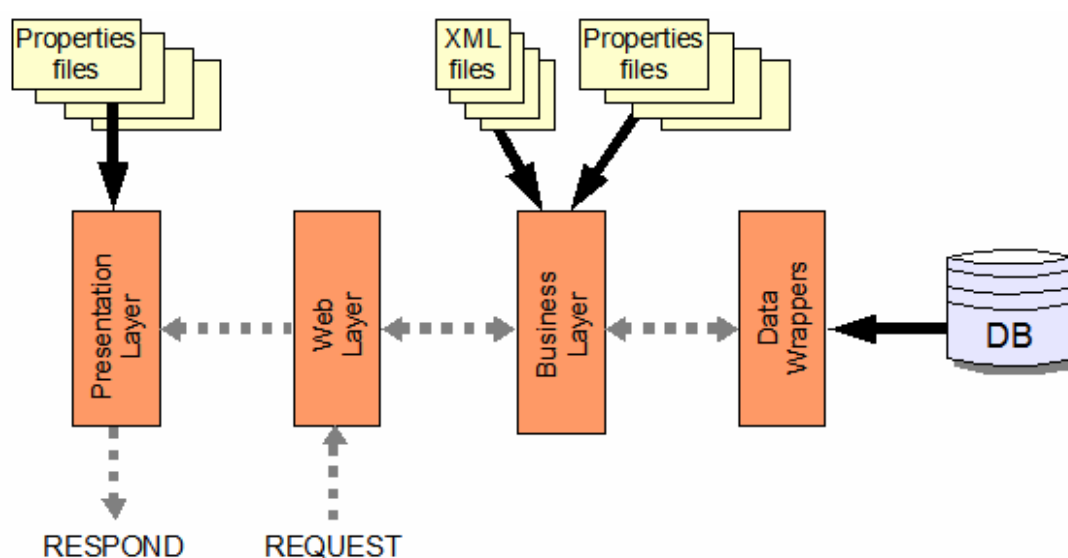


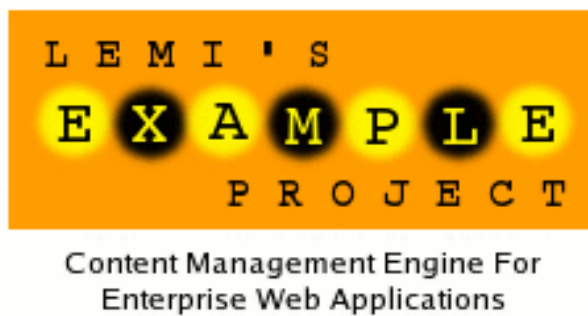
Figure II.8. Different content fetching mechanisms in several architectural layers

Usually the content is stored in databases and it is fetched by data wrappers and transferred to the business layer. Business classes can use XML files and properties

files for configuring the system. Most of the time, configurations determine how the content should be searched and fetched. Then the content is forwarded to servlets in web layer where governs the life cycle of the request. Since all multi-lingual data is stored in properties files, all these information is taken into account by the presentation layer.

PART III

eXaMpLe PROJECT: THE CONTENT MANAGEMENT ENGINE



eXaMpLe Project is a content management engine that provides a set of tools and frameworks in order to develop, maintain, manage and administer content intensive enterprise web projects.

eXaMpLe Project is not a content management system. In contrast, it is a set of frameworks and tools that serve a well-designed environment for both developers and business analysts and make them work as a team.

There are some bottlenecks of developing and managing enterprise web projects that, these appears while there exist a group of people working as different roles in the team. eXaMpLe Project aims to solve these bottlenecks by defining a clear distinction of layers during the development process.

In the following three sections, we try to introduce eXaMpLe Project and its all sub-projects with its architectural information.

III.1. Key Motivations and the Aim of Develop

Key Motivations for developing this thesis are stated as follows.

- Developing time of a portal should be decreased (time is money).
- The component based architecture makes the portal more scalable by reusing components.
- Developing components / modules to a component based portal is easier and faster than developing a one-block application.
- Tools and platforms the software uses should be free.

The aims for developing this thesis are introduced as follows.

- To provide frameworks being developed on open-source platforms by using standards
- To construct an environment that makes portal development faster and more manageable.

III.2. Development in Enterprise

The very first aim that eXaMpLe Project is trying to fulfill is separating the development process into parts, so that different roles can work on their parts independently. For the fact that for medium and high range development groups, a java developer should not develop in database layer or a web interface and should focus on its work.

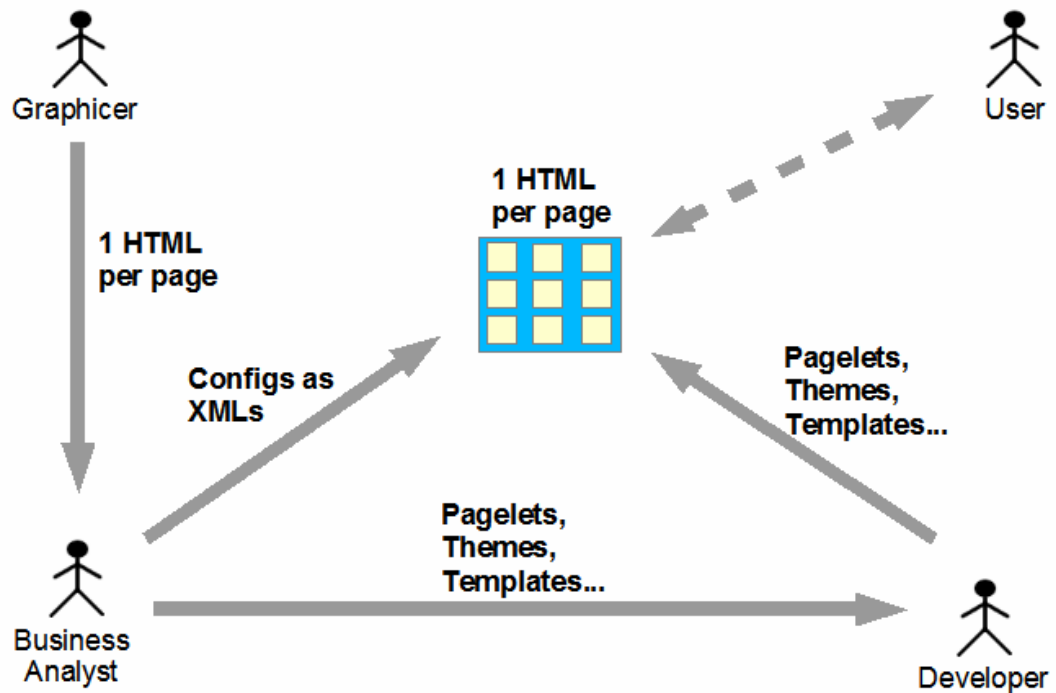


Figure III.1. Developing a single web page by using JEWEL System in a development team

JEWEL System enables us to separate configuring web pages from developing web pages. So, that could lead to work as a team.

III.3. History of Development

Sized projects need sophisticated designs and well-known requirements. However sometimes sized projects emerge from the evolution of a simple developments. eXaMpLe Project has grown by evolving itself for several times. In the history of development, the steps of evolution can be identified.

III.3.1. Development Branch for Pre-Releases

For the development branch for pre-releases, eXaMpLe Project had contained only JEWEL System. JEWEL System has proudly implemented all four web layout generations even though this is not the best way to do it.

- Version 0.1: JEWEL as HTMLs having JSP codes inside (January 2004)
- Version 0.2: JEWEL as JSP pages with includes
- Version 0.3: JEWEL as JSP templates

- Version 0.4: JEWEL as parameterized components containing JSP templates (April 2004)

III.3.2. Development Branch for Releases

In the development branch for production releases, JEWEL was using XML pages for configuring web page layouts. At this branch, QUARRY, JCategoryizer and JConfiX were added to the whole picture.

- Version 1.0: JConfiX is added to the system. JEWEL is now using JConfiX for governing configuration files. (July 2004)
- Version 1.1: JEWEL is now responsible for managing the navigation system.
- Version 1.2: Cache management and session management modules are embedded.
- Version 2.0: JCategoryizer is built onto JConfiX and added to the system.
- Version 2.1: JCategoryizer is now using JEWEL System for managing the presentation layer. (December 2004)
- Version 3.0: QUARRY is build on top of JCategoryizer and added to the system. (April 2005)

III.4. Birds-Eye-View to the Whole System

eXaMpLe Project contains two main sub-projects. These are JEWEL System and QUARRY. JEWEL System is responsible for the presentation layer of the whole project and QUARRY is for providing a flexible framework that manages organizable content.

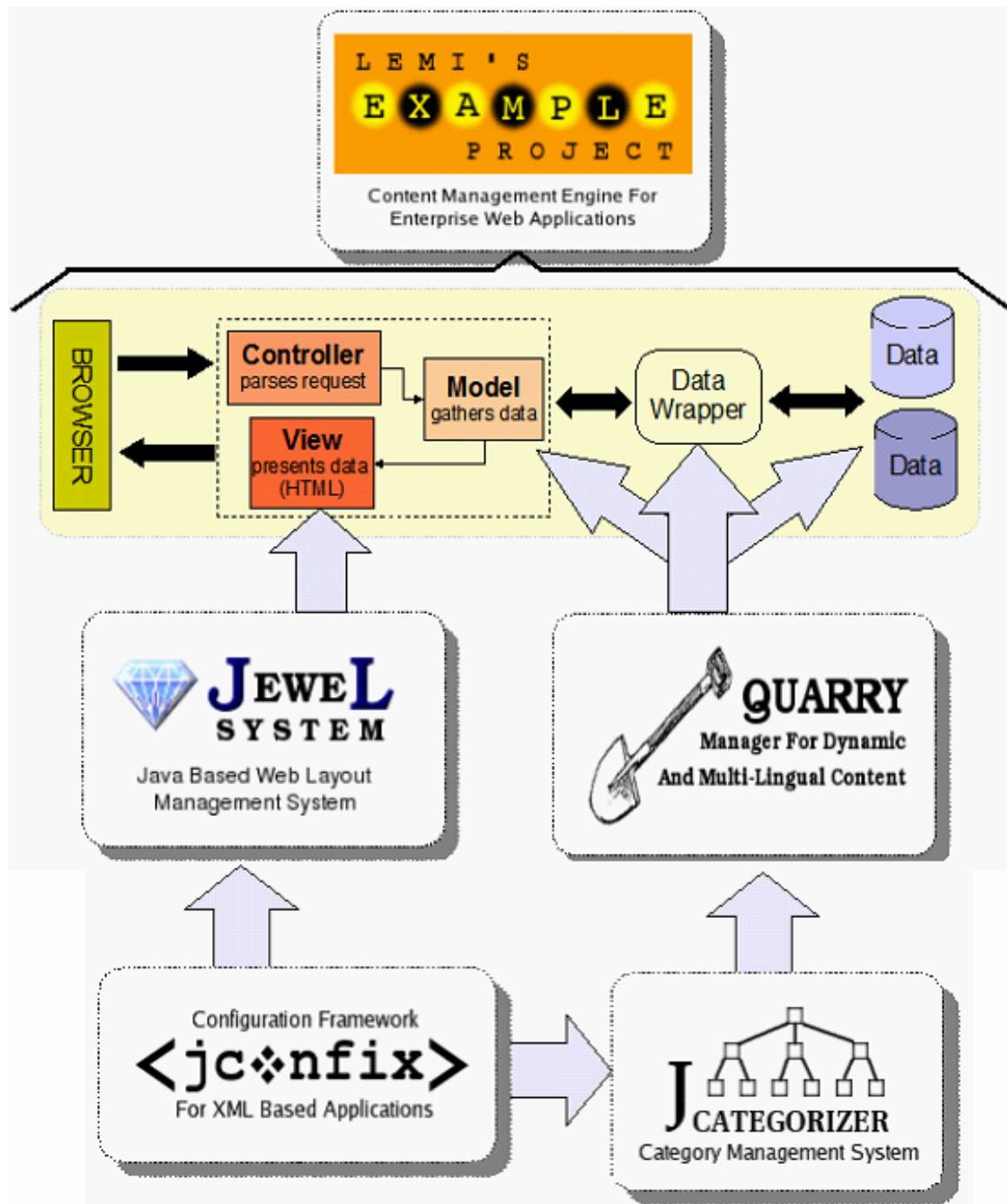


Figure III.2. eXAMPLE Project and its sub-projects in one picture

Both sub-projects are built on top of JConfiX because these are using XML structure for defining the configuration values. JCategoryzer is actually a generic category management system, however QUARRY is built on top of it for categorizing the content.

PART IV

JConfiX: THE CONFIGURATION MANAGEMENT FRAMEWORK



Almost all web application projects having configuration values stored as XML files propose project-dependent solutions in order to access these values. But with JConfiX, we

generated a generic configuration framework and make it accessible by other projects using the same JVM.

IV.1. Key Purposes and Objectives

The main purpose is to develop a framework in order to get all configuration files (as XML) of different projects and let them accessible.

IV.2. Managing the Configuration Files

All projects having configuration XML files first registers their XML files to JConfiX. Registering process is simply adding the locations and filenames of the configuration XML files to the main JConfiX XML file.

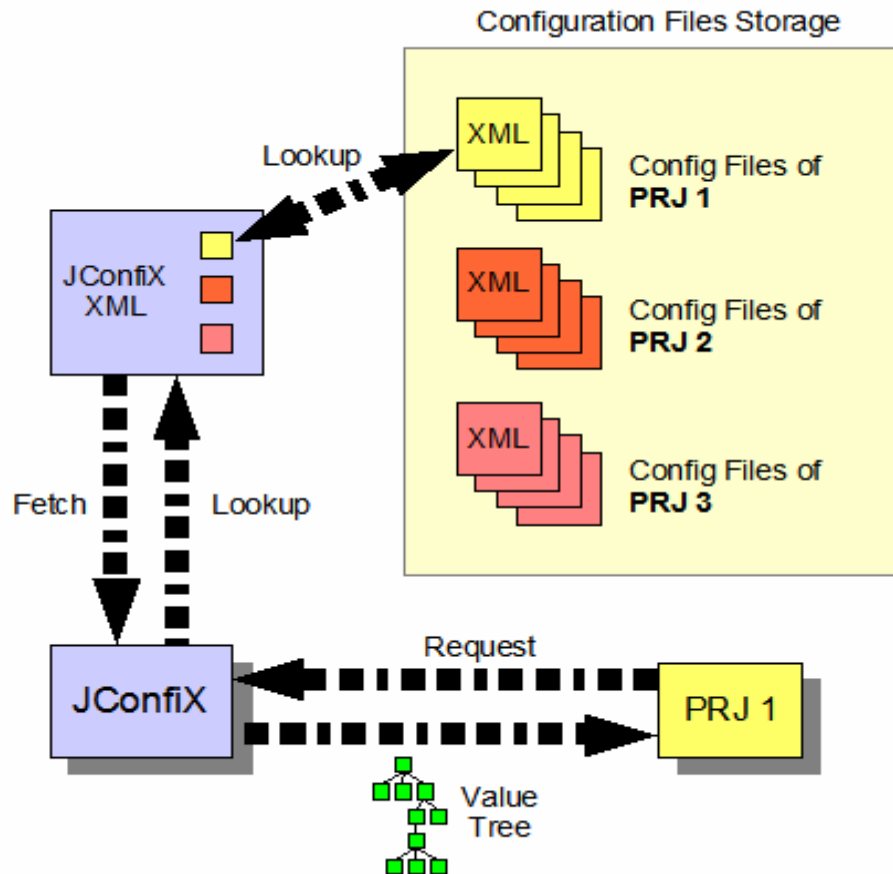


Figure IV.1. Sample workflow of configuration value generation process

After the registering process, all jobs are done by JConfiX. From now on, JConfiX is the main station responsible for serving configuration values. As in Figure IV.1, PRJ1 request to get its configuration values from JConfiX. JConfiX look ups its own XML file in order to get the location of PRJ1's configuration files. When it finds them, it fetches all related XML files and generates a value tree. Value tree resembles a DOM tree and it's the output format that PRJ1 can understand and use. This scenario happens when the web application starts. It means all configuration files are converted into value trees at the beginning of projects' lifecycles (that is the beginning of application server).

IV.3. The Structure of JConfiX

JConfiX has a single XML file that stores location information of all configuration files.

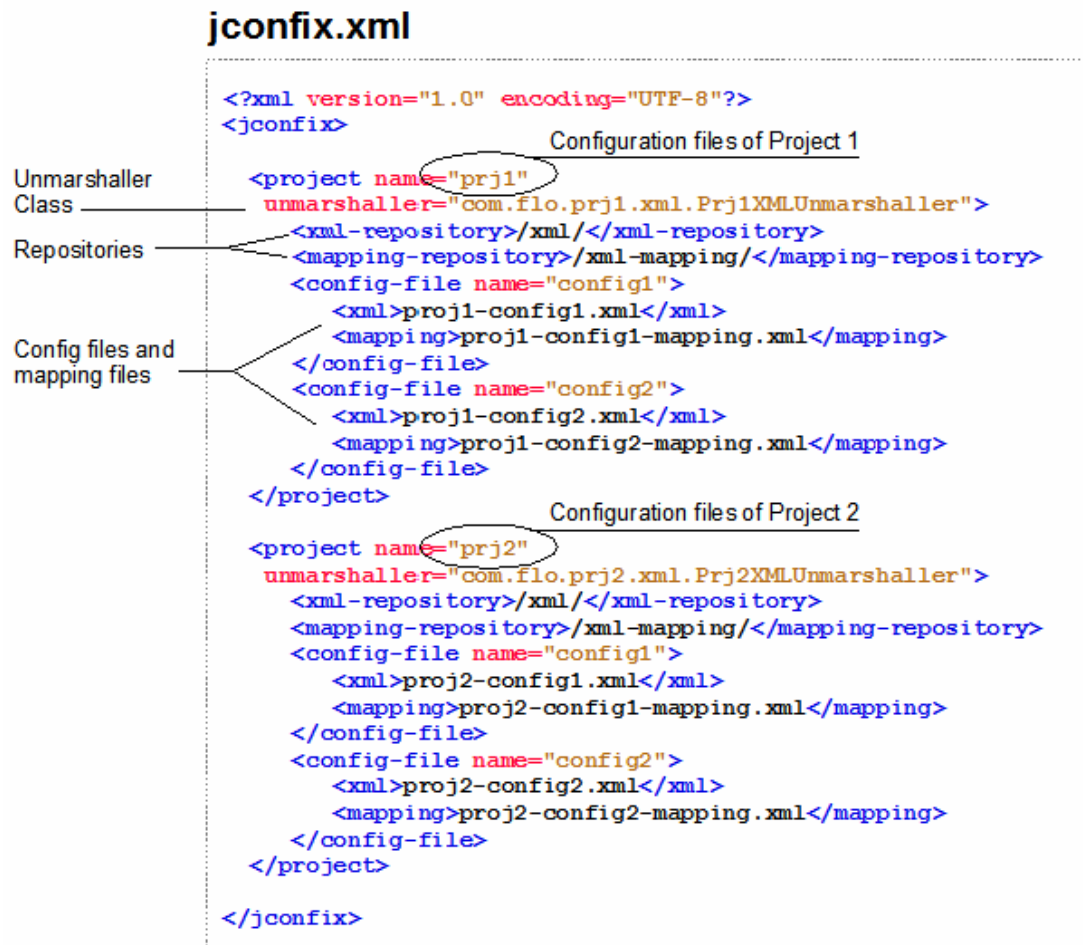


Figure IV.2. The main and the only configuration file of JConfiX

In Figure IV.2, it can be noticed that relative path for XML repository and the repository of the mapping files are defined. JConfiX can access to these registered files and mapping files and constructs value trees. JConfiX needs to define an unmarshaller class by design. Unmarshallers classes are the java classes that do the XML to value tree conversion process. All unmarshallers should extend from predefined JConfiXMLUnmarshaller class.

The relation between JConfiX and the other projects build on JConfiX is defined in Figure IV.3.

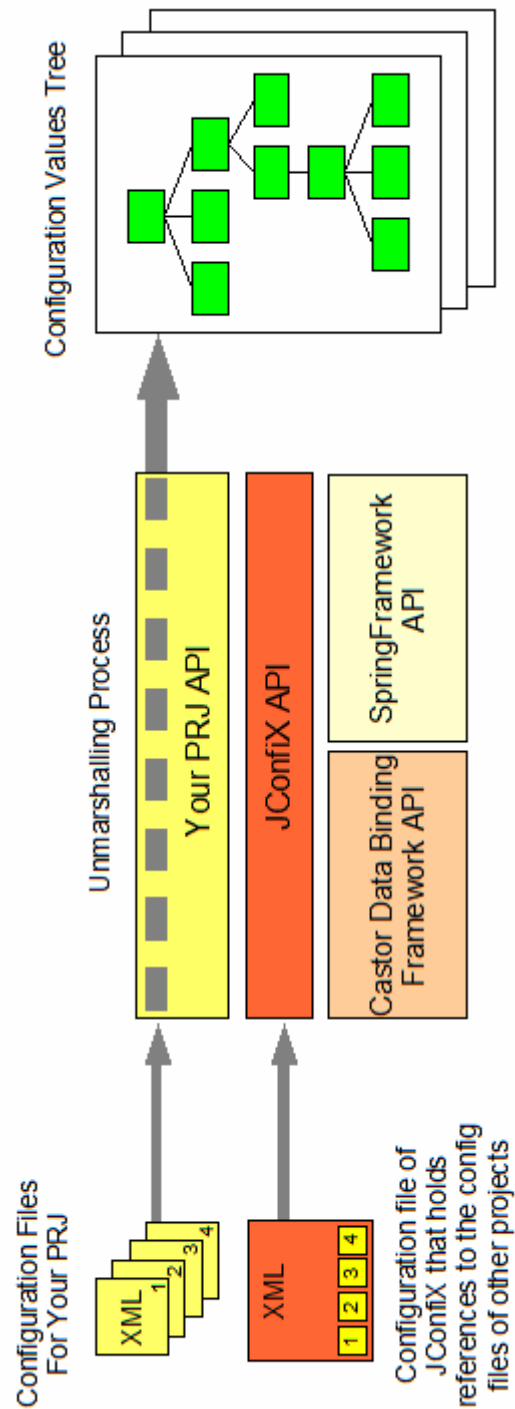


Figure IV.3. Workflow of a any project built on JConfiX

JConfiX is built on top of Castor Data Binding Framework API [CST] and Spring Framework API [SPR]. Developers should extend their unmarshaller class from JConfiX main unmarshaller class. Therefore, all project based configuration files are actually converted into value trees by the unmarshaller class of our project (i.e. PRJ in Figure IV.3).

PART V

JEWEL SYSTEM: WEB LAYOUT MANAGEMENT FRAMEWORK



Jewel is a Layout Management Framework that builds on the "include" feature provided by the Java Server Pages specification to provide a full-featured, robust framework for assembling

presentation pages from component parts.

Jewel is a Layout Management Environment that parses the configuration of each web page and gives ability to modify and customize theme and view of each web page. By using JEWEL System, a developer doesn't need to determine layout scheme. Developing and configuring processes become different tasks to be implemented.

The features can be listed briefly as below.

- 11 XML-based configuration files
- 2 parameters (action, role) determine all layout!
- View (Screen) definitions
- Extension mechanism for views: A view can extend from other views
- Pagelets: Page parts (as Tile)
- Templates for layouts, pagelets and navigations
- Dynamic page building: Layout Management
- Reuse of Pagelets / Components
- Internationalization (I18N)

- Authorization of actions: Role Based Customization Mechanism
- Navigation Management
- Layout Caching Mechanism
- Embedded Session Management

V.1. Key Purposes and Objectives

The key purpose is to provide a framework that governs all processes of the presentation layer of the system and separates the content from layout, so that developing an enterprise wide web portal manageable by different roles.

V.2. Multi-Layered Presentation Layer

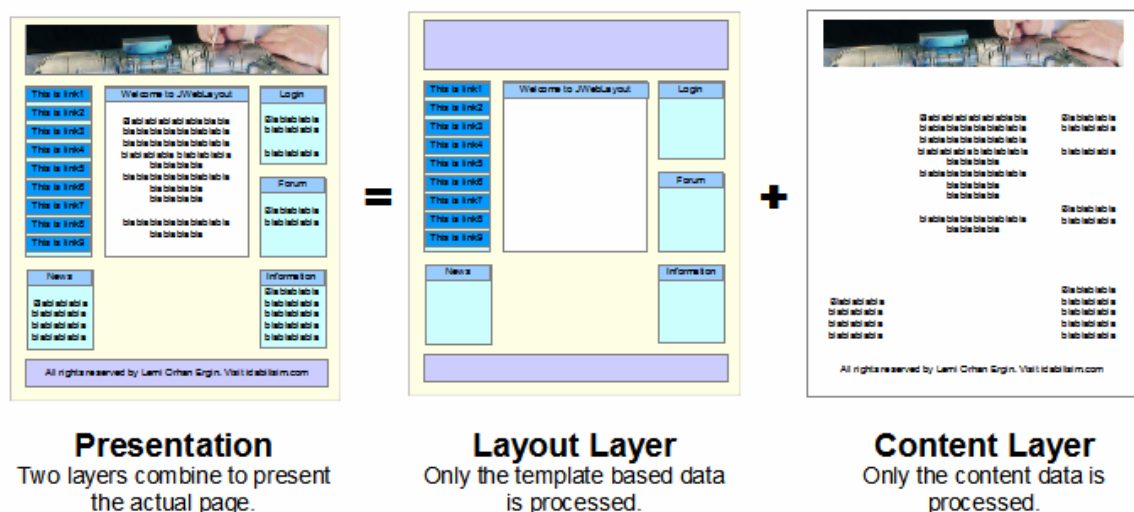


Figure V.1. Presentation layer spitted into two sub-layers

One of the basic enhancements that JEWEL System is proposing is a multi-layered structure of presentation layer. In most of the current J2EE frameworks, presentation layers use both layout data and content data. However, this mess the presentation code. Content and layout code forms a compound. But now, JEWEL System is proposing a way to fetch content and layout data one by one and let it display together.

V.3. Terminology

- **Layout Data:** A generated XML storing all the information about layout of the current view.
- **Html-Template:** Template that defines the general properties of the view, such as style sheets, JavaScript files and meta information.
- **Grid or Layout:** Placeholder / container for pagelets and other grids.
- **Cell:** Each part of the grid having the ability to hold a pagelet or a grid.
- **Pagelet:** Individual content container. (as a Tile in Struts-Tiles project)
- **Content:** Dynamic or static data that the entire layout is being prepared to serve it.

V.4. Integrating a Web Page

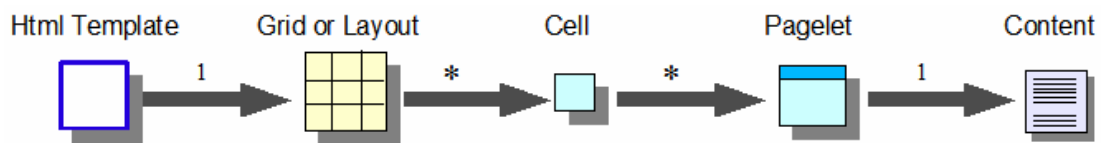


Figure V.2. Automaton-like flow of web page integration

In Figure V.2, generating a web page by integrating the layout layers can be accessed.

- An html template can hold only a grid. For special cases, an html template can hold pagelets.
- A grid (or a layout) has many cells in it. Actually grid resembles a simple html table.
- Each cell can have unlimited number of pagelets. A pagelet is the main and the only container for contents.
- A pagelet can hold only a content page, namely a jsp page.

We can view the integration steps in another way as follows (Figure V.3):

When a request comes to the dispatcher servlets of JEWEL System, the rights of the current user is checked by the rights management module of JEWEL System. The system determines whether the current user has rights to implement this action or not. Then layout management module generates a map object holding layout

information. If the user has rights, the layout information reflects the requested page structure. Else, the map stores the information of an error page.

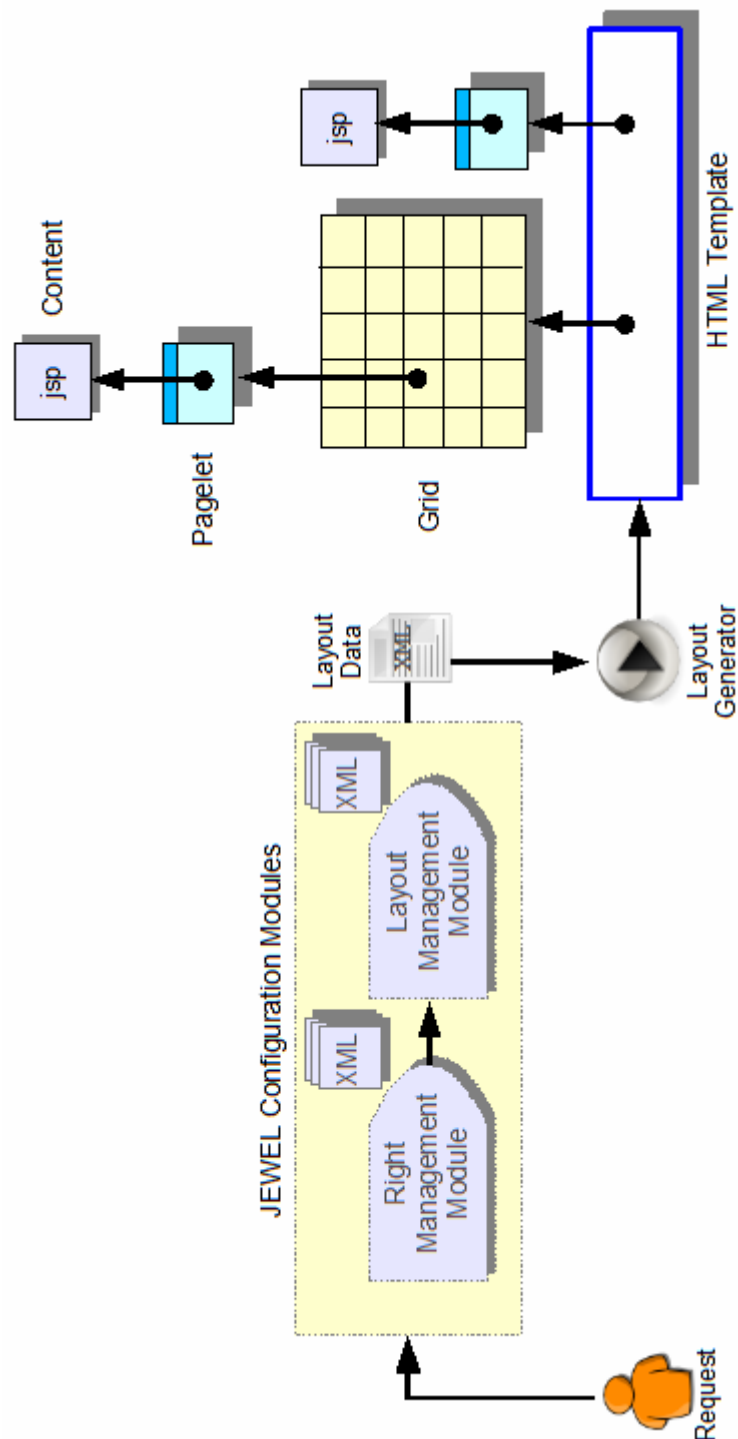


Figure V.3. The whole picture of web page integration with filtering processes

Layout data, i.e. simply a map object, transferred to the layout generator page. This page is simply a jsp page, but it is specialized in parsing the layout map and

combining the layout and the content by following the instructions stated in that map object. When integrating the web page is finished, it is responded to the user.

V.5. Role Based Customization Mechanism

JEWEL System provides a role based rights management system that controls the security of all architectural layers. At the very first stage, whether the current user has rights to implement the action is determined by the dispatcher servlet of rights management module. This filtering process uses a high level approach, that's why a second security mechanism is taken into account: Rights management in web layer. Although a user has rights to implement an action, it does not mean that, that user can access and execute all methods of the business logic corresponding to the requested action. By the help of the filtering mechanism in web layer, developers can forbid any classes from calling any method of the others.

JEWEL System also provides security in presentation layer. Like the security in web layer, developers can block the display of a web page section at any time. Even if a user can achieve to implement an illegal action, he cannot see the results in web pages. If you call *hasRight* method of the rights manager class with an array of action names and the current role as parameters, it will give you a true/false value for each action. This type of rights checking lets us disable/hide any portions of web page.

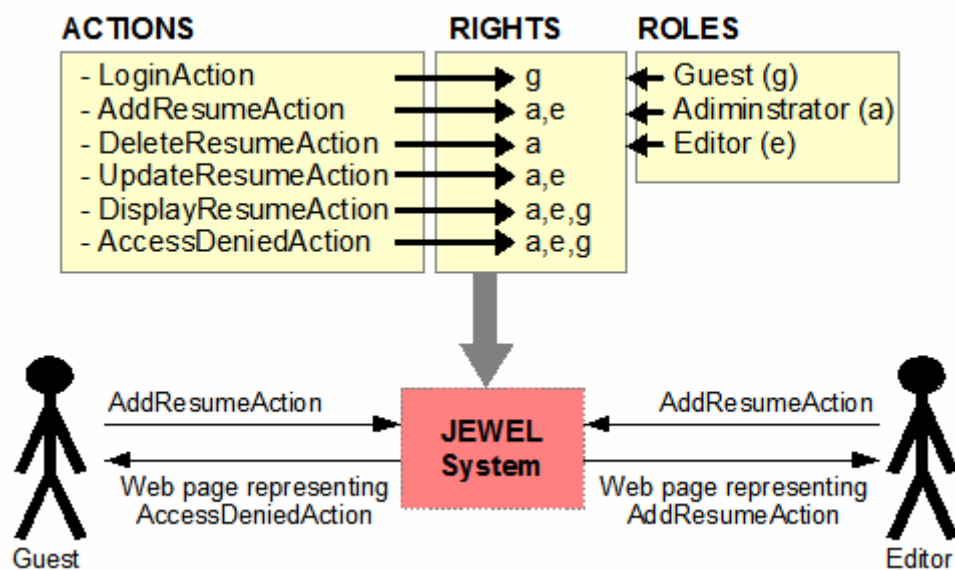


Figure V.4. Sample action and the responses of JEWEL System

As in Figure V.4, all roles and actions should be entered to the system. That will let us gain full control over the request flow and protect from any unexpected security holes. Rights inside rights.xml file are nothing more than a bunch of matching between roles and actions. That helps us to gain full control over page requests and protect. Then which roles are allowed to access which actions are determined.

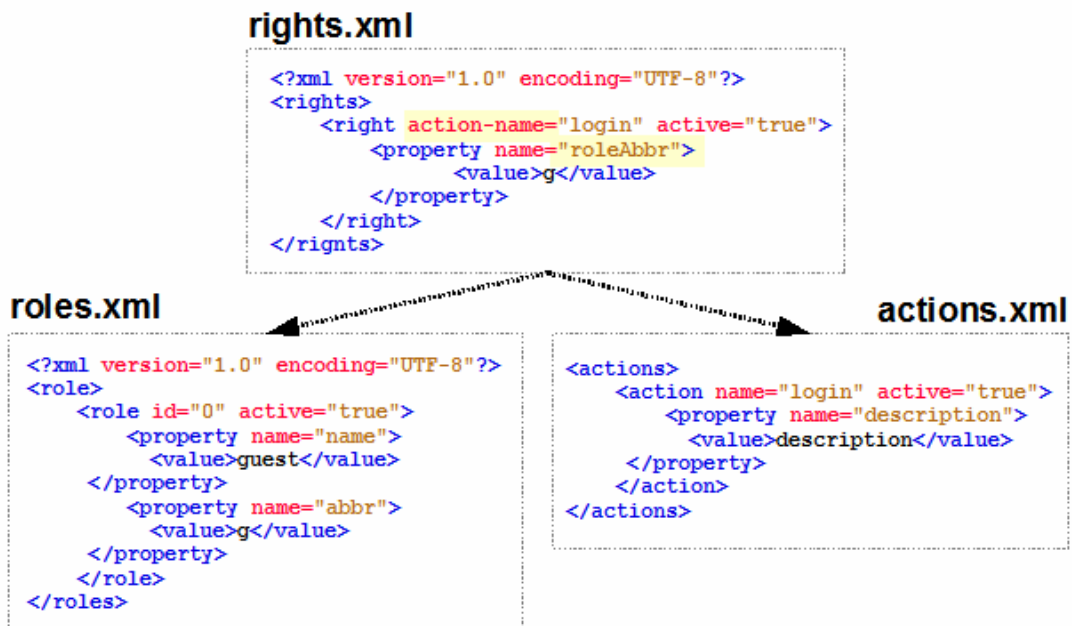


Figure V.5. Configuration files for rights management process

The details of the configuration files can be seen in Figure V.5. It can easily be noticed that the structure of these files is very simple and manageable. By looking at roles.xml, rights.xml and action.xml files above, JEWEL System determines which role has rights to implement which actions.

V.6. Navigation Management Module

Navigation system of the whole web site is governed by JEWEL System. In JEWEL System, we can define many navigation pagelets having unlimited number of links, even in different templates. The developer just needs to configure the navigation system to the navigations.xml file. The list of features of the navigation management module is listed as follows.

- Multiple navigation systems for the same web site are supported.

- Each role can display the links that allowed by the system. One role cannot access to the other role's link.
- Link names can be multi-lingual. That's why link names are stored in jewel.properties file. That lets us to support multi-lingual links.
- Link URLs are stored externally in jewel-static.properties file. That lets us not to hardcode the URLs and makes the navigation system more flexible.

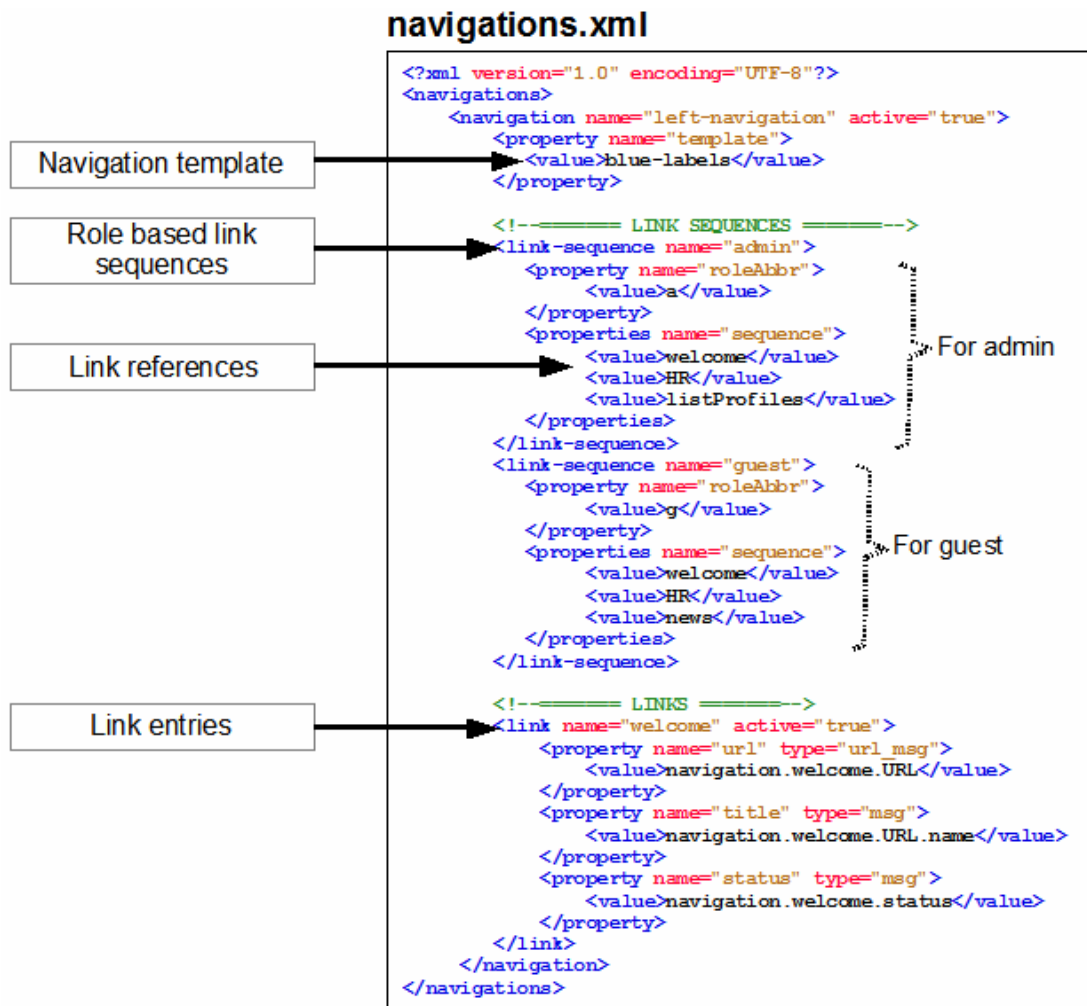


Figure V.6. Configuration file for navigation management

Above, you can notice that each role has different set of links. Different templates allow us to display the same navigation system in different styles.

V.7. Configuring Layouts

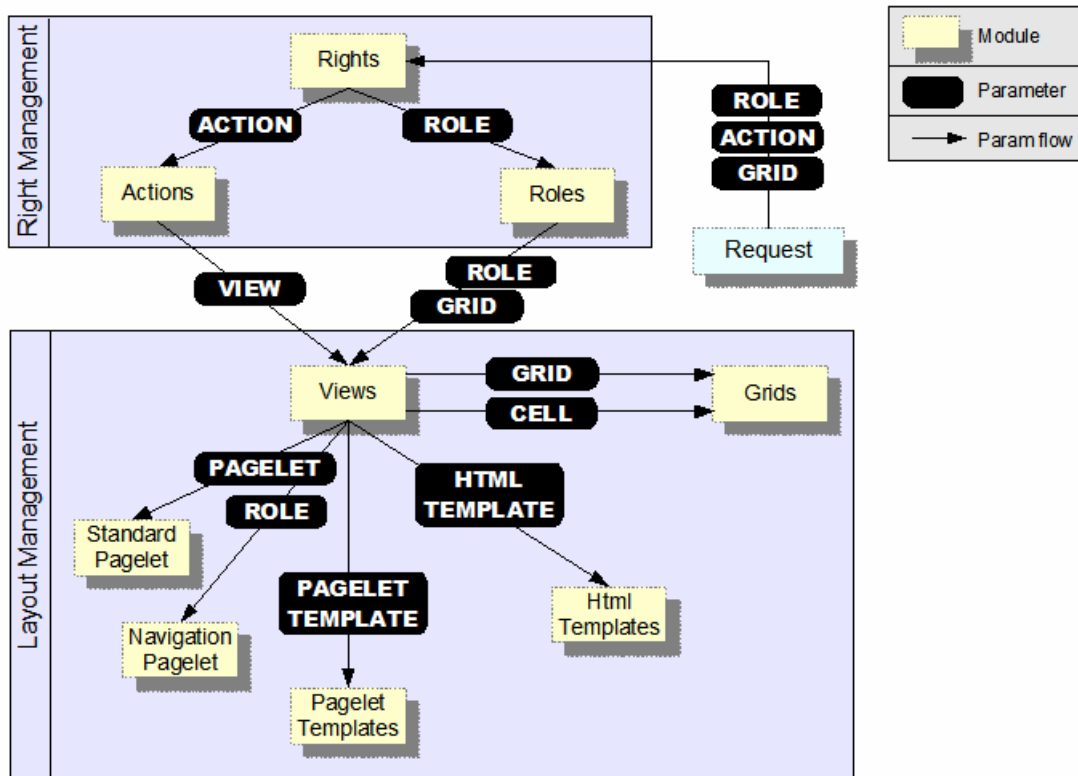


Figure V.7. Parameter flow between managers in JEWEL System

There is a high traffic between JEWEL Managers which are responsible for governing each structural element, such as grids, views and pagelets. In Figure V.7, we can determine which parameters are required for preparing which structural element.

In JEWEL System, there exists precedence while developing new pages (Figure V.8). For example, the action should be defined before the web page layout. Taking the precedence into account lets us develop page layouts in a proper way.

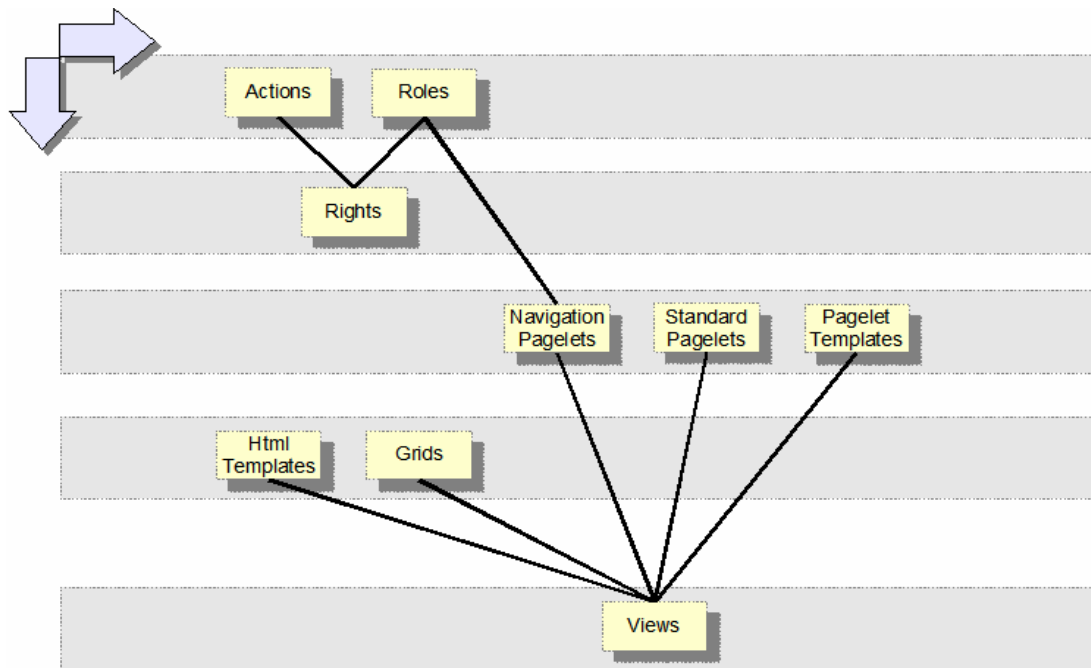


Figure V.8. The precedence of developing configuration files

In Figure V.8, the order of configurations that we should develop is as follows: Actions, Roles, Rights, Navigation Pagelets, Standard Pagelets, Pagelet Templates, Html Templates, Grids, and Views.

V.8. Embedded Cache Management

Since there exists hundreds of different layout data defined in the system and layout data generation process is not atomic, JEWEL System caches all different layout data when requested. That's why, if a layout is called for the first time (that is normally after an application server restart), the response time will be a little bit longer than normal. However, this makes the system cache the layout data and let the latter responses faster.

One additional feature could be a flushing the memory mechanism. Administrator can flush the cache at any time by entering a special URL from the browser.

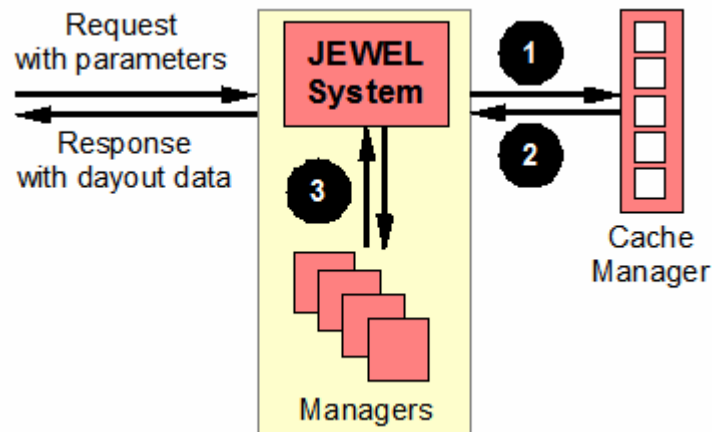


Figure V.9. Caching mechanism of JEWEL System

In Figure V.9, a sample workflow of caching mechanism is introduced. By design, layouts are requested by a set of parameters from JEWEL System. JEWEL System processes the parameters and applies to the cache manager. If the requested layout is inside the cache, cache manager serves it to JEWEL System (Step 1 and 2). Otherwise, it responses a MISS instead of a HIT.

If JEWEL System hits, then it responses this cached layout data to the requester. Else, if JEWEL System misses, then it forwards the parameters to its layout managers and let them generate the layout data (Step 3). After fresh layout data is generated, it is cached for latter requests and served to the requester.

V.9. Managers: Business Layer of JEWEL

There exists several configuration files and they're needed to be processed in the business layer. JEWEL has a managers API and a set of managers to process XML configuration files. The managers of JEWEL System are listed as follows:

1. JEWEL Manager: Governs all processes through layout generation workflow.
2. Action Manager: Manages action.xml file.
3. Rights Manager: Manages rights.xml and governs all rights management process. It also governs action and role managers to let them work correctly.
4. Role Manager: Manages roles.xml file.

5. Layout Manager: Manages layouts.xml file and is responsible for providing correct layout templates to the system.
6. Pagelet Manager: Manages pagelets.xml file and is responsible for providing correct templates of pagelets and navigations.
7. Navigation Manager: Manages navigations.xml file.
8. View-Definition Manager: Manages view-definition.xml file and governs the combination process for pagelets into layouts.
9. Action-View Matching Manager: Manages action-view-matching.xml file and determines which layout can represent which action.

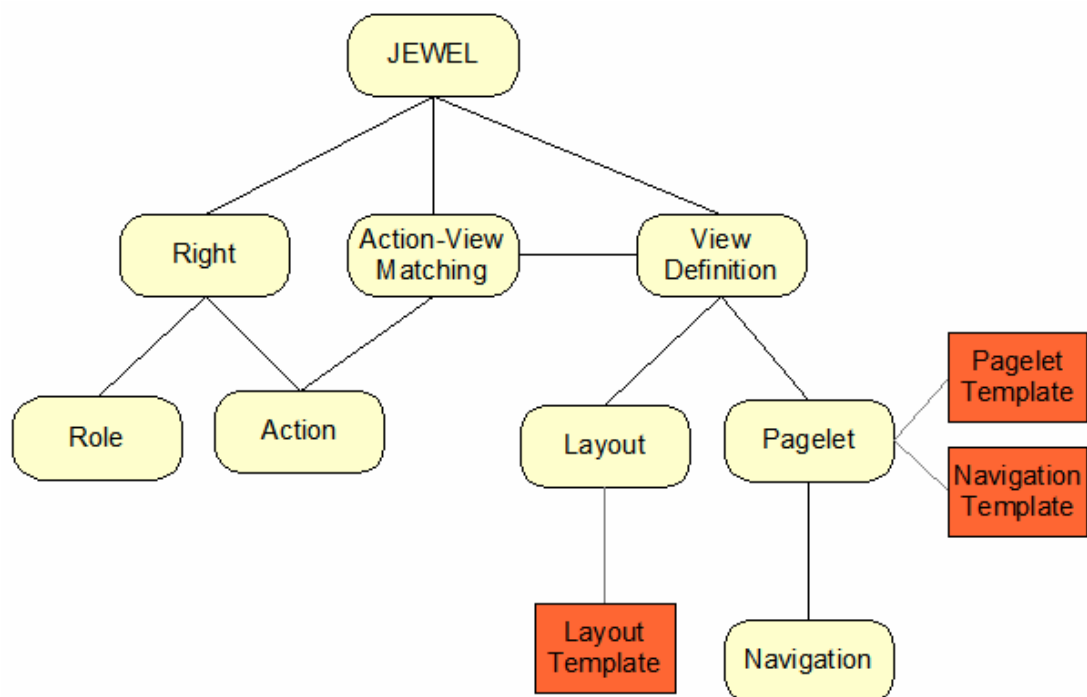


Figure V.10. Interactions between managers in JEWEL System

Each manager handles its specialized duty and works with other manager classes harmonically.

V.10. Workflow of JEWEL

The process of generating a layout for an ordinary request becomes a longer process than the usual workflow when we govern the process by JEWEL System. However, it serves us a better modularity and configurability for enterprise web applications.

As stated before, there are a set of managers working in harmony while generating layouts from requests. Below, you will find how JEWEL System handles two kinds of http requests.

V.10.1. Generating a Layout from an Ordinary Http Request

The numbers below indicate the step numbers on the graph. You can trace the flow by following the arrows and reading the corresponding step number on the graph.

1. A request comes as login.html to the dispatcher servlet of JEWEL System with action and role parameters. Here is the main gate where requests are welcomed.
2. The request is forwarded to a JEWEL Front Controller which is responsible for layout management actions (Layout Controller).
3. By using these parameters coming from the request, Layout Controller calls JEWEL Layout Manager for generating the layout data.
4. Layout data is prepared and sent back to Layout Controller.
5. Only layout data is available for now. Then layout data is sent to layoutGenerator.jsp which is responsible to combining a single jsp page from many individuals. Layout generator processes the data.
- 6-7. By runtime-includes, all parts and sections of the web page start to form a single page. Since the layout data does not hold any content data, all pagelets attempt to get the corresponding contents. Therefore, request is redirected to JEWEL Front Controller which is responsible for user management actions (User Controller) in order to fetch the content of login pagelet. As all requests, this request must pass through the dispatcher servlet.
8. The content of login pagelet is gathered by User Controller. Then the data is transferred to layoutGenerator.jsp again.
9. After combining login pagelet and all others, whole html is generated and served to the user.

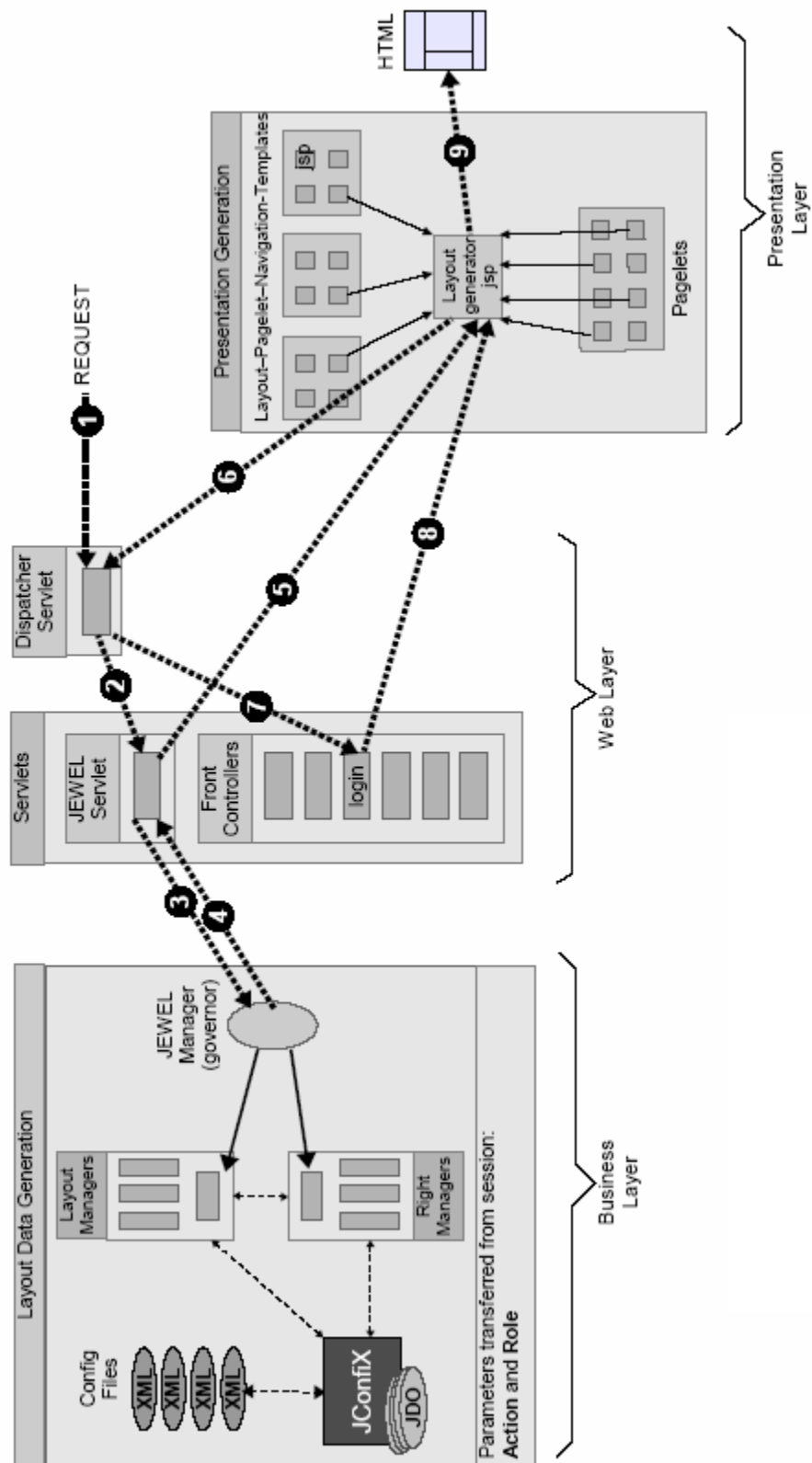


Figure V.11. Handling a simple http request

V.10.2. Generating a Layout from an Http Request after Form Submission

The numbers below indicate the step numbers on the graph. You can trace the flow by following the arrows and reading the corresponding step number on the graph.

1. A request comes from a form from the login page. It arrives first to the dispatcher servlet of JEWEL System with action and role parameters. Here is the main gate where requests are welcomed.
2. Since the request is directly sent to login pagelet, the request is forwarded to a JEWEL Front Controller which is responsible for user management actions (User Controller).
3. Login controller governs necessary business operations. As in the example, username and password is confirmed as valid and correct. Therefore the content for login pagelet is prepared and sent back to the dispatcher servlet. Since the user is now logged in the system, the request is redirected to welcome action.
- 4-5. The welcome action is implemented. Action and role parameters are sent to JEWEL manager to make it generate layout data
6. Layout data for welcome action is prepared and redirected to the layout controller
7. Layout data is sent to layoutGenerator.jsp. Layout generator processes the data.
- 8-9. By runtime-includes, all parts and sections of the web page start to form a single page. Since the layout data does not hold any content data, all pagelets attempts to get the corresponding contents. Therefore, request is redirected to JEWEL Front Controller which is responsible for content management actions (Content Controller) in order to fetch the content of welcome pagelet. As all requests, this request must pass through the dispatcher servlet.
10. The content of welcome pagelet is gathered by User Controller. Then the data is transferred to layoutGenerator.jsp again.
11. After combining login pagelet and all others, whole html is generated and served to the user.

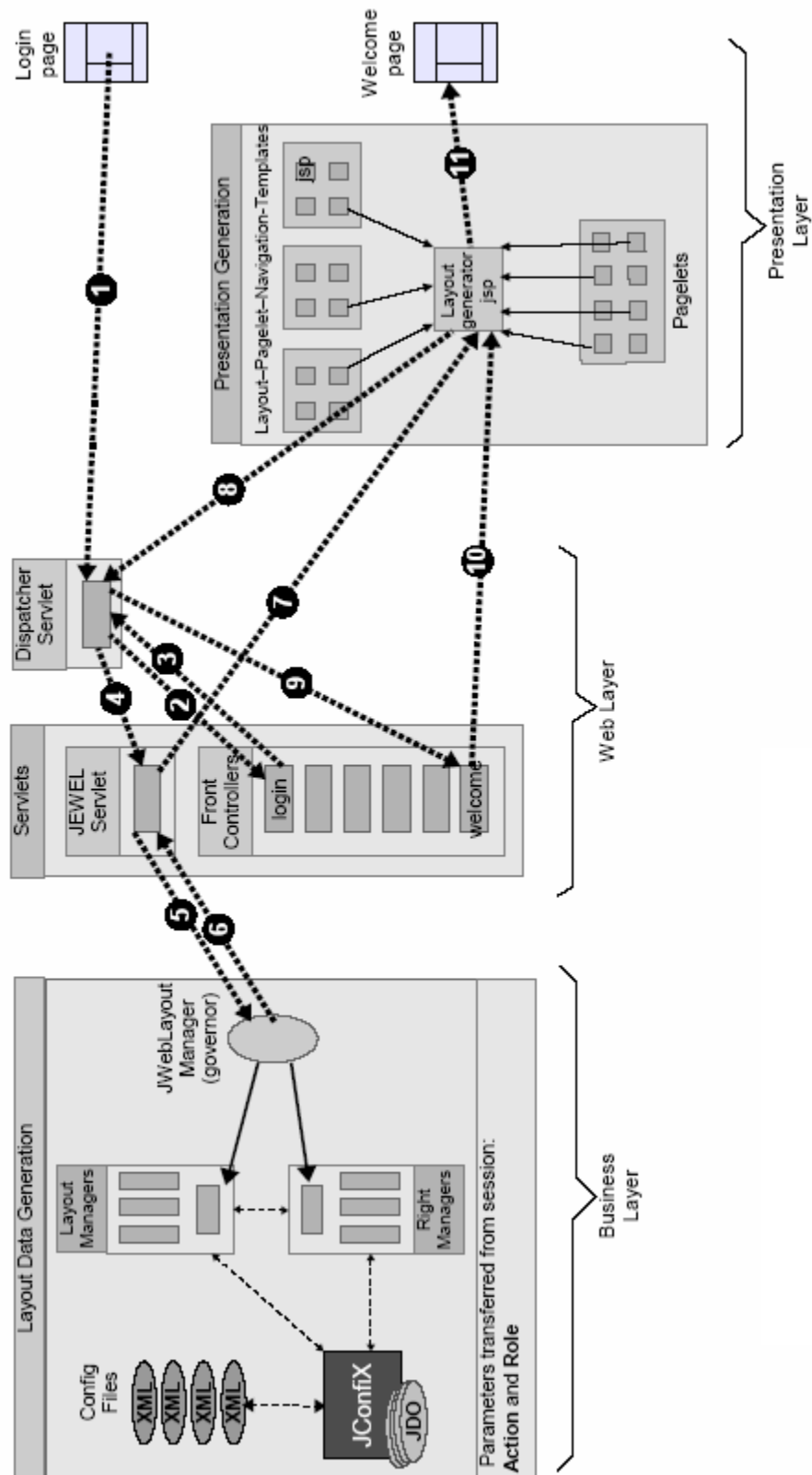


Figure V.12. Handling an http request after a form submission

PART VI

QUARRY: MANAGER FOR DYNAMIC AND MULTI-LINGUAL CONTENT



QUARRY is responsible for fetching requested content from a content storage, especially from a database. In order to dig into the details of the structure and the work principles, the packages that

QUARRY is build on should be introduced. QUARRY is not only a content management package of the whole system, but also a tailored sample showing how JClassifier (Category Management Framework) can extend.

VI.1. Key Purposes and Objectives

QUARRY is designed for providing to developers a flexible framework for serving organizable content. The main difference from other similar frameworks is that QUARRY is build on top of JClassifier. From later on, we will put JClassifier into the discussion whenever we call QUARRY.

VI.2. JClassifier: Category Management Engine



There exists an excessive amount of data being served in Internet today. We have been consuming a big portion of our time for searching data. We usually use category infrastructures of search engines like

Google Directory and Yahoo in order to access a specific data organized into categories. From search engines perspective, almost all data can be put into categories, not matter of it type. Actually, this postulate is the starting point that leads us to develop a category management engine.

Since each content can be organized into a category, we aimed to build our content management module onto eXaMpLe Project's category management framework: JClassifier.

The aim of JClassifier is to organize any kind of data. It could be a product data to be organized, or a content data containing texts and pictures, as usual. It aims to work likewise some very famous web sites serving categorized, organized, huge amount of data; like Amazon.com.

Its key features are:

1. Organizing category structure is very easy and flexible.
2. Easy to limit / Easy to expand.
3. Provides to represent almost all kind of tree structures.

VI.2.1. Architectural Overview

JClassifier is build on top of JConfIX and it is developed by using Java and XML technologies. Configuration files are parsed and served by JConfIX API. As in Figure VI.1, you can extent your project (in this thesis, we construct QUARRY as our project API) on top of JClassifier. One important point is that all categorizable items should be stored inside a database. So JClassifier can access to the database server and fetch the requested items.

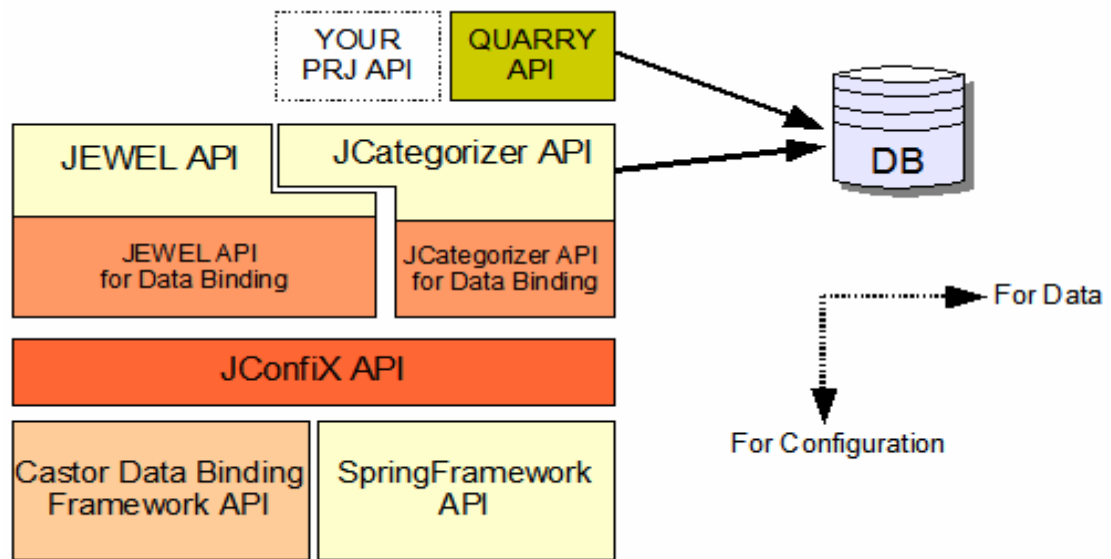


Figure VI.1. Build structure of QUARRY and JCategoryizer

In order to use all advanced JEWEL features from JCategoryizer and QUARRY, the presentation layer of JCategoryizer is build on top of JEWEL API. This allows us to integrate with JEWEL System seamlessly.

As you may notice, JCategoryizer is only responsible for preparing a tree structure for categories and for making the item beans accessible by the managers. The developer is the one who should fill the items beans with data from (preferably) a database. However, despite this fact, JCategoryizer provides some well-known methods to the managers, like *getItemByID()* or *getItemByName()* methods by interfaces.

VI.2.2. Configuring JCategoryizer

JCategoryizer can be configured by three XML files. It allows you represent any kind of category hierarchy storing any kind of data. All category hierarchy and the category-item relations are stored inside a database.

itemType.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<itemTypes>
  <itemType id="1" name="productItem" active="true"
    manager="com.flo.manager.ProductManager"
    bean="com.flo.bean.Product" >
    <max-hierarchy>1</max-hierarchy>
  </itemType>
  <itemType id="2" name="contentItem" active="true"
    manager="com.flo.manager.ContentManager"
    bean="com.flo.bean.Content" >
    <max-hierarchy>1</max-hierarchy>
  </itemType>
</itemTypes>
```

Figure VI.2. A configuration file of JClassifier storing item types

In the configuration file for items, you can determine how many parent categories an item is able to be linked. In the sample above, each item (i.e. products and contents) can have only one parent category. In other words, one item cannot be placed under more than one category.

By design, the manager and bean class of each item type should be defined. The item managers extend from predefined `com.flo.jclassifier.ItemManager` class and item beans extends from predefined `com.flo.jclassifier.Item` class. We gain an extra flexibility and stability by extending.

categoryType.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<categoryTypes>
  <categoryType id="1" name="productCategory" active="true">
    <maxParent>5</maxParent>
    <maxChild>2</maxChild>
    <maxDepth>10</maxDepth>
    <nonTerminalCategoryHasAnyItem>true</nonTerminalCategoryHasAnyItem>
    <maxItemPerCategory>1000</maxItemPerCategory>
  </categoryType>
  <categoryType id="2" name="contentCategory" active="true">
    <maxParent>5</maxParent>
    <maxChild>10</maxChild>
    <maxDepth>10</maxDepth>
    <nonTerminalCategoryHasAnyItem>true</nonTerminalCategoryHasAnyItem>
    <maxItemPerCategory>5000</maxItemPerCategory>
  </categoryType>
</categoryTypes>
```

Figure VI.3. A configuration file of JClassifier storing category types

Categories can be defined to `categoryType.xml` and it allows us to configure a high range of features. These features are as follows.

1. A category type can be disabled by setting its *active* attribute as false.

2. *maxParent* feature defines the maximum number of parents that a category can have.
3. *maxChild* feature defines the maximum number of child categories that a category can have.
4. *nonTerminalCategoryHasAnyItem* feature defines whether a non-terminal category (i.e. having child categories) can have items or not.
5. *maxItemPerCategory* features defines the maximum number of items a category can have.

typesMatching.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<typesMatching>
  <match active="true">
    <categoryType>productCategory</categoryType>
    <itemType>productItem</itemType>
  </match>
  <match active="true">
    <categoryType>contentCategory</categoryType>
    <itemType>contentItem</itemType>
  </match>
</typesMatching>
```

Figure VI.4. A configuration file of JClassifier storing the matchings between items and categories

This configuration file allows us to match the categories with the items. Here, we can also match different kinds of items to a single category. For example, book items, audioDevice items, videoDevice items can be attached to an eStore category.

VI.3. The Structure of JClassifier and QUARRY in MVC Architecture

JClassifier stores all relations between items and categories in a database in predefined tables. QUARRY also stores content data in a database, but in user defined tables. When a request for a content comes, JClassifier finds its category and QUARRY finds its data. We can summarize the interaction between JClassifier and QUARRY like that as in Figure VI.5.

Since JEWEL System welcomes all requests and governs the page layout process, it communicates with JClassifier whenever a content request comes.

When JEWEL System gets the content data and it displays in a predefined web page layout.

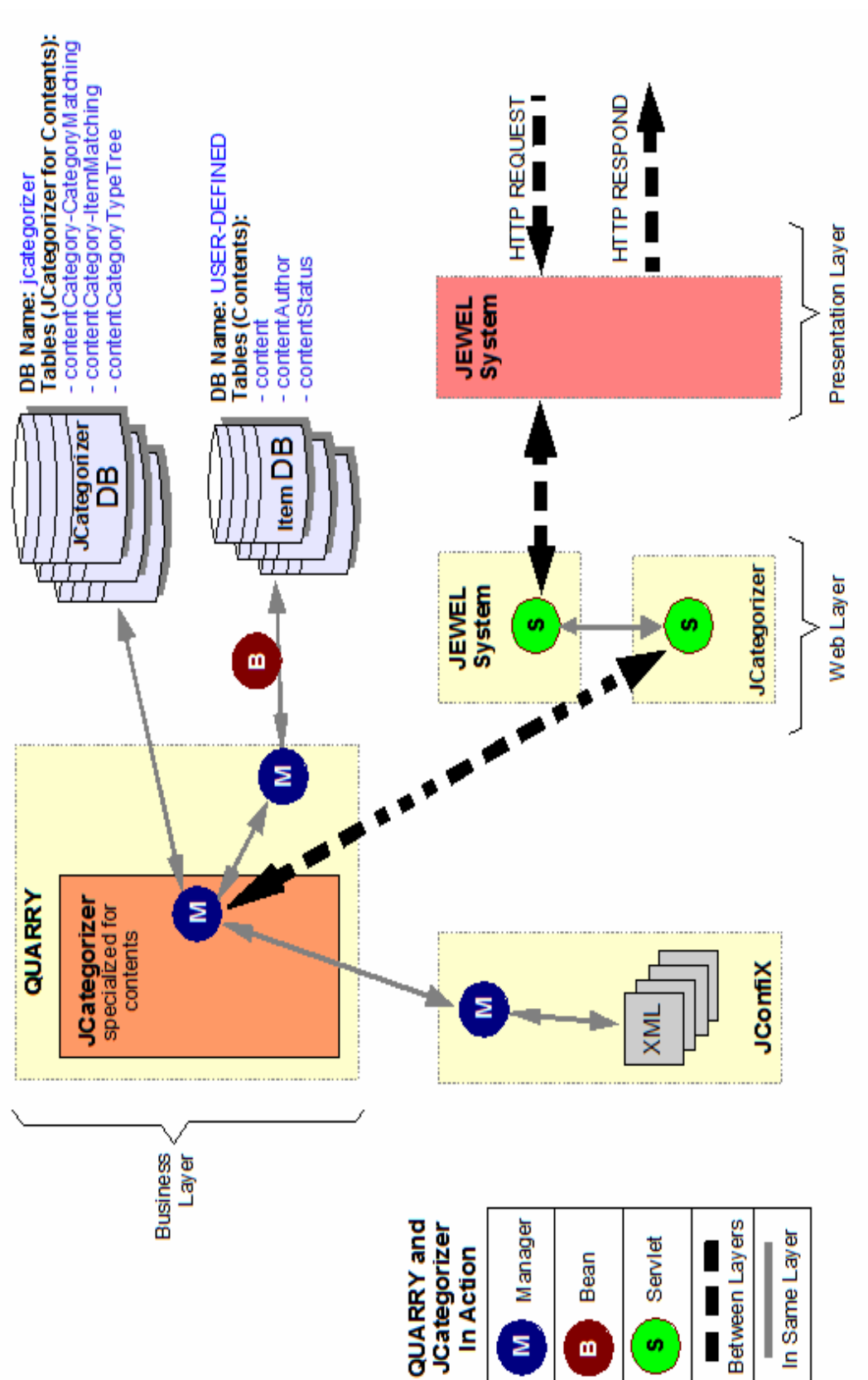


Figure VI.5. The Structure of JCategoryizer and QUARRY in MVC Architecture

When a request comes to JEWEL System in order to fetch content, it simply redirects the request to JClassifier Controllers (Servlets). JClassifier servlet interacts with QUARRY directly, with JClassifier indirectly. JClassifier module of QUARRY fetches the category information from the database. At the same time, QUARRY gathers the content and content related information from the database. JClassifier uses JConfX in order to configure the system both fills the beans with data and serves them to the servlets. Finally servlets response to the requester by these beans.

PART VII

CONCLUSIONS AND FUTURE WORK

This thesis aims to find answers to the issues presented in the introduction. We have proposed a set of frameworks in order to develop content intensive web applications. Each framework is designed to fulfill a necessity in content management systems and in enterprise java applications areas. Also eXaMpLe Project lets people work as a team by distinguishing the roles and determining these roles jobs.

JEWEL System offers a promising environment and a set of enhancements to J2EE developers. In JEWEL System, we have developed a featured framework in web page layout management area. Besides providing an API to the developers, it proposes a new way of web page creation. JEWEL System which is the layout management framework of eXaMpLe Project, uses 11 XML files in order to support navigation management, user management, role-based security and dynamic layout management. Although two famous competitors of JEWEL System became mature in their lifecycle in these days, JEWEL System is open to improve in design and in coding by using new technologies. That's why JEWEL System can offer additional features and new enhancements on its new releases.

QUARRY is not only a content management engine, but also a very convincing sample showing JClassifier's capabilities and features. QUARRY is able to use the "organizable and categorizable content" concept by building over JClassifier. JClassifier lets us build new frameworks by using "organizable items" concept for categorizing books, items, products, articles, etc. Once again, JClassifier can offer new enhancements and new frameworks in its new releases.

JConfiX is a small framework developed for handling all configuration files of several different frameworks from one place. JConfiX is mainly using Castor Data Binding Framework. However, new technologies like JDOM offer improvements in performance and flexibility. JConfiX could be rewritten from scratch by these new technologies for the next releases without affecting the other frameworks.

REFERENCES

Books

- [CAV04] Chuck Cavaness, *Programming Jakarta Struts*, 2nd Edition, O'Reilly Press, ISBN: 0-596-00328-5 (2004)
- [HAR05] Rob Harrop and Jan Machacek, *Pro Spring*, Apress, ISBN: 1590594614 (2005)
- [SIG05] Bill Siggelkow , *Jakarta Struts Cookbook*, O'Reilly Press, ISBN: 059600771X (2005)

Web Sites and Resources

- [FRM] A comparison matrix of tens of frameworks, http://www.frameworks-boulder.org/Application_Frameworks.html,
Last accessed 30 July, 2004
- [CST] Castor, an Open Source data binding framework for Java,
<http://www.castor.org>,
Last accessed 21 July, 2005
- [CYV] Cyveillance, July 2002, <http://www.cyveillance.com>,
Last accessed 30 July, 2004
- [GOG] Google Web Site: <http://www.google.com>,
Last accessed July 22, 2005
- [ITU] International Telecommunication Union, Dec 2002,
http://www.itu.int/ITU-D/ict/statistics/at_glance/Internet02.pdf,
Last accessed July 30, 2004
- [J2EE] Sun Developer Network, Java 2 Platform, Enterprise Edition (J2EE),
<http://java.sun.com/j2ee>,
Last accessed 23 July, 2005

- [JDOM]** JDOM: Open source library for Java-optimized XML data manipulations, <http://www.jdom.org>, Last accessed 23 July, **2005**
- [JSP]** Sun Developer Network, JavaServer Pages Technology, <http://java.sun.com/products/jsp>, Last accessed 23 July, **2005**
- [JSTL]** Sun Developer Network, JavaServer Pages Standard Tag Library, <http://java.sun.com/products/jsp/jstl>, Last accessed 23 July, **2005**
- [SHIN]** Free Struts basics course notes by Sang Shin, www.javapassion.com/j2ee/StrutsTiles.pdf, Last accessed 20 July, **2005**
- [SMH]** OpenSymphony SiteMesh Official Homepage: <http://www.opensymphony.com/sitemesh/>, Last accessed July 22, **2005**
- [SPR]** SpringFramework: Layered Java/J2EE Application Framework, <http://www.springframework.org>, Last accessed 23 July, **2005**
- [TILE]** Official Tiles Guide, http://struts.apache.org/userGuide/dev_tiles.html, Last accessed 20 July, **2005**
- [WHIS]** Whatis IT Encyclopedia, <http://www.whatis.com>, Last accessed 30 July, **2004**
- [WHOS]** Whois Source, <http://www.whois.sc>, Last accessed 30 July, **2004**
- [XML]** W3C - Extensible Mark-up Language (XML), <http://www.w3.org/XML>, Last accessed 18 August, **2005**

Papers

- [BRA00]** T. Bray, et al., “*Extensible Markup Language (XML) 1.0*”, W3C Recommendation, 6 October **2000**
- [BRO01]** Paul Browning and Mike Lowndes, *JISC TechWatch Report: Content Management Systems*, **2001**
- [WID99]** Jennifer Widom, *Data Management for XML: Research Directions*, July **1999**

APPENDIX A

A SAMPLE WEB APPLICATION

We have developed a simple web application in order to show some concrete outcomes to the readers and demonstrate what these frameworks can achieve.

The web site would be a personal homepage and the content was planned to be only static, but also dynamic and multi-lingual. For defining the aim of the web site, we referenced the current homepage of Prof. Dr. M. Akif Eyler (<http://www.eng.marmara.edu.tr/~eyler>). Also, we decided to get the navigation plan and the static content from our reference site.

First we separated the development steps into layers and delivered each layer to the corresponding role for development. The system would have 5 layers: Presentation, web, business, data wrappers and database layers. All these layers should have been developed in coordination but individually.

Business Analyst role made requirements analysis and generated all use cases. That helped a lot while integrating web pages one by one. Business Analysts had used JEWEL System and configure all pages of the web site to the configuration files. When their job had finished, the only thing about web pages was developing the pagelets.

Developers first installed JConfiX and JClassifier to their project. They installed a MySQL database server to their servers. Three already defined tables for JClassifier and three user defined tables for content serving were created to the database server. In order to access to contents from JClassifier, necessary classes of QUARRY was added to the project. After that, installation of the backbone of content management process was completed.

Second, developers should have worked with designers and business analysts in order to define the layout templates. Themes, templates for pagelets, display pages and navigation and pagelets of the web pages were created by developers. As mentioned above, which files the developers should develop had already defined by business analysts in the configuration files of the JEWEL System.

Below, you will see the screenshots of the reference web application. The application was working on the environment stated as follows:

Table A-1. Environmental information about the reference application

Application Server:	Jakarta Tomcat 4.1.x
Database Server:	MySQL 4.1.x
Operating System:	Linux and Windows 98+
Browser:	Mozilla Firefox 1+ and Internet Explorer 5+
Java Environment:	J2SDK 1.4

In order to let you estimate how big is eXaMpLe Project in terms of lines of codes, I prepared a table having this information as follows.

Table A-2. Source code measurements as lines of codes

File extension	Lines of Code	
	eXaMpLe Project	Project Specific
java	11077	748
jsp	998	1520
xml (Configuration)	489	2718
properties (Configuration)	98	2251
js (JavaScript)		699
css (Style Sheet)		601
TOTAL	12662	8507



Figure A-6. Homepage of the reference web application

Above, you can see the main page with 9 pagelets. Pagelets are highlighted by black rectangles. Pagelet 1 and 9 are the header and the footer parts of the page respectively. Pagelet 7 is for username and password validation and pagelet 8 is reserved for banners and ads. The text content of the page is displayed on pagelet 5 and 6. Especially pagelet 6 is for established for a news section.

On the fourth pagelet, the main navigation module is showing links for the guest role. For example, when an editor logs in, the same pagelet will show links for editor role. This is configured by JEWEL System. The structure of the page layout is also governed by JEWEL System. Static and dynamic content are managed by QUARRY.

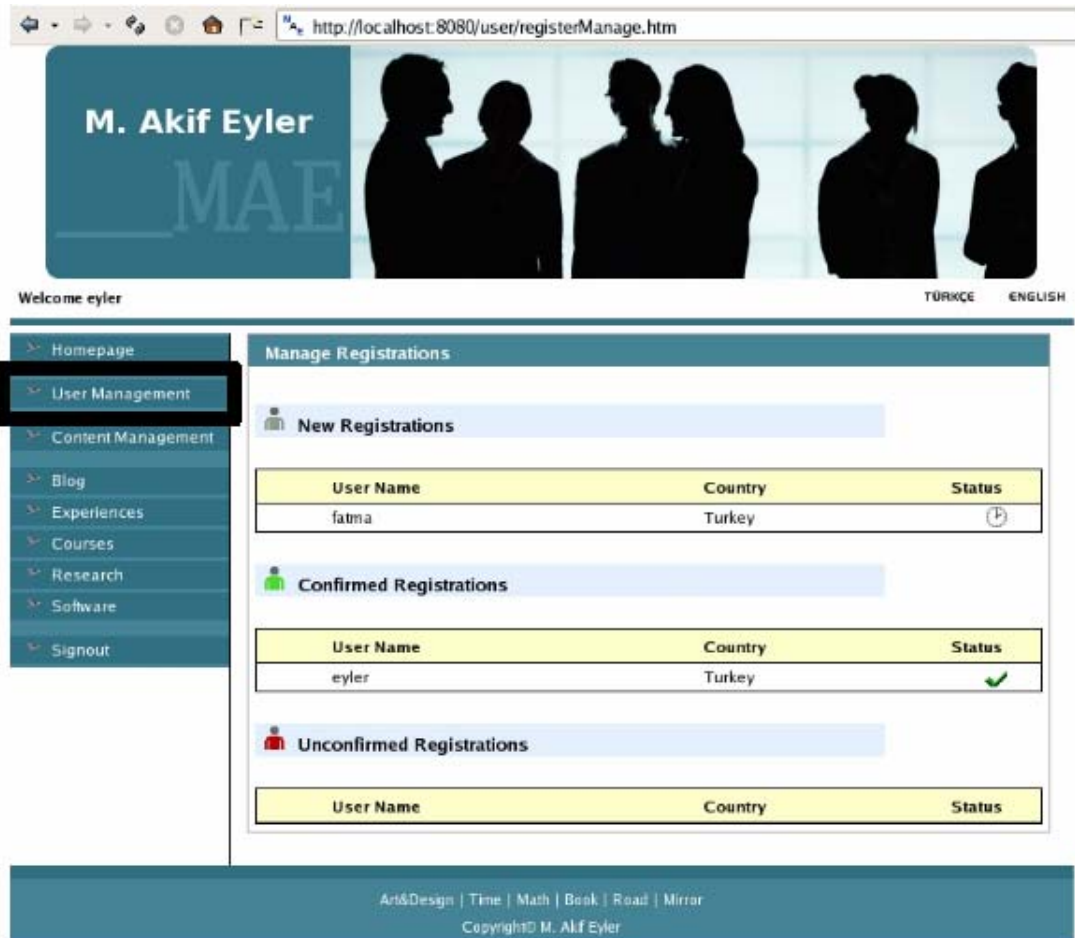


Figure A-7. User management console of the reference web application

According to the business analysts' analysis, the administration role is the one who can manage user registrations. However, content management process can be handled by not only the administration role, but also the editor role.

Above, you can see that the administrator of the web site opened the user management module. The administrators can handle any user registration from the console. This module is built on top of JEWEL System Embedded User Management Module.

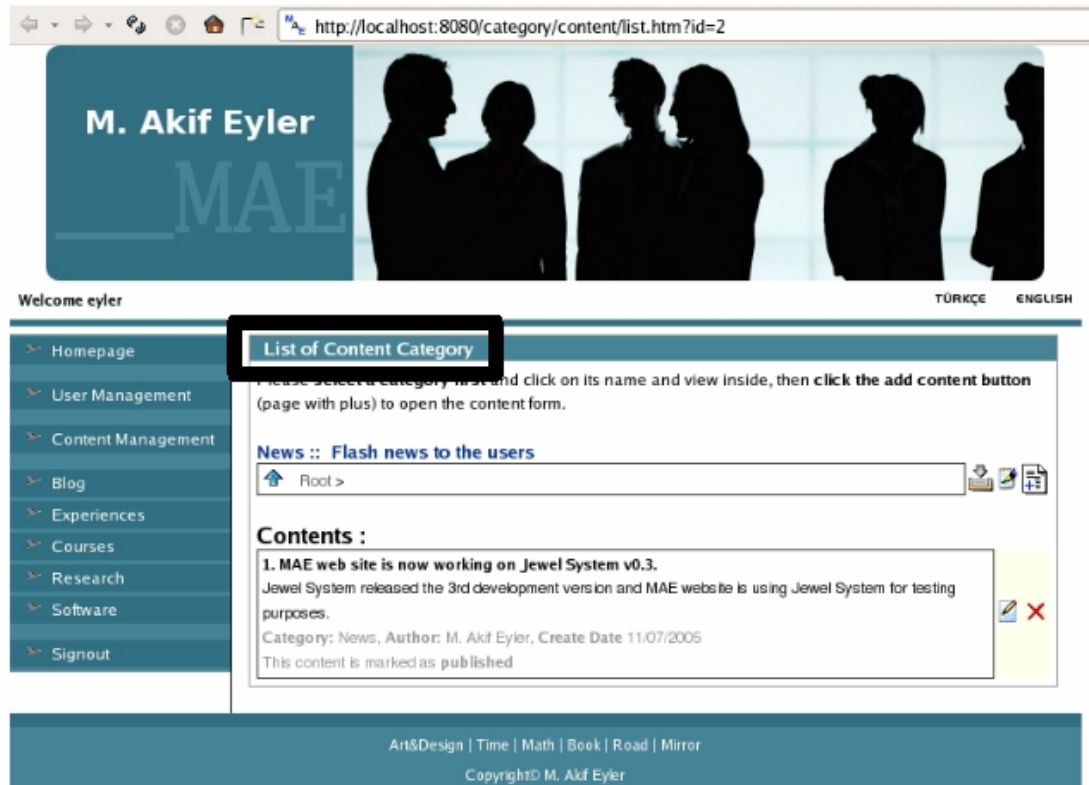


Figure A-8. Content category console of the reference web application

Above, you can see the pages that only JCategoryizer can handle: Categorizing the content. JCategoryizer supports additional content categories. Unlimited number of categories can be generated; it's up to the editors. Additionally QUARRY manages the content categorized and organized by JCategoryizer.

http://localhost:8080/category/content/item/edit.htm?id=23

M. Akif Eyler
MAE

Welcome eyler TÜRKÇE ENGLISH

- Homepage
- User Management
- Content Management
- Blog
- Experiences
- Courses
- Research
- Software
- Signout

Edit Content

Category

News

Author

M. Akif Eyler

Create Date

11/07/2005

Expire Date

31/08/2005

Publish Date

11/07/2005

Retire Date

31/08/2006

Heading

MAE web site is now working on Jewel System v0.3.

Abstract

Jewel System released the 3rd development version and MAE website is using Jewel System for testing purposes.

Data

Jewel System is being developed by MS student Leni Orhan Ergin. In order to understand what it really is, we should look at the evolution of enterprize web appliations.

Figure A.9. Content update page of the reference web application

This “edit content” page is governed by QUARRY and the layout is generated by JEWEL System. You can determine the publish date, the expire date and the retire date of the content. QUARRY is also responsible for managing the status of content by time.

APPENDIX B

PSEUDO CODED WORKFLOW

In order to understand how JEWEL system is handling requests and configure and generate web pages, it would be good to trace down the workflow of http requests. You can access the pseudo codes of the whole JEWEL system.

B.1. Coming Parameters

RoleAbbr, action and layout parameters are global constants for each request, so they should be accessed from everywhere by JEWEL System.

```
ROLEABBR = HttpRequest.roleAbbr  
ACTION = HttpRequest.action  
LAYOUT = HttpRequest.layout (optional, nullable)
```

Figure B-1. Http Request variables coming to JEWEL System

B.2. Rights Checking

Normally, each action has a (one) right. It should be checked that a right exists for the coming action. Rights are searched by the name of the given action. But before checking the right, the existence of the given action should be proved.

```
Check if ACTION exists&isActive in actions.xml ?  
True : VIEW = ActionsXML.actions.action[@name=ACTION].view  
False: ACTION = ActionsXML.actions.action[@default=true]
```

Figure B-2. Determining the view of an action

If the action is not found in actions.xml, action is set to its default value. Default action is determined by selecting an action as default in actions.xml by assigning the *default* attribute to *true*.

```
<action name="loginAction" active="true" default="true">
  <view>loginView</view>
  <description><string type="msg">desc.loginAction</string></description>
</action>
```

Figure B-3. Defining an action

The checking for existence process is the same for roles.

```
Check if ROLE exists&isActive in roles.xml ?
  True : Check if LAYOUT exists ?
    True : LAYOUT remains the same
    False: LAYOUT = RolesXML.roles.role[abbr=@roleABBR].
layout
  Check if LAYOUT exists&isActive in layouts.xml ?
    True : Continue
    False: ERROR! (Layout not found in layouts.xml)
  False: ROLE = RolesXML.roles.role[@default=true]
```

Figure B-4. Determining the default layout for a role

If the role is not found in roles.xml, role is set to its default value. Default role is determined by selecting a role as default in roles.xml by assigning the default attribute to true.

```
<role abbr="g" id="-1" default="true">
  <name><string type="msg">role.guest</string></name>
  <layout>EightJune</layout>
</role>
```

Figure B-5. Defining a role

Actions have no rights by default. If you need to give permission to an action for a set of roles, a right corresponding to that action should be entered to rights.xml.

If the given action exists but the given role has no permission to precede the action, action parameter is converted to PERMISSION-DENIED action.

PERMISSION-DENIED action is accessible by all users and it is defined by default in actions.xml. Users should not change the right for that action. The action attribute is set as system and it is active by default (i.e. active=true) and not visible to ordinary users.

```
<action name="PERMISSION-DENIED-ACTION" active="system">
  <view>permissionDeniedView</view>
</action>
```

Figure B-6. Permission Denied action and its corresponding view

There exist several possibilities implying that the given action has no permission. These are listed.

1. Action has right but not to the given role
2. Action has right to the given role but is inactive
3. Action has no right

```
<right action="loginAction" active="true">
  <abbr>g</abbr>
</right>
```

Figure B-7. Guest (Abbr: g) is determined to access and use login action

All parameters are used by rights.xml first. Each right is indexed by its action attribute. Role abbreviations are written as comma-separated. System actions have rights by all roles by default and have no entry in rights.xml. If you want to re-define the permissions of the system actions and assign to specific roles, you should put an entry about that right to rights.xml. If any system right is overwritten, the active attribute should be specified as system in order to hide to ordinary users.

```
Check if RIGHT exists&isActive in rights.xml ?
True : Check if ROLEABBR exists in RIGHT.abbr
      True : hasRight and ACTION remains the same
      False: ACTION = permission-denied-action
False: ACTION = permission-denied-action
```

Figure B-8. Checking if a role has right to execute an action

B.3. Defining the View

View is determined when we access to the given action in actions.xml. In order to get the grid information stored in that view, layout should be known. Layout is determined by the coming request or by the given role. Information of the current grid is extract from the given layout of the given view.

```

<view name="rootView">
  <definition layout-name="sampleLayout">
    <pagelet-ref location="header" value="logo"/>
    <pagelet-ref location="header" value="headerInfo"/>
    <pagelet-ref location="footer" value="simpleFooter"/>
  </definition>
</view>
<view name="loginView" extend="rootView" >
  <property name="title" type="msg">
    <value>view.user.login.title</value>
  </property>
  <definition layout-name="sampleLayout">
    <pagelet-ref location="middle" value="welcomeImage"/>
    <pagelet-ref location="middle" value="welcomeText"/>
    <pagelet-ref location="middle" value="news"/>
    <pagelet-ref location="right" value="login"/>
    <pagelet-ref location="right" value="rightBanner"/>
  </definition>
</view>

```

Figure B-9. View definition configuration

As mentioned above, PERMISSION-DENIED-ACTION calls permissionDeniedView, so an entry for that view should be available. It's the developer's responsibility.

Title value is put to <title> tags in html page. Placements have 3 references. One to a cell of layout in layouts.xml, one to a pagelet in pagelets.xml and one to a pagelet template in templates.xml.

```

Check if VIEW exists&isActive in views.xml ?
  True: Check if definition exists in VIEW.definition
    True: get all pagelet-refs
      foreach pagelet-ref
        PEGELET-REF-PROCESSING
      endForEach
    False: ERROR! (Illegal layout is specified in view-definitions.xml)
  False: ERROR! (View not found in view-definitons.xml)

```

Figure B-10. Integrating the pagelets defined in a view

Views are searched by the name attributes. If a view is not found for a requested action, it throws an error. It's an undesired condition actually. If a view is found for an action but a grid is not detected, it throws an error again.

```

PAGELET-REF-PROCESSING

Check if LAYOUT.cell[pagelet-ref.location] exists in layouts.xml ?
True:
  Generate layout (LAYOUT-PROCESSING)
  Check if pagelet-ref.value exists&isActive in pagelets.xml ?
  True :
    Generate pagelet (PAGELET-PROCESSING)
    Check if pagelet-ref.template exists&isActive in templates.xml?
    True:
      Generate template (TEMPLATE-PROCESSING)
      Import pagelet into template
    False: ERROR! (Template not found in layouts.xml)
    EndCheck
  False: Contine (nothing imported)
  EndCheck
  Import template into cell
False: ERROR! (Illegal cell is specified in layouts.xml)
EndCheck

```

Figure B-11. Processing of pagelets and templates

As you mentioned above, include feature of jsp works on generating process. Layout includes templates and templates include pagelets (or navigations, that is navigations are specialized pagelets).

APPENDIX C

TECHNOLOGIES / PLATFORMS USED

There are several technologies are used for developing the eXaMpLe Project. We would like to give brief information about them, so that the reader can imagine the development cycle.

C.1. Application Framework

In this section, we will explain what an application framework is. *“An application framework is a term usually used to refer to a set of libraries or classes that are used to implement the standard structure of an application for a specific operating system.”* [FRM]

By bundling a large amount of reusable code into a framework, much time is saved for the developer, since she is saved the task of rewriting large amounts of standard code for each new application that is developed.

Application frameworks became popular with the rise of the GUI since these tended to promote a standard structure for applications. It is also much simpler to create automatic GUI creation tools when a standard framework is used, since the underlying code structure of the application is known in advance.

Object Oriented programming techniques are usually used to implement frameworks such that the unique parts of an application can simply inherit from pre-existing classes in the framework.

C.2. Ant

Ant is an amazingly useful tool for building, testing, deploying, and managing Java-based projects. It replaces make as the tool of choice for building projects, as well as providing an excellent cross-platform utility for basic programming-task-related scripting.

C.3. Castor

Castor is a Java framework that provides data binding services for XML and SQL, and supports a JDO interface [CST]. Castor is a popular and powerful way to load, modify, and save XML documents based on XML Schema Definitions. Given a sample XML document, it can also generate an XML Schema Definition and Java source able to read and write any XML document based on the schema.

C.4. Display Tag Library

Display Tag is a JSP Tag Library that can be used to present lists of data in a table format. Display Tag handles column display, sorting, paging, cropping, grouping, exporting, smart linking and decoration of the table in a nice and customizable XHTML style.

C.5. J2EE - Java 2 Platform, Enterprise Edition

“The J2EE platform provides a framework for developing and deploying web services on the Java platform. Faster solutions delivery time to market. The J2EE platform uses “containers” to simplify development. Simply it separates logics. Freedom of choice. J2EE technology is a set of standards that many vendors can implement.” [J2EE]

The Java 2 Software Development Kit is at the core of all things Java®. While every Java developer relies on the JDK for compiling and debugging programs written in Java, one of the most important aspects of the JDK is the Java Runtime

Environment (JRE) which lets Java programs execute on a platform. Because most operating systems support a JRE, including Linux®, nearly all other flavors of Unix®, Mac® OS, Windows®, most IBM® platforms, including AS/400® and mainframes, and many more, Java programs can run on almost any operating system.

C.6. JDOM

JDOM is an XML manipulation technology for Java that makes XML parsing, printing, and transformation fast and easy.

“While JDOM interoperates well with existing standards such as the Simple API for XML (SAX) and the Document Object Model (DOM), it is not an abstraction layer or enhancement to those APIs. Rather, it seeks to provide a robust, light-weight means of reading and writing XML data without the complex and memory-consumptive options that current API offerings provide.” [JDOM]

C.7. JSP - Java Server Pages

“Use JSP technology without having to learn the Java language. Although scriptlets are no longer required to generate dynamic content, they are still supported to provide backward compatibility. Extend the JSP language. Java tag library developers and designers can extend the JSP language with "simple tag handlers". Easily write and maintain pages. JSTL expression language is now integrated into JSP technology.” [JSP]

C.8. JSTL

“The Java Server Pages Standard Tag Library (JSTL) encapsulates as simple tags the core functionality common to many Web applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.” [JSTL]

Using this tag library is a smart choice for developers who want to make sure their code is portable across Tomcat, WebLogic, WebSphere, etc. with minimal work.

C.9. Spring Framework: Layered Java/J2EE Application Framework

Spring is a layered Java/J2EE application framework based on code published in 'Expert One-on-One J2EE Design and Development' by Rod Johnson (Wrox, 2002). Spring is a very flexible and somewhat less complex alternative to J2EE. It works with a variety of web frameworks and persistence mechanisms, and can be used either in conjunction with an application server or standalone. This application server independence makes testing much easier. [SPR]

It is an IoC Container, or we can call it by its new-brand name: “*Dependency injection*”. [SPR, HAR05]

It serves us a powerful JavaBeans-based configuration management, applying Inversion-of-Control principles and a generic abstraction layer for transaction management.

It's a flexible MVC web application framework. This framework is highly configurable via strategy interfaces, and accommodates multiple view technologies like JSP, Velocity, Tiles, iText, and POI. Note that a Spring middle tier can easily be combined with a web tier based on any other web MVC framework, like Struts, WebWork, or Tapestry. It lets us integrate our projects seamlessly with Hibernate, JDO, and iBATIS SQL Maps. [SPR]

C.10. XML – Extensible Mark-up Language

It's a flexible way to create common information formats and share both the format and the data on the World Wide Web, intranets, and elsewhere. XML is "extensible" because, unlike HTML, the mark-up symbols are unlimited and self-defining. [XML]

SGML, the standard for how to create a document structure, is the parent of HTML and XML. It is expected that HTML and XML will be used together in many Web applications.

APPENDIX D

FILE HIERARCHY IN JEWEL SYSTEM

JEWEL uses several files like XML configuration files which stores all information that layout generators (a.k.a. managers) need. Since these generators need to bind this information to Java Data Objects (JDO), Castor Data Binding Framework is used. Therefore there should be a mapping file for each XML configuration file. The beans instantiated by Castor are concrete and usable object-view of the virtual information.

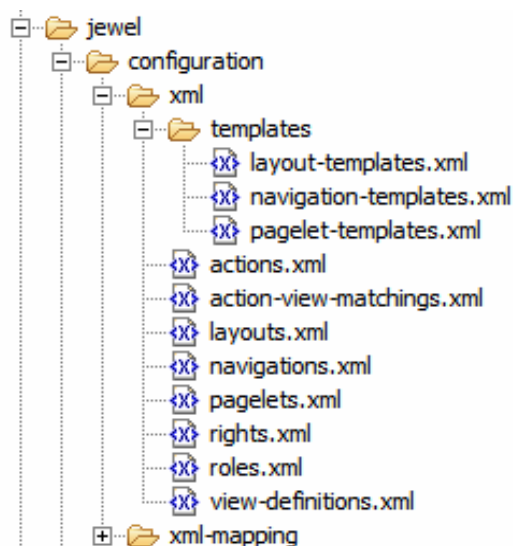


Figure D.1. Configuration files of JEWEL System

There exist 11 XML configuration files in two directories.

Here you can see the configuration files of JEWEL in XML format. Templates for Web User Interface are listed in templates folder. Other XML files correspond to role, right, view and layout managements.

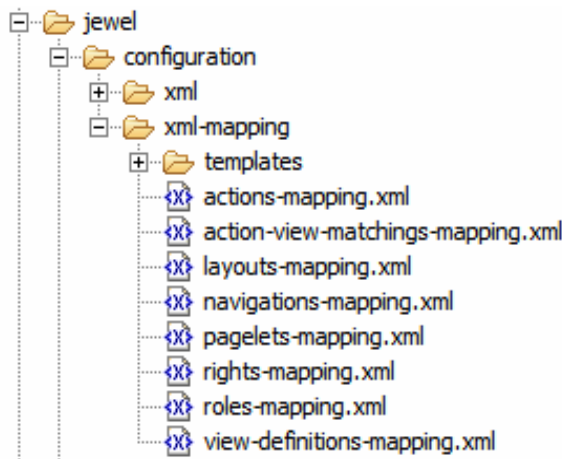


Figure D.2. Mapping files of the configuration files of JEWEL System

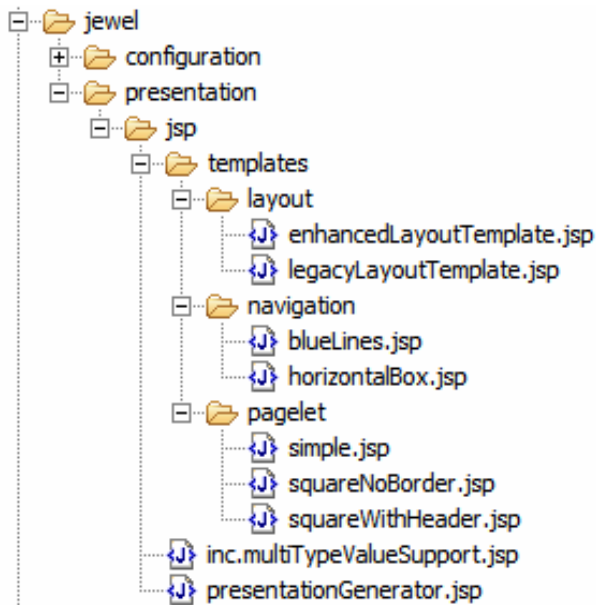


Figure D.3. Templates in JEWEL System

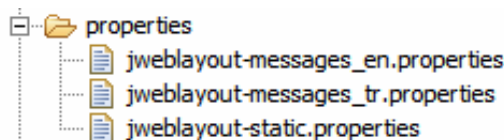


Figure D.4. Properties files of JEWEL System

Xml-mapping folder holds mapping specifications stored as XML. Each XML configuration file matches with its corresponded mapping-XML file. Castor Data Binding Framework uses these mapping files and binds XML to JDO.

On the left, we can see all templates and required jsp pages for web page layout generation process. PresentationGenerator.jsp is the main file that generates all pages and web page layouts. MultiTypeValueSupport.jsp is included to every template for using XXX.properties files by these XMLs.

Like other sub-projects, JEWEL System uses several xxx.properties files. By using these, multi-linguality is supported.

D.1. Files for Rights Management

actions.xml: Definable actions in the portal. Each action has zero or one view but a view has one or more actions. There is no obligation that actions are defined for layouts. Actions may be used by other modules of the portal for managing rights.

rights.xml: Determines who has a right to implement which action. Each action should have a right.

roles.xml: Roles in the system is stored here. Roles are used when determining rights of actions in rights.xml and rights of links in navigations.xml.

D.2. Files for Layout Management

view-definitions.xml: Layout of each page is formatted here. There exist references to pagelets to form a page. Each view means a page layout type. Views may extend from other views.

pagelets.xml: All pagelets are defined. Actual URLs of each pagelet should be stated in its corresponding property.

navigations.xml: Defines all links, link sequences for each role. Rights for each link sequence is determined here.

layouts.xml: Main table types that holds all pagelets in its cells are defined here. Do not forget that just by changing the values of "colspan" and "rowspan values" of "td" tag, you are able to generate all types of tables.

D.3. Files for Layout Template Management

layout-templates.xml: All layout-templates are entered and registered to JEWEL.

navigation-templates.xml: All navigation-templates are entered and registered to JEWEL.

pagelet-templates.xml: All pagelet-templates are entered and registered to JEWEL.

D.4. Files for Right-Layout Interaction:

action-view-matchings.xml: Each action has zero or one view but a view has one or more actions. These matchings are stored here.

RESUME



Lemi Orhan Ergin is a java developer, who is working at Sony Europe's Istanbul Development Centre as of July 2005. Mainly he has specialized in development of web applications on J2EE platform and in XML based technologies. He likes working on new technologies and creating new applications and tools from scratch. He is experienced in product information management systems, e-commerce and online auctioning systems. He is also a supporter of open-source technologies and Linux That's why; he is a member of Linux Users Group of Turkey since 2004.

He graduated from Marmara University Faculty of Engineering as a computer engineer in 2002. In graduate years, he had two interest areas: Parallel processing and web development. He has developed a new task scheduling algorithm as his graduate thesis on parallel processing area. Also, he participated in several web application development projects by using several tools, technologies and environments.

He also got a master of science degree in computer science from the same department in Marmara University in 2005. For three years, he has been mainly working on java and XML based projects, especially on J2EE platform.

He is hoping to add some innovative ideas to J2EE and web development community one day as a motivated developer.