H. ERGÜN

M.S Thesis

2008

SOA BASED FORM DEVELOPMENT FRAMEWORK WITH WEB 2.0

HÜSEYİN ERGÜN

IŞIK UNIVERSITY
2008

SOA BASED FORM DEVELOPMENT FRAMEWORK WITH WEB 2.0

HÜSEYİN ERGÜN

M.S., Computer Engineering, Işık University, 2008

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

IŞIK UNIVERSITY

2008

IŞIK UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

SOA BASED FORM DEVELOPMENT FRAMEWORK WITH WEB 2.0

HÜSEYİN ERGÜN

APPROVED BY:

(Prof. Selahattin Kuru) (Bahçeşehir University) _____
(Thesis Supervisor)

(Assist. Prof. Taner Eskil) (Işık University) _____

(Assoc. Prof. Seyhun Altunbay) (Işık University) _____

APPROVAL DATE: …./…./….

SOA BASED FORM DEVELOPMENT FRAMEWORK WITH WEB 2.0

Abstract

Enterprise companies require central, web based, performance proved robust applications. There are different types of frameworks and design patterns to fulfill this requirement. The author of this thesis and a team of engineers have developed a platform for software houses and it departments, easy and standards based of Ajax web applications with a WYSIWYG graphical user interface design studio, XML structure of defining a web page and a complete Ajax rendering mechanism to parse this XML system. This way, companies are able to build a web application like a desktop platform even without writing any code in the front part.

SERVİS TABANLI WEB 2.0 İLE FORM GELİŞTİRME PLATFORMU

Özet

Bilgi işlem departmanları ve yazılım firmaları, merkezi, web tabanlı, performanslı çalıştığı garanti edilen, güçlü uygulamalara ihtiyaç duymaktadır. Bu ihtiyacı karşılamak için çeşitli uygulama geliştirme alt yapıları ve tasarım kalıpları bulunuyor. Bu tezin yazarı ve araştırma ekibindeki mühendisler standart temelli, Ajax ile uygulama geliştirmeye olanak sağlayacak ve kolay kullanılan bir tasarım stüdyosuna sahip, XML tabanı sayesinde birden çok platformda çalışan yapıya sahip bir platform geliştirdiler. Bu sayede firmalar, tıpkı masaüstü uygulaması geliştirir gibi web uygulaması yazabilir ve standart temelli bir çok sistemden otomatik tasarım aktarabilir.

Acknowledgement

I would like to express my gratitude to my thesis advisor Professor Selahattin Kuru, for this kind attitude in consulting, suggestions, giving support and patience during project development of project and writing the thesis.

Table of Contents

Abstract ii

Özet iii

Acknowledgement iv

Table of Contents v

List of Tables vii

List of Figures viii

Abbreviations x

1. Introduction 1

2. State of the Art 3

2.1 SOA (Service Oriented Architecture)…….....…….3

2.1.1 Definition……...…….....…….....3

2.1.2 SOA Service Characteristics.....……..4

2.1.3 SOA Components…….……..6

2.2 Web 2.0…….……...6

2.2.1 Ajax…….……...…….……7

2.2.2 Javascript…….…….…..8

2.2.3 XML…….…….....8

2.3 Architectural Components……...……...9

2.3.1 EJB…….……9

2.3.2 Hibernate…….……...9

2.3.3 Eclipse…….…….....10

3. Motivation for the Proposed Work 11

3.1 Similar Applications.....12

3.2 Oracle Forms.....12

3.3 Software AG, Natural for Ajax.....13

3.4 Alternative Frameworks for Web Based Java Software Development..13
4. SOA Based Web 2.0 Development System 14
4.1 XML Structure to Complete an End to End Application…….....14
4.2 SOA Architecture and Server Side Mechanism.....16
4.3 Beans to Provide an Application.....22
4.4 Configuration Files and Their Definitions for This System to Work.....22
4.5 Rendering Mechanism for XML Ajax.....29
4.5.1 About Qooxdoo.....……...…….……..29
4.6 Service Execution Mechanism.....30
4.7 Jgroups.....31
4.8 Cache Mechanism.....32
5. Sample Application and Design Studio 36
5.1 Eclipse Plugin.....36
5.2 Sample Application Tutorial.....43
6. Conclusions and Recommendations for Future Work 50
References 52
Appendix A. List of Configuration Files 55
Appendix B. CD Containing Code Listing and Installation Package 60
Curriculum Vitae 61

List of Tables

Table 4.1 The Definitions of XML Structure 16
Table 4.2 List of Beans and Their Descriptions on the System 24
Table 4.3 XML Configuration Files 27

List of Figures

Figure 2.1 Service Oriented Architecture Components 7
Figure 4.1 Client Server Communication Mechanism 14
Figure 4.2 GUIML Page Structure Building Framework Applications 15
Figure 4.3 Server Side SOA Architecture 20
Figure 4.4 Server Configuration with Different Context Bean 21
Figure 4.5 Sample Context Bean Code 23
Figure 4.6 Sample Service Code 23
Figure 4.7 Sample Configuration File 27
Figure 4.8 Ajax Rendering Mechanism 29
Figure 4.9 Sample Bean Code for Combo Box 30
Figure 4.10 Execute Service Method on Ajax 31
Figure 4.11 Cache Mechanism Sample Code 32
Figure 4.12 Jgroups Architecture 33
Figure 4.13 Cache Data structured as a Tree 35
Figure 5.1 Eclipse Configuration Files 37
Figure 5.2 Sample Application Screen 38
Figure 5.3 Actions Tab to Show Client Side Method Development 39
Figure 5.4 Services Tab to Show How a Service is Implemented 40
Figure 5.5 Variables Tab to Show the List of the Variables 41
Figure 5.6 Events Tab to Show the List of Events on a Page 42
Figure 5.7 Creating a New Project 43
Figure 5.8 Creating a New Package 44
Figure 5.9 A New Page, After It is Created 45
Figure 5.10 Fill Combo Service Code 46
Figure 5.11 Combo Box Set Model Data 47

Figure 5.12 Button and Click Event 47
Figure 5.13 Binding the Event to the Action 48
Figure 5.14 Running the Application for Test 49

Abbreviations

API	Application Programming Interface
ACID	Atomicity, Consistency, Isolation, Durability
AJAX	Asynchronous Java Script and XML
BPM	Business Process Management
EJB	Enterprise Java Bean
GUI	Graphical User Interface
GUIML	Graymound User Interface Markup Language
JDBC	Java Database Connectivity
JEE	Java Enterprise Edition
JSON	Java Script Object Notation
JTA	Java Transaction API
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
UML	Universal Markup Language
URL	Uniform Resource Locator
WSDL	Web Service Description Language
WYSIWYG	What You See is What You Get
XML	Extensible Markup Language

Chapter 1
Introduction

Enterprise companies require central, web based, performance proved robust applications. There are different types of frameworks and design patterns to fulfill this requirement. The author of this thesis and a team of engineers have developed a platform for software houses and it departments, easy and standards based of Ajax web applications with a WYSIWYG graphical user interface design studio, XML structure of defining a web page and a complete Ajax rendering mechanism to parse this XML system. This way, companies are able to build a web application like a desktop platform even without writing any code in the front part.

This way it will be easier for software developers to form a web application, maintain the code and make changes if necessary.

Technically the system is n-tier application framework for JEE developers to develop JEE applications rapidly with a design studio. In this way JEE developers could develop and distribute Rich Interface Applications. This entire framework is based on open source JEE solutions and they have the following features:

- Database tier built on JDBC and Hibernate [1]
- Presentation tier built on Qooxdoo [2], Swing and JasperReports [3]
- Client reaches application via Web Browser or Java Web Start
- Presentation tier renders User Interface (GUIML) pages and generates JavaScript and Swing output.
- Server-Client communication built on JSON, SOAP, Java contexts
- User Interfaces developing on Eclipse based Graymound Editor Plug-in

Upon the completion of this thesis project, companies in the IT industry have started using this framework and began to implement their applications on top of this framework. Furthermore, the author continues to implement new features. On the following years, the service layer between client and server will grow richer with BPM Engine.

The Framework has the features of XML rendering. This way it is able to generate both a desktop application and an Ajax front end. Another feature is SOA based client server communication. This methodology generates a loosely coupled relation between server components and modules. Also it has a server based transaction management, report integration and web service generation components which easies the development of enterprise applications and makes the developers target business logic instead of technology details on back end side. When the system is deployed on server side, it has a distributed cache mechanism, for increasing the performance and increases the availability.

The motivation of the proposed framework arises with providing a standard based application development platform with easy of customization and development. Java applications have many server side components to learn and configure like EJB, servlet, client server communication, session, security management, database connection pooling, distributed cache etc. There are application development environments on the market. This framework is the new generation of oracle forms [4] or any other alternative form generation frameworks with N tier SOA based structure and 100% Ajax compatibility.

The thesis contains six chapters and two appendices. The first chapter is the introduction. The second chapter gives an overview of the state of the art. The third chapter implies the motivation of the proposed work. The fourth chapter explains framework and its architecture details. Fifth chapter is the evaluation of the project and the last chapter is the conclusion and the recommendations for the future work.

Chapter 2
State of the Art

In this chapter state of the art of SOA is investigated. We first discuss SOA itself, which is followed by a discussion of XML, and then we give the state of the art in Web 2.0 and architectural components. The system is based on Service Oriented architecture which became popular over the last years. It combines the design pattern or services via Ajax development front end results a web based, fast robust and enterprise application development platform. The system itself uses many external tools and functions as an end to end framework.

2.1 SOA (Service Oriented Architecture)

Service-Oriented Architecture (SOA) is a software architecture where functionality is grouped around business processes and packaged as interoperable services. SOA also describes IT infrastructure which allows different applications to exchange data with one another as they participate in business processes. The aim is a loose coupling of services with operating systems, programming languages and other technologies which underlie applications [5].

2.1.1 Definition

"SOA is a form of technology architecture that adheres to the principles of service orientation, when realized through the Web service technology platform; SOA establishes the potential to support and promote these principles throughout the business process and automation domains of an enterprise [6]."

Service-Oriented Architecture (abbreviated as SOA) is an architectural pattern that defines the use services as defined in the statement and accesses the parts of the system as a generic service mechanism. Services are software components that allow remote access over standard protocols and provide declarative descriptions of their requirements and capabilities [7]. These services have well-defined and mostly XML based interfaces that are platform, and protocol independent which forms a loosely coupled relationship. With the help of this structure it is easy for an external system to bind its application or communicate with these services. These heterogeneous services can interact with each other in a uniform and universal manner and forms a interoperability. This interoperability benefits companies that adopt a SOA strategy, which highly increases of the reusability of the existing software system. The need of a business to change can be caused by partnerships, mergers, acquisitions, changed business focus, changing policies, etc. Currently big vendors like Oracle, IBM, Software AG. And HP has solutions on this SOA migration and has a SOA development strategy and platforms.

2.1.2 SOA Service Characteristics

All the services in SOA architecture have similar characteristics. These characters can be stated as follows:

- Services are stateless in nature. A SOA service should operate independently of other services, without pre-conditions and side effects. In most systems, to form the authentication session data is passed between the service calls. In addition, since services are stateless, they are transactional.
- Services are technology independent. Developers might implement the service with the programming language and platform of their choice. However calling a web service and connecting is platform and language independent.
- Services are discoverable and dynamically bound [8]. A service consumer that needs a service discovers what service to use based on a set of criteria at runtime.

The service consumer does not know the format of the request message or response message or the location of the service until the service is actually needed.

- Services are self-contained and modular. Since all the services are stand alone and do not depend to each other, with making a package of services it is possible to make a modular development and modular system.
- Services are loosely coupled. *Coupling* refers to the number of dependencies between modules. There are two types of coupling: loose and tight. Loosely coupled modules have a few well known dependencies. Tightly coupled modules have many unknown dependencies. All software architecture strives to achieve loose coupling between modules. Service-oriented architecture promotes loose coupling between service consumers and service providers and the idea of a few well-known dependencies between consumers and providers [9].
- Services have a network-addressable interface. A consumer on a network must be able to invoke a service across the network. The network allows services to be reused by any consumer at any time. The ability for an application to assemble a set of reusable services on different machines is possible only if the services support a network interface.
- Services have coarse-grained interfaces. The appropriate level of granularity for a service and its methods is relatively coarse. A service generally supports a single distinct business concept or process. It contains software that implements the business concept so that it can be reused in multiple large, distributed systems.
- Services are location-transparent. Consumers of a service do not know a service's location until they locate it in the registry. The lookup and dynamic binding to a service at runtime allows the service implementation to move from location to location without the client's knowledge.
- Services are composable. A service's composability is related to its modular structure. Modular structure enables services to be assembled into applications the developer had no notion of when designing the service. Using preexisting, tested services greatly enhances a system's quality and improves its return on investment because of the ease of reuse.

- Service-oriented architecture supports self-healing. For instance, a service implementation may run in a clustered environment. If a single service implementation fails, another instance can complete the transaction for the client without the client's knowledge.

2.1.3 SOA Components

The following are the SOA Components. In Figure 2.1 these components are illustrated.

- Service provider: Service provider registers itself and waits for a consumer to connect to the service. The service provider has the working service and ready to connect. The start of a service is always initiated by the request of a service consumer.
- Service consumer: A service consumer requests a service from a service provider and supplies it with data. In the case the service providers sends a response, the service consumer will process this result.
- Service registry: As with a basic Web Services architecture, a services repository is important to dynamically discover services that can execute a certain task.

2.2 Web 2.0

Although in the last few years, Web 2.0 is a very popular term, it is difficult to give a precise definition. Tim Berners-Lee, the inventor of the Internet explains Web 2.0 as: "*Web 1.0 was all about connecting people and facilitating new kinds of collaboration. It was an interactive space, and I think Web 2.0 is of course a piece of jargon, nobody even knows what it means. If Web 2.0 for you is blogs and wikis, then that is people to people. But that was what the Web was supposed to be all along*" [10]. In short, Web 2.0 is the Web where people meet, collaborate and share anything that is popular by using social software applications [10]. The term refers to second generation of Internet-based services: blogs, wikis, communication tools and

platforms like del.icio.us [11], Flickr [12], Wikipedia [13], last.fm [14], Technorati [15].

Web 2.0 applications derive from new techniques such as rich internet applications (RIA), Asynchronous JavaScript and XML (AJAX), semantically valid Extensible Hyper Text Markup Language (XHTML), Cascading Style Sheets (CSS), Syndication and aggregation of data in RSS or Atom, clean and meaningful URLs. A user of Web 2.0 must feel as if he/she used traditional desktop applications to share anything with the community.

2.2.1 Ajax

AJAX is a web development technique to create web applications as if they were desktop ones. The aim is to exchange only small amounts of data with a server; this should be performed behind the scenes. No longer should entire page be (re)loaded. One of the first Web 2.0 applications was Google Maps, a set of interactive maps of the world. One can watch diverse views of the world, change the way the views are displayed and personalize them. There is a constant dialog between the server and client application, but a page is not reloaded.

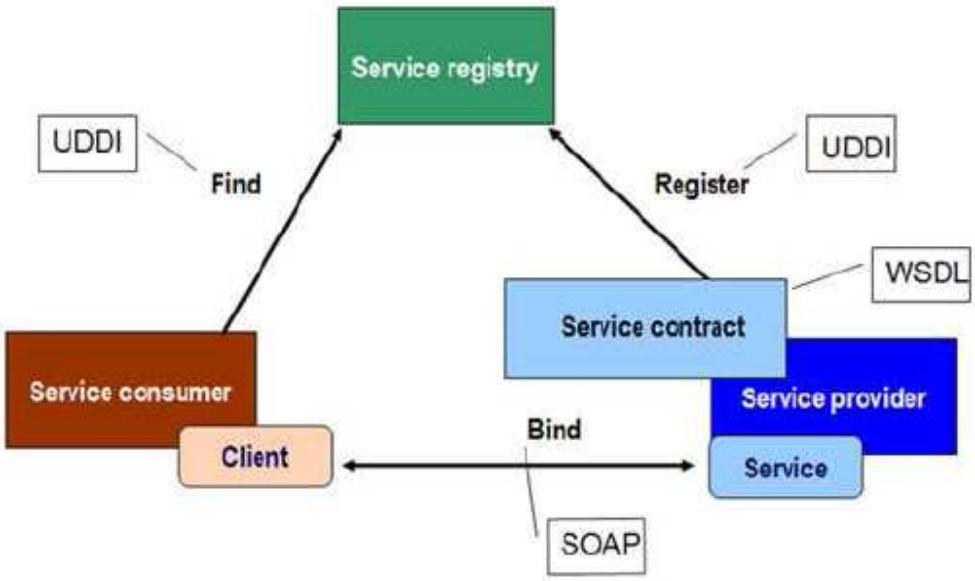

Figure 2.1 Service Oriented Architecture Components

What makes Ajax interesting is that it is based on existing web standards like HTML, CSS, DOM, XML and JavaScript. Modern web browsers are in a sense already Ajax-enabled. Ajax unlike other RIA technologies does not require the user to install various plug-in. Unlike ordinary web applications an Ajax application would be able to change and update parts of the view without reloading the web page entirely. A user could for instance read e-mails in one part of the webpage while another part of the web page is updated automatically with new incoming messages. This update is done by the web application calling a web service asynchronously. When the response has been completed the view is updated without interrupting the user. Any Ajax application would rely heavily on JavaScript. JavaScript has not always been taken seriously by developer but this new dependence requires a new view on JavaScript.

2.2.2 Javascript

The dictionary definition of Javascript is "*a scripting language most often used for client-side web development. It was the originating dialect of the ECMAScript standard. It is a dynamic, weakly typed, prototype-based language with first-class functions*." [16] JavaScript was influenced by many languages and was designed to look like Java, but be easier for non-programmers to work with.

2.2.3 XML

The Extensible Markup Language (XML) is a general-purpose *specification* for creating custom markup languages. It is classified as an extensible language because it allows its users to define their own elements. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet, and it is used both to encode documents and to serialize data.

2.3 Architectural Components

On the server side of the system there are open source components used for a fast and robust development environment. A stateless session bean EJB is used to handle the transactions. Hibernate is used for object relation persistence. It maps the tables with objects inside the code. Eclipse is used as a development environment. And design studio is written as a plugin in Eclipse so that users can define the user interface fast and efficiently.

2.3.1 EJB

The EJB specification is one of several Java APIs in the Java Platform, Enterprise Edition. The EJB specification intends to provide a standard way to implement the back-end 'business' code typically found in enterprise applications (as opposed to 'front-end' user-interface code). Such code was frequently found to reproduce the same types of problems, and it was found that solutions to these problems are often repeatedly re-implemented by programmers. Enterprise JavaBeans were intended to handle such common concerns as persistence, transactional integrity, and security in a standard way, leaving programmers free to concentrate on the particular problem at hand.

The system uses stateless session bean. Stateless Session Beans are distributed objects that do not have state associated with them thus allowing concurrent access to the bean [17]. The contents of instance variables are not guaranteed to be preserved across method calls. The lack of overhead to maintain a conversation with the calling program makes them less resource-intensive than statefull beans.

2.3.2 Hibernate

Hibernate [1] is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional

relational database. Hibernate solves Object-Relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions. On the framework, Hibernate is used as a standard ORM tool. With its eclipse plugin, it is easy to generate mappings and form the application.

2.3.3 Eclipse

Eclipse [19] is a software platform comprising extensible application frameworks, tools and a runtime library for software development and management. It is written primarily in Java to provide software developers and administrators an integrated development environment (IDE). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Language packs provide translations into over a dozen natural languages. The key to the seamless integration (but *not* of seamless interoperability) of tools with Eclipse is the plugin. With the exception of a small run-time kernel, everything in Eclipse is a plugin. This means that every plugin developed integrates with Eclipse in exactly the same way as other plugins; in this respect, all features are created equal. Eclipse provides plugins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plugins include UML plugin for Sequence and other UML diagrams, plugin for Database explorer, etc. The application design studio is written as an Eclipse plugin, thus enjoys all features of Eclipse development environment.

Chapter 3
Motivation for the Proposed Work

Enterprises are searching versatile, robust and high performance transaction processing platforms that can be accessed on the Web and administered by centralized IT departments. The vision of Web-enabling enterprise applications on any platform with user friendly and real-time interactivity is a compelling one. Such a pretentious target has been almost proven difficult to attain with given restrictions of the Web's existing "click-and-use" model.

Enterprises are really in search of today is the unique harmonization of the following best of breeding approaches: rich functionality of desktops, robust and continuous service infrastructure of client/server together with ubiquitous accessibility and low cost of ownership of the Web. Java Enterprise Editions are robust for many years to fulfill the requirements of enterprises such as banks, telecom companies and government departments. However when combining with a web solution on top of SOA based system, with a brick and click solution there appears a huge gap. With the improvement of Ajax technologies and frameworks on top of this technology, it is now possible to develop "desktop looking" applications easier. However there lacks a complete solution providing an end to end solution with a robust and good looking design studio, SOA based architecture and complete N tier enterprise application requirements.

In this chapter we discuss the motivation of the framework. We first discuss the similar applications and their strengths and weaknesses comparing with the framework.

3.1 Similar Applications

The system has two important components. First one is rendering an application from XML and second one is complete SOA based architecture. For the SOA system, enterprises have solutions for Enterprise Service Bus (ESB) for connecting and orchestrating the web services. There are also many design platforms for service development. Starting from open source like, Eclipse Web Tools Platform, which is also, developed by a local Turkish company Etheration, IBM rational development studio, and a global propriety application. Oracle has SOA Development Suite to fulfill the service development requirements. However, combining service development with web applications (not automatic generation) and maintaining them fast is not available with these platforms. XML rendering and generating a cross platform application from XML, there is a Cross Platform Development Tool in Software Ag [20]. It has some similar parts like generating xml in the back ground and Ajax front end, however, this structure is currently works more efficiently and flexible comparing to those alternative.

3.2 Oracle Forms

Oracle Forms is a tool with a PL/SQL code [21], which allows a developer to quickly create user-interface applications which access an Oracle database in a very efficient and tightly-coupled way. It was originally developed to run server-side in character mode on any UNIX box, before Windows existed. It was then ported to Windows to function in a client-server environment. Recent versions have been ported to Java. It now runs in a J2EE container and can integrate with Java and web services.

The Advantages of Oracle Forms is fast development and rich client features for application development. Also there are many developers experienced on this technology. The system works fast and efficiently over Oracle Database. It has wizards for screen generation and client server connectivity for session. This increases the data management performance and communication.

On Oracle Forms having direct connection with client server has some disadvantages. Due to that reason it becomes impossible to make http based web application. The web in nature is stateless and you cannot have a direct connection with server side. Another disadvantage is, Pl/SQL is on the server side and database dependent. You cannot use any other database instead of Oracle. Also on Pl/SQL, object oriented programming or using software engineering tools like code coverage, unit tests, version control system or automated build mechanism.

3.3 Software AG, Natural for Ajax

Software AG is Germany based software Development Company whom acquired WebMethods, a company targeting middleware. The Natural for Ajax platform [22] of WebMethods has a drag and drop based desktop like development environment which is similar to the proposed framework. The advantages of this system is advanced GUI development tool for Ajax and many embedded design and sample applications. The disadvantage of the system is it does not have a client side control, thus for every action it has to connect with the server side which decreases performance and requires more server investment.

3.4 Alternative Frameworks for Web Based Java Software Development

On Java world, there are many popular frameworks targeting web development and increase performance. Most popular technologies are Java Server Faces [23], Spring platform [24], Google Web Toolkit [25]. There are many technologies which are derivatives of this system like Oracle ADF [26], IBM Rapid Application Development [27] or ZK [28], IceFaces frameworks [29] which extends JSF somehow. Spring community provides many tools for developing software applications in many aspects like web services development, Model View Controller Framework or Acegi Security [30]. The main methodology of these systems are based on web development and controlling the workflow of this code system. JSF has many capabilities of getting extended. However, the systems are not capable of desktop like applications, XY layout and only SOA based implementation.

Chapter 4
SOA Based Web 2.0 Development System

In this chapter SOA system of the Ajax framework is explained. We first describe the xml system of the platform and then SOA architecture of the server mechanism. On the following parts bean mechanism to extend the system, configuration files and rendering mechanism is discussed.

4.1 XML Structure to Complete an End to End Application

In order to generate an Ajax application or even a Java Desktop application, the mechanism has a unique XML structure where it is suitable to build a complete application. The dtd of the XML structure is given in Figure 4.2. The system both generating a desktop application and a web application is illustrated in Figure 4.1.

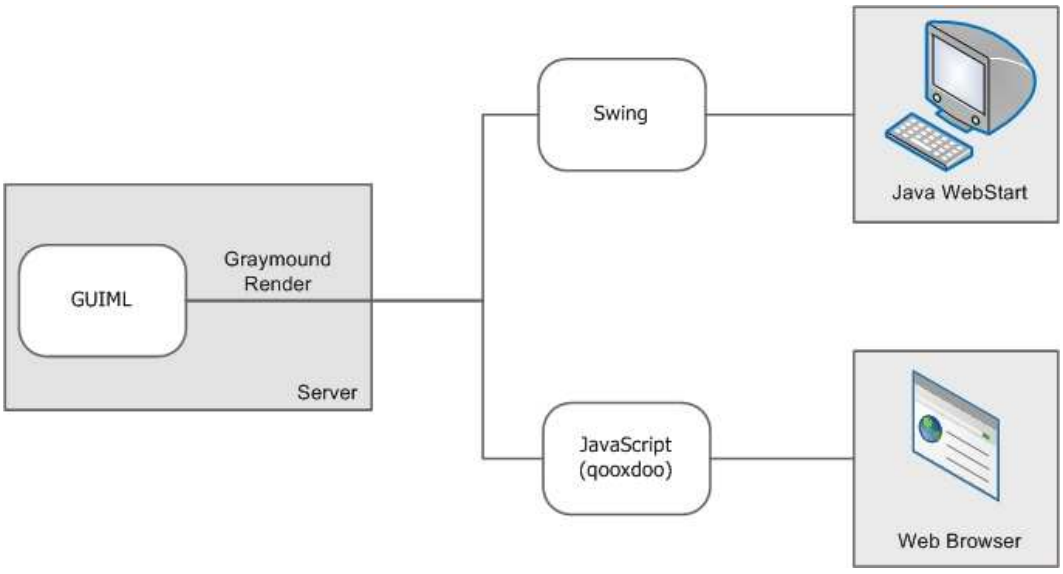

Figure 4.1 Client Server Communication Mechanism

The description of this XML structure is given in Figure 4.2. With the help of this XML model, it is possible to generate an application definition using this XML system. It has a well formed and generic structure; hence adding new components to this system is easy. Another advantage of this xml system is, it has no relation or bounds with any programming language or technology. Ajax, Swing, .NET, Flash or any GUI based development technologies.

```
<!ELEMENT guiml            (page,actions?,services?,variables?)>
<!ELEMENT page       (layout,bean*)     >
<!ELEMENT layout     (properties?)      >
<!ATTLIST layout type (XYLayout|BorderLayout|GridLayout) #REQUIRED>
<!ELEMENT bean       (layout?,properties?,listeners?)>
<!ATTLIST bean type              CDATA #REQUIRED>
<!ATTLIST bean layoutConstraint CDATA #REQUIRED>
<!ATTLIST bean name              CDATA #REQUIRED>
<!ELEMENT properties (property)  >
<!ELEMENT property (data*)>
<!ATTLIST property name CDATA #REQUIRED>
<!ELEMENT data >
<!ATTLIST data en CDATA #REQUIRED>
<!ELEMENT listeners  (listener)>
<!ELEMENT listener   (listener-method)>
<!ATTLIST listener name CDATA #REQUIRED>
<!ELEMENT listener-method>
<!ATTLIST listener-method name CDATA #REQUIRED>
<!ATTLIST listener-method action CDATA #REQUIRED>
<!ELEMENT actions (action+)>
<!ELEMENT action (call+)>
<!ATTLIST action name CDATA #REQUIRED>
<!ELEMENT call (call+)>
<!ATTLIST call id          CDATA #REQUIRED>
<!ATTLIST call type        CDATA #REQUIRED>
<!ATTLIST call method      CDATA #IMPLIED>
<!ELEMENT services (service+)>
<!ELEMENT service (inputs?,outputs?)>
<!ATTLIST service name           CDATA #REQUIRED>
<!ATTLIST service target   CDATA #IMPLIED>
<!ELEMENT inputs (input+)>
<!ELEMENT input (call)>
<!ATTLIST input name CDATA #REQUIRED>
<!ELEMENT outputs (output+)>
<!ELEMENT output (call)>
<!ATTLIST output name      CDATA #REQUIRED>
<!ELEMENT variables (variable+)>
<!ELEMENT variable>
<!ATTLIST variable name          CDATA #REQUIRED>
<!ATTLIST variable value   CDATA #IMPLIED>
```

Figure 4.2 GUIML Page Structure Building Framework Applications

4.2 SOA Architecture and Server Side Mechanism

As stated in the initial chapters it is more reusable and flexible to use service oriented architecture in the client server communication. Table 4.1 shows the general structure and architecture of client server side communication.

Table 4.1 The Definitions of XML Structure

Tag	Definition	Sub Tags	Parent Tag	Attributes
Guiml	Guiml is the XML definition tag. This is the out most tag, to define that it is an guiml application. It contains the application components underneath.	Page (required), actions (optional), services (optional), variables (optional)	NONE	NONE
Page	The page tag gives that it is a page	Layout (required), bean (zero or more bean tag)	guiml	NONE
layout	The page tag gives that it is a page	Layout (required), bean (zero or more bean tag)	Guiml	type (required) alternatives are xylayout, orderlayout, gridlayout
Bean	Bean is a noteworthy tag in the xml structure. It is used to list the application components such as text box, combo box, table etc inside the application. The list of beans is dynamic and users are able to extend this part with their own requirements.	Layout (optional). This is used for layout structure of the specific bean. Inside an application panels, tabbed panes etc. might have their own layout. If not specified it gets this value from super. Properties (optional). ists the properties of the beans. The lists of bean properties are derived from bean definition xml's. Listeners (optional). Listeners are used to associate specific bean events to some actions. For example, when button is pressed, do X action. Or when focus lost in a text box do Y action.	page	NONE

Table 4.1 The Definitions of XML Structure (cont'd)

properties	Properties tag is used to hold specific properties for components inside the XML structure.	property	Page, beans	NONE
Property	Property is used to store individual property of beans, pages etc.	NONE	Page, beans, properties	It has name attribute with data, to store the named property
data	Data is used to store the value of the selected element property. The list of properties are listed in bean definition files. They are stated in the following parts of the thesis.		property	en (default English value, others are up to the languages)
listeners	They list the listener data in order to bind with the beans. It has the list of listeners underneath.	listener	Beans	NONE
listener	The list of listeners, bind with actions. When button pressed, when focus lost etc. It has the listener name and the listener action. These are defined in bean definition xml's in the following chapters.	Listener-method	listeners	NONE
actions	The list of actions is inside these element. These are stored under guiml bean. It has list of whole actions in a page. They bind with events.	Action	actions	NONE

Table 4.1 The Definitions of XML Structure (cont'd)

actions	The list of actions is inside these element. These are stored under guiml bean. It has list of whole actions in a page. They bind with events.	Action	actions	NONE
call	In all actions, there is a call. These are similar to statements in a function. Call can be actions too. This way it becomes a recursive action. They also call services. It has action id, its type, method , text etc. and method name.	Call		Id, type, method
service	Services are the server side communication mechanism. They have the inputs, collected from page and output as a result of the server side return mechanism.	Input, output	Guiml	Name , target
Input , output	Input and output elements are used to send data to server andreceive a data from server side code. It is the way to communicate to the server side Java mechanism. All services are transaction based.		Service	name
	Variables are the client side xml based page variables. With this information. They are generic type object and can be used inside the xml for comparison etc. requirements.	Guiml		Name, value

Table 4.1 The Definitions of XML Structure (cont'd)

Input , output	Input and output elements are used to send data to server and receive a data from server side code. It is the way to communicate to the server side Java mechanism. All services are transaction based.		Service	name
Variable	Variables are the client side xml based page variables. With this information. They are generic type object and can be used inside the xml for comparison etc. requirements.	Guiml		Name, value

The client side may vary with different protocols and technologies. In Figure 4.3 client server side mechanism with properties discussed are illustrated. Using a standard technology like SOAP can solve the varying technologies problem and all clients may communicate with the client with server. However in high performance environments, like banks, telecom companies etc. where there are a huge number of clients the communication delay and processing affects the system a lot. To solve this issue Client side uses a "context" to call the server and different context have different communication protocols. A Java servlet checks this context and responds the clients differently. In Figure 4.4 there is the sample client context configuration file. On the server context parts there are currently four contexts. Java is for Swing clients to directly connect to the server side with object serialization method. Thus rather than xml's passing from one side to another, it serializes the objects and sends it to other side. JSON stands for Javascript Object Notation. Thus, via this notation it is possible to send directly objects to Javascript without parsing the primitive object types. Another one is set for reports. Report mechanism is different than service executing. It includes compiling, checking and generating reports on the server side and passing parameters to the reports when necessary. The system currently uses

Jasper Reports for the report mechanism. The last one is the SOAP context. It is the standard SOAP communication protocol mechanism, for third party applications to call Web Services from this SOA based service mechanism.

The server side mechanism has other properties of transaction management database persistence and session management. For the transaction mechanism, the system uses a stateless session bean. This type of EJB (Enterprise Java Bean) handles the transactions. It is stateless and all services are single transactions with ACID properties. With Java 6.0 there is JTA standard. Java Transaction Agent are the persistence standard and since system supports that standard Hibernate or any other persistence framework can easily be connected to the service execution mechanism. In Figure 4.5 a sample context code is shown.

The servlet gets the request with different context and starts a transaction with an EJB. The standard for client server parameter passing is via HashMap. By definition HashMap is data structure that associates keys with values. The primary operation it supports efficiently is a *lookup*: given a key (e.g. a person's name), find the corresponding value (e.g. that person's telephone number). It works by transforming the key using a hash function into a *hash*, a number that is used as an index in an array to locate the desired location ("bucket") where the values should be.

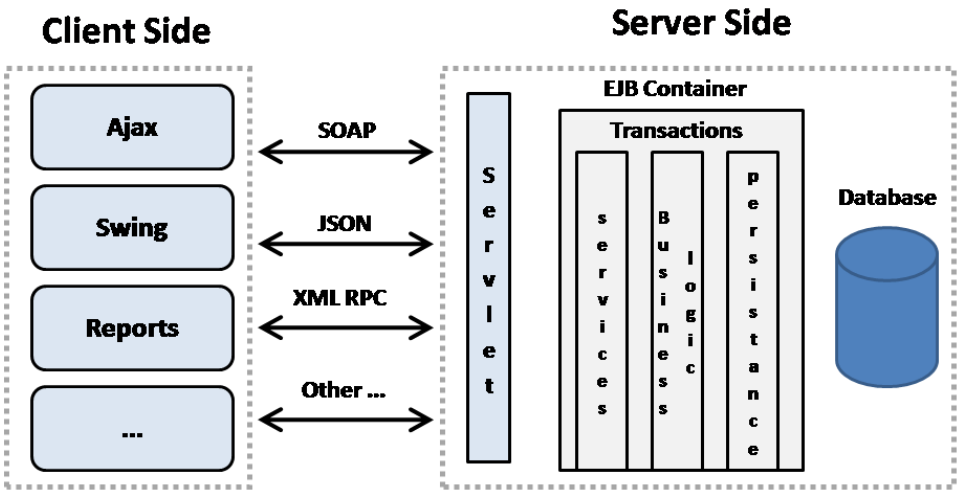

Figure 4.3 Server Side SOA Architecture

```
<?xml version="1.0" encoding="UTF-8" ?>
<gm-configuration>
- <server-configuration>
  - <server-properties>
      <property name="graymound.home" value="C:\GMDevelopment/Graymound" />
      <property name="ejb.jndi.name" value="GMEJB" />
      <property name="transaction.coordinator"
        value="com.graymound.server.coordinator.GMTransactionCoordinatorDefault" />
      <property name="datasource" value="java:/GraymoundDS" />
      <property name="LOGIN" value="GM_ADM_USER_AUTHENTICATE" />
    </server-properties>
  - <server-startup>
      <class name="com.graymound.server.GMServerStartupHibernate" />
      <class name="com.graymound.server.GMServerStartupDefault" />
    </server-startup>
  - <server-contexts>
      <context name="/JAVA" class="com.graymound.server.servlet.context.GMContextJava" />
      <context name="/JSON" class="com.graymound.server.servlet.context.GMContextJson" />
      <context name="/REPORT" class="com.graymound.server.servlet.context.GMContextReport" />
      <context name="/SOAP" class="com.graymound.server.servlet.context.GMContextSoap" />
    </server-contexts>
  </server-configuration>
</gm-configuration>
```

Figure 4.4 GM Server Configuration with Different Context Bean

The servlet gets the request with different context and starts a transaction with an EJB. The standard for client server parameter passing is via HashMap. By definition HashMap is data structure that associates keys with values. The primary operation it supports efficiently is a *lookup*: given a key (e.g. a person's name), find the corresponding value (e.g. that person's telephone number). It works by transforming the key using a hash function into a *hash*, a number that is used as an index in an array to locate the desired location ("bucket") where the values should be.

Using HashMap, the functions has no longer needs to change. The parameters can dynamically bind inside the HashMap. For example think there is a function foo (Name, Surname) with two parameters from the client side, Name and Surname. You need to change the function parameters to add ID to the function foo (Name, Surname, ID). With that change all functions that are calling them will give compile errors. Also it is practically impossible to call this function generic. When all the functions gets a HashMap bag like foo (HashMap<String,Object> iMap), you can

add whole parameters inside this HashMap and when the number of parameters change dynamically.

Client side calls services generically with a service name. Like ACC_ACCOUNT_SAVE with parameters. The developer needs to register this service to the server in order to bind it. In Java 6.0 there is a notion of annotations. Annotation is extra information asserted with a particular point in a document or other piece of information using tags. An annotation, in the Java computer programming language, is a special form of syntactic metadata that can be added to Java source code. Classes, methods, variables, parameters and packages may be annotated. Java annotations are reflective in that they are embedded in class files generated by the compiler and may be retained by the Java VM to be made retrievable at run-time.

Using the annotations it becomes possible to call a service on the server side. In Figure 4.6 There is a sample server side service code with annotations. The annotation is named as GraymoundService and developer can extract all the necessary parameters from this annotation. Calling the services from the client side and using input/outputs are defined in the design studio section.

4.3 Beans to provide an application

In order to fill out the xml structure and fill to form an application there is a notion of "beans". They are the defined text boxes, combos, tables etc. In Table 4.2 there are the full lists of beans currently supported by Ajax. Their listeners and codes will be in the attachment.

4.4 Configuration Files and Their Definitions for This System to Work

The system has many configuration files for client and server side. In order for the Ajax front end to work or server side to communicate with the client end side. In figure 4.7 a sample configuration files is shown. The configuration files are

completely XML based and used by the system. The definition of the configuration files are shown in Table 4.3. The full list of XML files are given in Appendix B. System has a complete configuration capability thus it is easy and possible to change all classes to redirect the layers to a different group of execution mechanism. For example, transactions are handled in an EJB inside the application server. However, users might want to change the transaction execution mechanism and use another method for this purpose. Without changing any code, it is possible to make it inside the configuration files.

```
public class GMContextJava extends GMContext {
    private String      sessionID   ;
    public GMContextJava() {
        super();
    }
    @Override
    protected GMRequest createRequest(HttpServletRequest servletRequest) throws IOException{
        InputStream iStream = null;
        try {
            iStream = servletRequest.getInputStream();
            GMRequest request = (GMRequest)GMToolkit.deserialize
            (GMToolkit.inflate(GMToolkit.read(iStream)));
            this.sessionID = request.getHeader("JSESSIONID");
            return request;
        } catch (DataFormatException e) {
            throw new IOException(e.getMessage());
        } finally {
            GMToolkit.close(iStream);
        }
    }
    @Override
    protected byte[] createResponse(HttpServletResponse servletResponse,
            GMResponse response) throws IOException {
        response.addHeader("JSESSIONID", getSessionID());
        return GMToolkit.deflate(GMToolkit.serialize(response));
    }
```

Figure 4.5 Sample Context Bean Code

```
//annotation to call the service
@GraymoundService("ACC_ACCOUNT_SAVE")

// the method has to be public, static with inputs and outputs of HashMap
public static HashMap<String, Object> saveAccount (HashMap<String, Object> iMap)
{
    String accNname  = (String)iMap.get("ACC_NAME");
    Integer accNumber = (Integer)iMap.get("ACC_NUMBER");
    String accDefinition  = (String)iMap.get("ACC_DEFINITION");
    String accCode  = (String)iMap.get("ACC_CODE");

    HashMap<String, Object> oMap = new HashMap<String, Object>();
    oMap.put("VALUE", "server");
    return oMap;
}
```

Figure 4.6 Sample Service Code

Table 4.2 List of Beans and Their Descriptions on the System

Name	Definition	Methods and Events
Utility Beans		
Frame	This is the Java Frame Bean for storing the beans underneath.	**Methods:** open, exit
MainOutlook	The screen design varies on different applications. MainOutlook is the page system with a menu on the left.	NONE
MainInternet	MainInternet is the default page with a menu on top	NONE
MainSimple	MainSimple is the simple browser window without a menu and shows beans	NONE
Browser	Browser bean is used to open a new page container and show other pages	**Methods:** open, openAndKeepCurrent, openandRemoveCurrent, close, setvariable
Compare	Compare bean is used to compare different type of objects with each other. It has methods for isequal is null, less than or equal or not etc.	**Methods:** equals, lessThan, lessThanorEqual, greaterThan, greaterThanorEqual, isNull
Date	Date bean is used for getting the date and making date operations on an object.	**Methods:** getNow, addYear, addMonth, addDay
KeyEvent	Key event is used to make shortcuts on a specific bean.	**Methods:** createKeyEvent, createKeyEventFor
Logic	Logic bean is used to change Boolean operations	**Methods:** not, and, or
Math	Math bean is used to convert Strings to bigdecimal, integer or any other class type and makes arithmethic operations between numbers	**Methods:** sum, sumInt, subtract, multiply, divide, pow, isZero, toBigdecimal, toInteger, round
MessageBox	Message box is the warning, info or error boxes. Errors show automatically via server side exception	**Methods:** info, warn
Popup	Popup bean is used to open a popup of choice. User can set and get parameters to the popup. Popup shows two buttons, ok and cancel at the bottom. User can set actions for these operations.	**Methods:** setParameter, getParameter, setActionForCancel, setActionForOk, show
Session	Session is for storing the session information. Variables of choice individually or as a group can be mapped to session. Users can put and get from session and clear it when necessary.	**Methods:** clear, put, get, getLanguage, setLanguage, remove, putToGroup, getFromGroup, removeFromGroup, containsKey

Table 4.2 List of Beans and Their Descriptions on the System (cont'd)

StatusBar	Status bar is the bottom line in the application. User can put information to this status bar.	**Methods:** setItem, removeItem
String	String operations are done with this utility bean. System concats, gets substring, checks if the string is empty or not etc.	**Methods:** isEmpty, length, concat, subString
ToolBar	Toolbar is the topmost part with buttons. The buttons and their corresponding actions are set in the configuration files.	**Methods:** setActionEnabled
Beans		
Button	Button is the standard button in Ajax and Swing	**Methods:** getThis, requestFocus **Events:** ActionPerfomed
DynamicButtonGroup	DynamicButtonGroup is a set of radiobuttons. It acts like a combo, and events can be fired on top of that.	**Methods:** setModelData, setSelectedItem, setSelectedItemWithKey, getSelectedValue
CheckBox	Checkbox is the standard checkbox it has some properties in the configuration file.	**Methods:** isSelected, setSelected, getThis, requestFocus **Events:** actionPerformed
ComboBox	Combo box is the standard combo box. User can select items from that	**Methods:** setModelData, addItem, getSelection, getNodeData, setSelectedItemWithKey, getThis, clear, clearModelData, requestFocus **Events:** actionPerformed
ContentAssist	Content assist is the tool where the context is filled automatically while the user is typing. It is extended from textbox.	**Methods:** SetModelData, getData, getText, getThis, requestFocus **Events:** focusLost
CurrencyField	Currency field is for money. User can define separator and how many digits.	**Methods:** getCurrency, requstFocus, getThis **Events:** actionPerformed, FocusLost, onChange
DatePicker	Date picker is for selecting a date.	**Methods:** getDate, setDate, getThis, requestFocus **Events:** ActionPerformed, onChange, focusLost
DynamicPart	Dynamic part is for storing a page inside a page. Like frames in the web technology. This way it is possible to use a page again and again.	**Methods:** setGuiml, callAction, setVariable, getVariable
File	File bean is for selecting and uploading a file to the server side.	**Methods:** getContent, download, getText, setText, clear, getFileName
ImageButton	Image button is same as button with a complete image in the front end.	**Events:** ActionPerformed
Label	Label is the text on the pages.	NONE

Table 4.2 List of Beans and Their Descriptions on the System (cont'd)

Link	Link extends Label, with a link to open a url.	**Methods:** open, openwithURL
List	List is the, list of items where user can select one.	**Methods:** getModelData, setModelData, getNodeData, setNodeData, createNode, removeNode, getThis, requestFocus **Events:** ValueChanged
MaskField	Mask field is the custom user defined text box, where user can define its own mask. Like telephone numbers, identity code etc.	**Methods:** getThis, requestFocus **Events:** ActionPerformed, FocusLost
Page	Page bean stores and opens a page.	**Methods:** getThis, clear, isEnabled, setEnabled
Panel	Panel is for storing beans underneath	**Methods:** toggle, getThis, clear, isEnabled, setEnabled
PasswordField	Password field is like a textbox with asterix character in the front end.	**Methods:** getPassword, requestFocus, getThis **Events:** actionPerformed, focusLost
RadioButton	Radio Button is selection from various items.	**Methods:** isSelected, requestFocus, setSelected **Events:** actionPerformed
Report	Report bean shows a report underneath. On Ajax server side generates a pdf, and report area shows a pdf in that part. Swing generates a Jasper Report and shows a report via jasper viewer aplet.	**Methods:** setReport, addParameter, execute, print, getThis
TabbedPane	Tabbed pane stores panels under various tabs.	**Methods:** getSelectedIndex, requestFocus, setSelectedIndex, setEnableMask **Events:** stateChanged
Table	Table is the standard generic table. User can add different type of columns to that table.	**Methods:** GetPointedColumnIdentified, removeSelectedRow, appendRow, clear, RequestFocus, getThis, addRow, setModelData, getSelectedRow, setSelectedRow, setSelectedRowWith, sum, setValueAt, setColumnVisible, getValueAt **Events:** MouseListener, ListSelectionListener, TableModelListener.
TableColumn	Table column bean is the different type of columns on a table. This is a generic super class. Other beans extend this one, and table may contain many column types.	**Methods:** SetEditorData, getData **Events:** columnChanged
TextArea	Text area stores text in multiple lines.	**Methods:** getThis, requestFocus
TextField	Textfield stores text. User can select different types of input	**Methods:** getThis, requestFocus **Events:** actionPerformed, onChange, focusLost

Table 4.2 List of Beans and Their Descriptions on the System (cont'd)

CheckBoxTree	Checkbox tree is a hierarchical tree with option to select top or bottom items.	
Tree	Tree is showing the lines in a tree order.	**Methods:** getRoot, getSelectedNode, setSelectedNode, getNodeData, setNodeData, clear, getThis, addNodeToParent, expandAll, collapseAll, hasChildren, addNode, setModelData, getModelData, requestFocus, removeSelectedNode **Events:** MouseListener, TreeSelectionListener
TreeTable	Tree table is a tree with columns.	**Methods:** getNodeData, getNodeDataWithSearchKey, setNodeData, setNodeDataWithSearchKey, clear, getThis, expandAll, collapseAll, setModelData, getModelData **Events:** TreeSelectionListener, CellEditorListener

```
<?xml version="1.0" encoding="UTF-8" ?>
- <configuration>
  - <properties>
      <property name="size" value="1024,768" />
      <property name="resizable" value="true" />
      <property name="lookAndFeel" value="com.graymound.uiml.browser.util.GMLookAndFeel" />
      <property name="loginPage" value="GM_ADM_LOGIN.guiml" />
      <property name="firstPage" value="0" />
      <property name="debug" value="true" />
    </properties>
  </configuration>
```

Figure 4.7 Sample Configuration File

Table 4.3 XML Configuration Files

Client Side	
File	Explanation
GMBeans.xml	Bean definition files are listed on this XML. System has the formation of "beans" as working small widgets on the client side. The list of the beans is shown on this configuration file thus it can increase dynamically. Also, framework user interface plugin uses this system as the list of items on palette.

Table 4.3 XML Configuration Files (cont'd)

GMBrowserConfiguration.xml	The client side of the framework has a browser which converts the XML files to an application dynamically. This configuration file stores the size, look and feel and other visual related configuration.
GMBrowserToolBar.xml	The browser has a shared toolbar on top part. The GM Browser Configuration file stores the toolbar menu items and their related actions, which action will be called on press.
GMConnection.xml	The client system connects server side with a url, username password and other connection related security information. For a fast start this information can be stored on this XML file.
GMLayouts.xml	Layouts of the client side system. The layout definitions, their corresponding java files and properties are listed.
GMListeners.xml	On client side, there are different listeners on beans. These listeners are listed on this file. Some examples are cellEditorListener, FocusListener, DocumentListener.
GMPalette.xml	On user interface development plugin, the list of possible items is listed here. Using this file, palette is dynamically formed.
GMResourceFactory.xml	This file is used to store the connection information between client and server side. Connection type, the context and the url is stored on this file. The difference between the connection xml file is, this is used on client system services, and the other is used on services called in java codes.
GMToolkit.xml	Language information is stored on this configuration file. There are keys and the corresponding values.
Server Side	
GMCacheConfiguration.xml	It stores the cache configuration of the server side. The framework has cache integration of variety of mechanisms.
GMClusterConfiguration.xml	Cluster configuration stores the configuration of cluster mechanism in distributed cache and service mechanism.
GMServerConfiguration.xml	Server configuration is the server side, transaction, data source context and other server related configuration files.
GMServerDatasources.xml	GMServerDatasources stores the data source information for service related components or reading the service methods. It normally reads these from the application server datasource, however this is an external mechanism for the same purpose.
GMServiceConfiguration.xml	This stores the service execution mechanism and the class. Users can either store the service classes in database or in this file as a configuration.

4.5 Rendering Mechanism for XML Ajax

The front end GUI system has XML based definition of the front end. Ajax mechanism on the server side renders the XML, converts this as a standard Swing, and the Qooxdoo framework, which automatically retrieves this swing and builds up an Ajax GUI. This system is illustrated on Figure 4.8. The system also has a custom implemented service calling mechanism for server side. The actions and if-then-else or while loop properties on XML are the basic functions triggered by events to handle the simple requests on client side.

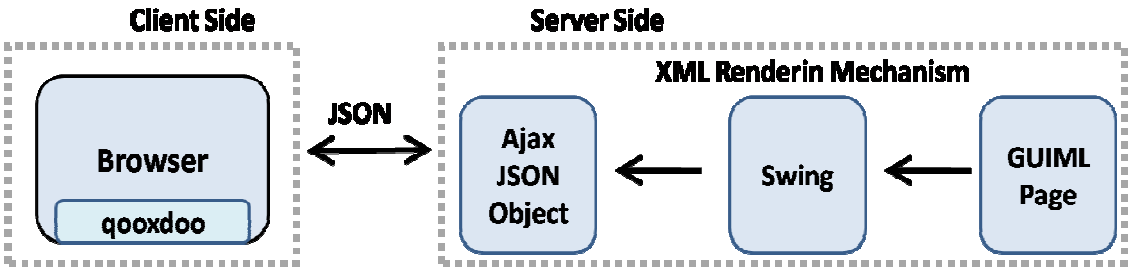

Figure 4.8 Ajax Rendering Mechanism

On the client side of the system all the beans are implemented in Javascript files. So that when parser gets into these bean definitions it uses the referenced files. A sample bean code is shown in Figure 4.9.

4.5.1 About Qooxdoo

Qooxdoo is a comprehensive and innovative Ajax application framework. It is entirely class-based and tries to leverage the features of object-oriented JavaScript. It is fully based on namespaces and does not extend native JavaScript types to allow for easy integration with other libraries and existing user code. The system supports most of the modern browsers and targeting to become free of memory leaks. It comes with a comprehensive API reference that is auto-generated from Javadoc-like comments and from the syntax tree representing the code. The fast and complete

JavaScript parser allows doc generation, but is an integral part of the automatic build process that makes optimizing, compressing, linking and deployment of custom applications. The system also has internalization and localization.

```
qx.Class.define("com.graymound.uiml.bean.JGMComboBox", {
    extend      : qx.ui.form.ComboBox,
    construct   : function() {
        this.base(arguments);
        this.fireEvent = true;
        this._getChildControl("textfield").setReadOnly(true);
    },
    members     : {
        fireEvent   : null,
        setTabOrder : function(){
        },
        setModelData: function(data) {
            if (data != null) {
                this.fireEvent = false;
                var list = this._getChildControl("list");
                list.removeAll();
                for (var i = 0; i < data.length; i++) {
                    var item = new qx.ui.form.ListItem(data[i].NAME,null,null);
                    item.setUserData("data",data[i]);
                    list.add(item);
                }
                this.fireEvent = true;
            }
        },
        getNodeData : function(key){
            if (this.getSelection() != null) {
                var data = this._getChildControl("list").getSelectedItem().getUserData("data");
                if (data != null) {
                    return data[key];
                }
            }
        }
    }
});
```

Figure 4.9 Sample Bean Code for Combo Box

4.6 Service Execution Mechanism

Service execution mechanism is the key part in communication with the server side. Client side development system is based on the service calling mechanism for any server side Actions. The XML system stores the GUI definitions, variables and service information. The service system uses a HashMap as a parameter to the service. This way if the parameters on a service increase, HashMap can dynamically bind any number of parameters with a “key, value” system. The service reaches the

server side with a HashMap package and the name of the service to be called. On the server side there is a servlet with different context. With the help of different contexts, it is possible to call services with different standards. For example JAVA context receives objects as the service parameter and the whole request is a JAVA object. The SOAP context receives a web service implementation with a standard XML. The Ajax system uses a JSON context which stands for Java Script Object Notification. This system is parsed on the server side, then a stateless session bean EJB is called to start a transaction. The whole service is transactional and provides ACID features. It requires an application server to run this single service execution EJB. The service execution class is configured in an XML thus developers can implement their own service execution. The input and output parameters of the service are primitive type data structures since Ajax and Web Browser does not support any object oriented type or complex data structures. A sample execute service method is given on Figure 4.10.

```
executeService  : function(serviceName,iMap){
    var serverData  = {};
    if (this.getSessionID()){
        serverData.sessionId = this.getSessionID()
    }

    serverData.serviceName  = serviceName;

    var rpc          = new qx.io.remote.Rpc(this.getURL("Server/JSON"),null);
    rpc.setTimeout(10000);
    rpc.setServerData(serverData);

    var response = rpc.callSync("EXECUTE_SERVICE",iMap);
    this.setSessionID(response.sessionId);
    return  response;
},
```

Figure 4.10 Execute Service Method on Ajax

4.7 Jgroups

Jgroups is a group multicasting solution used in the system in clustering and caching. This project is worldwide popular and in the hearth of largest Java projects. It has a well implemented and performance proved application system.

Jgroups is defined as toolkit for reliable group communication [31]. Processes can join a group, send messages to all members or single members and receive messages from members in the group. The system keeps track of the members in every group, and notifies group members when a new member joins, or an existing member leaves or crashes. A group is identified by its name. Groups do not have to be created explicitly; when a process joins a non-existing group, that group will be created automatically. Member processes of a group can be located on the same host, within the same LAN, or across a WAN. A member can be part of multiple groups.

4.8 Cache Mechanism

The system is organized to implement Enterprise Projects. These projects have a high load and capacity requirements. In order to fulfill this request systems mostly require clustering to become both productive and fault tolerant. Cache is the key part in increasing the efficiency of the system. The architecture has a unique cache mechanism to store different cache architectures on its own. The caching is a layer to "put" to the cache and "get" from the cache. On Figure 4.11 sample caching code is shown.

```
@SuppressWarnings("unchecked")
public void messageRecieved(GMClusterMessage message) {
    String manager = message.getHeader("MANAGER");
    if (manager.equals(GMCacheManagerReplication.class.getName())){
        String action = message.getHeader("ACTION");
        if ("SET".equals(action)){
            GMCacheEntry    entry       = (GMCacheEntry)message.getData();
            GMCache         cache       = GMCacheFactory.getInstance().getCache(entry.getCacheName());
            cache.putInternal(action, entry);
        } else if ("PUT".equals(action)){
            GMCacheEntry    entry       = (GMCacheEntry)message.getData();
            GMCache         cache       = GMCacheFactory.getInstance().getCache(entry.getCacheName());
            cache.putInternal(action, entry);
        } else if ("REMOVE".equals(action)){
            GMCacheEntry entry = (GMCacheEntry)message.getData();
            GMCache         cache       = GMCacheFactory.getInstance().getCache(entry.getCacheName());
            cache.removeInternal(entry.getKey());
        } else if ("FILL".equals(action)){
            HashMap<?, ?> state = (HashMap<?,?>)message.getData();
            for (Iterator<?> iterator = state.keySet().iterator(); iterator.hasNext();) {
                String cacheName = (String) iterator.next();
                GMCache cache = GMCacheFactory.getInstance().getCache(cacheName);
                cache.putAll((Collection<GMCacheEntry>)state.get(cacheName));
            }
        }
    }
}
```

Figure 4.11 Cache Mechanism Sample Code

In technical details, a cache is organized as a tree, with a single root. Each node in the tree essentially contains a Map, which acts as a store for key/value pairs. The only requirement placed on objects that are cached is that they implement `java.io.Serializable`. This way cache items can serialize, sent to another cache group and desterilized. Cache mechanism can be either local or replicated. Local trees exist only inside the JVM in which they are created, whereas replicated trees propagate any changes to some or all other trees in the same cluster. A cluster may span different hosts on a network or just different JVMs on a single host. JVMs on a single host. Figure 4.12 shows communication between two clients and the Jgroups role on this system.

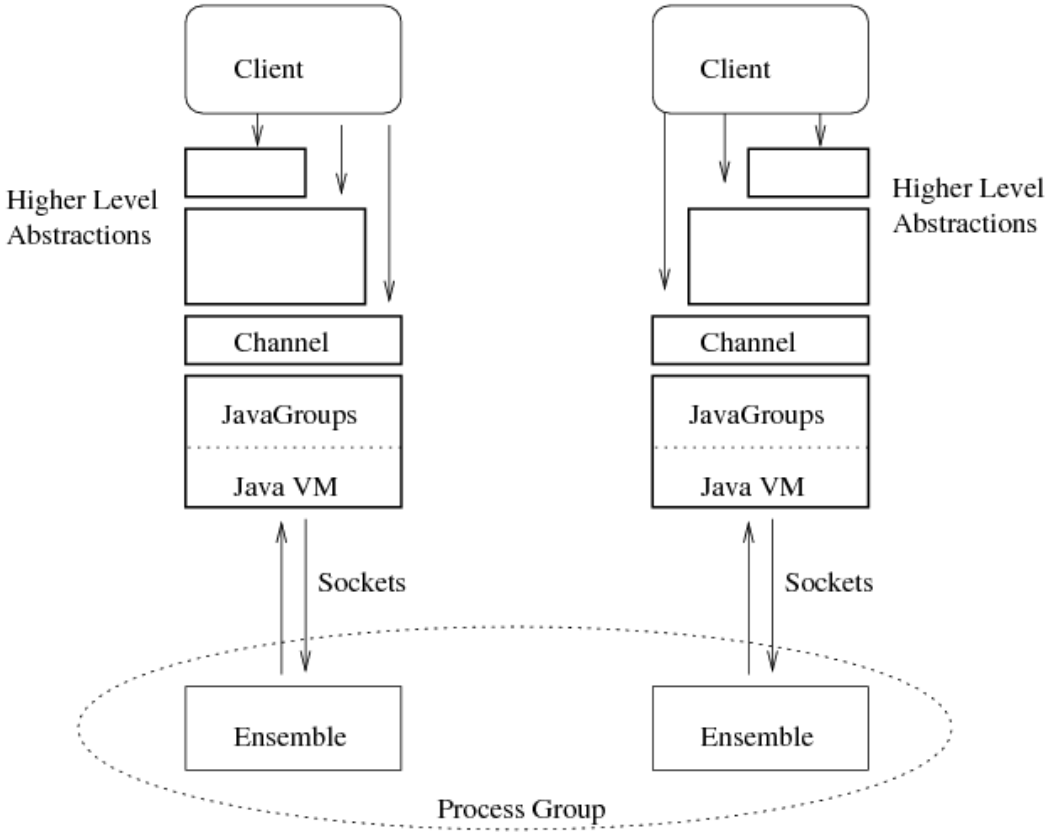

Figure 4.12 Jgroups Architecture

When a change is made to an object in the cache and that change is done in the context of a transaction, the replication of changes is deferred until the transaction commits successfully. All modifications are kept in a list associated with the transaction for the caller. When the transaction commits, cache mechanism replicate the changes. Otherwise, (on a rollback) system simply undo the changes locally resulting in zero network traffic and overhead. For example, if a caller makes 100 modifications and then rolls back the transaction, system does not replicate anything and it does not result any traffic. This cache mechanism is also completely thread-safe. It uses a pessimistic locking scheme for nodes in the tree by default, with an optimistic locking scheme as a configurable option. With pessimistic locking, the degree of concurrency can be tuned using a number of isolation levels, corresponding to database-style transaction isolation levels, i.e., SERIALIZABLE, REPEATABLE_READ, READ_COMMITTED, READ_UNCOMMITTED and NONE.

A Cache consists of a collection of Node instances, organized in a tree structure. Each Node contains a Map which holds the data objects to be cached. It is important to note that the structure is a mathematical tree, and not a graph; each Node has one and only one parent, and the root node is denoted by the constant fully qualified name, ROOT. The reason for organizing nodes as such is to improve concurrent access to data and make replication and persistence more fine-grained.

In Figure 4.13, each box represents a JVM. It shows 2 caches in separate JVMs, replicating data to each other. These VMs can be located on the same physical machine, or on 2 different machines connected by a network link. The underlying group communication between networked nodes is done using Jgroups. Any modifications in one cache instance will be replicated to the other cache. Naturally, developers can have more than 2 caches in a cluster. Depending on the transactional settings, this replication will occur either after each modification or at the end of a transaction, at commit time. When a new cache is created, it can optionally acquire the contents from one of the existing caches on startup.

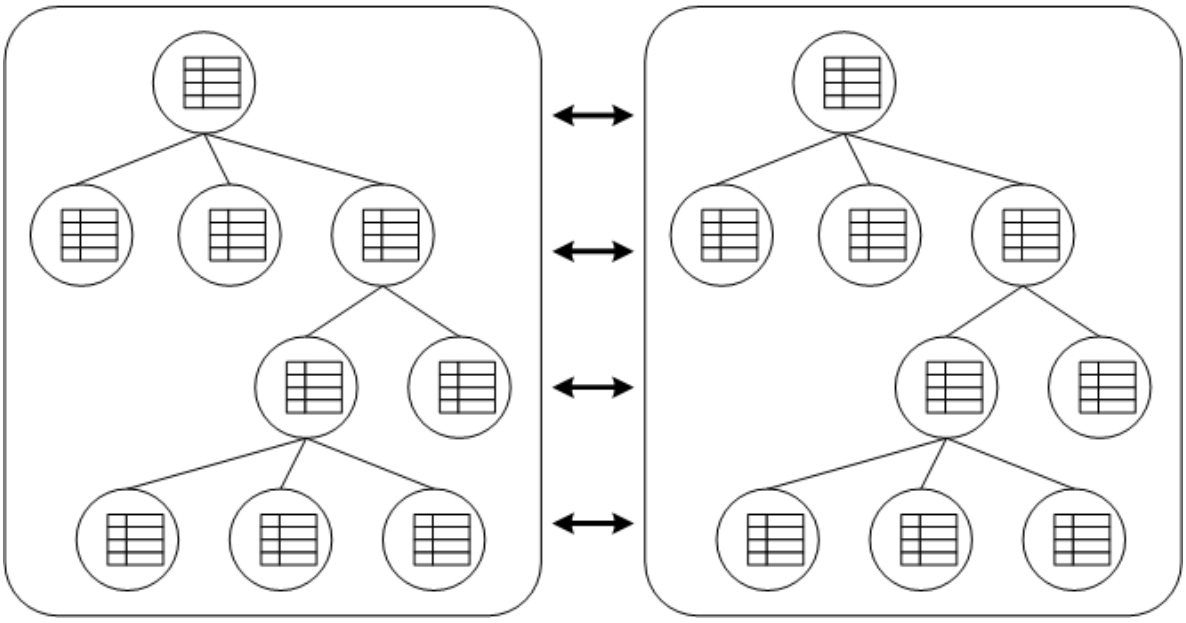

Figure 4.13 Cache Data Structured as a Tree

Chapter 5
Sample Application and Design Studio

This chapter shows the visual part of the plugin with the explanation of Eclipse design studio and then a step by step tutorial of how to generate a first "hello world" application using this tool. It is easy to use and develop an application using this technology. Eclipse is used on client side to inherit all the features of this development environment. Tutorial includes many figures to easily illustrate the development steps. Installation package of the framework can either be downloaded from url http://open.sabanciuniv.edu/graymound or, Appendix B contains the latest version of the development framework.

5.1 Eclipse Plugin

This chapter displays the details of the framework's eclipse plugin design studio and its properties. The eclipse design studio has a properties window, which is available at menu, window - preferences.

Figure 5.1 has the configuration file of the Eclipse framework. Install home is the installation directory of the system. This file is chosen during the installation process of the framework. Login page is the initial page during testing or application development. Normally the system works under a server. During the development phase, it is possible to test the pages one by one. To make session mechanism and server authentication work, system first opens the login page and this page automatically logins and redirects to the development page. Language configuration is the internalization option. The framework has built in multiple language support.

Hence, when a language context is provided either from internet browser or from this configuration file or via regional settings of the operating system, it shows the text with that language file. “GUIML test URL” is the testing URL of the pages. When a page is opened for testing inside the eclipse plugin, it redirects to this URL and then opens the test page. The system automatically searches for server side web services to be able to easily select from client side. Thus the connection driver, connection URL, username and password are for this connection purpose.

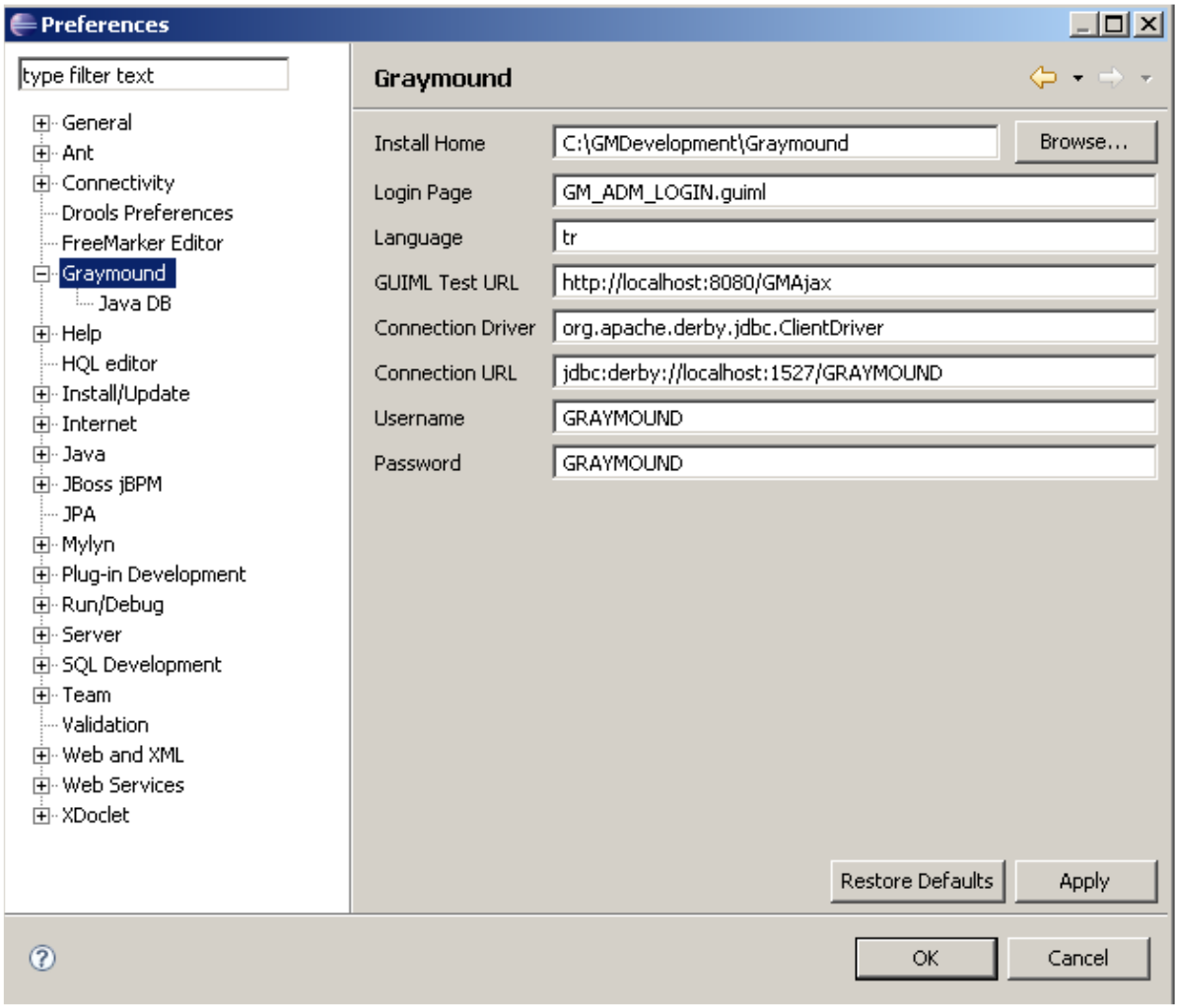

Figure 5.1 Eclipse Configuration Files

Eclipse plugin contains the palette and the XML design studio. In Figure 5.2, a sample application screen on framework design studio is given. Using this studio,

users can select the appropriate bean from the right pane, and drag and drop that to the inner screens. The system has the properties option, thus when an item is selected on the page, its properties automatically listed, where users can edit these properties. All the labels and text has the internalization option, and Turkish and English text can be written at the same time. The screenshot on Figure 5.2 is from a health insurance application of an insurance company.

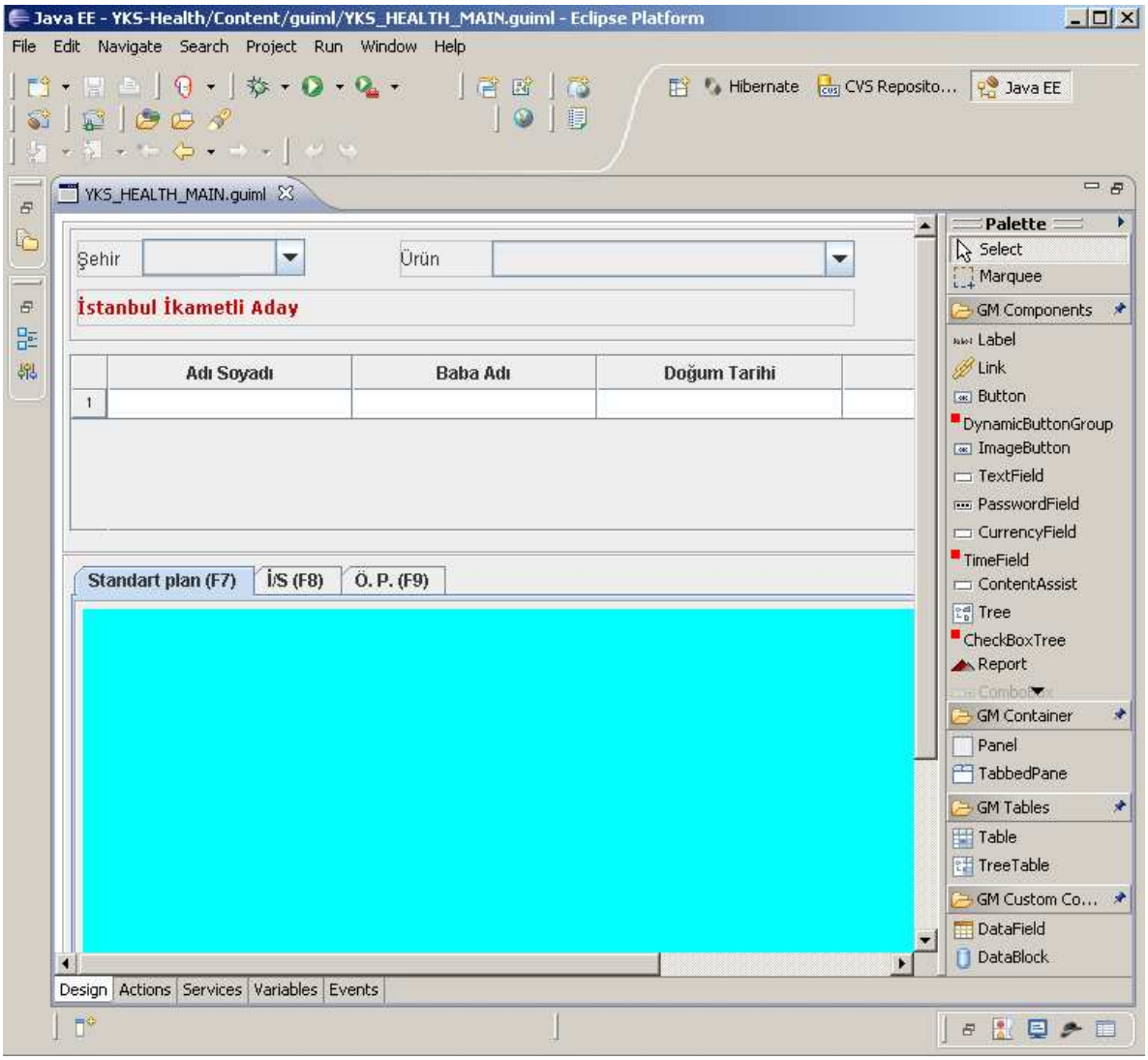

Figure 5.2 Sample Application Screen

The system has the notion of Actions. Actions correspond to the methods in regular code. In an action pane, users can call methods of beans, call services or call actions for recursive action. Also it is possible to use if-then-else block inside the action. With this property primitive controls and methods can be executed on the client side without server side service calling or any other operation. The actions are standard and users cannot write any code on client side. This way all the codes written are based on XML and easily be transformed or maintained. Figure 5.3 gives sample actions tab from an existing project.

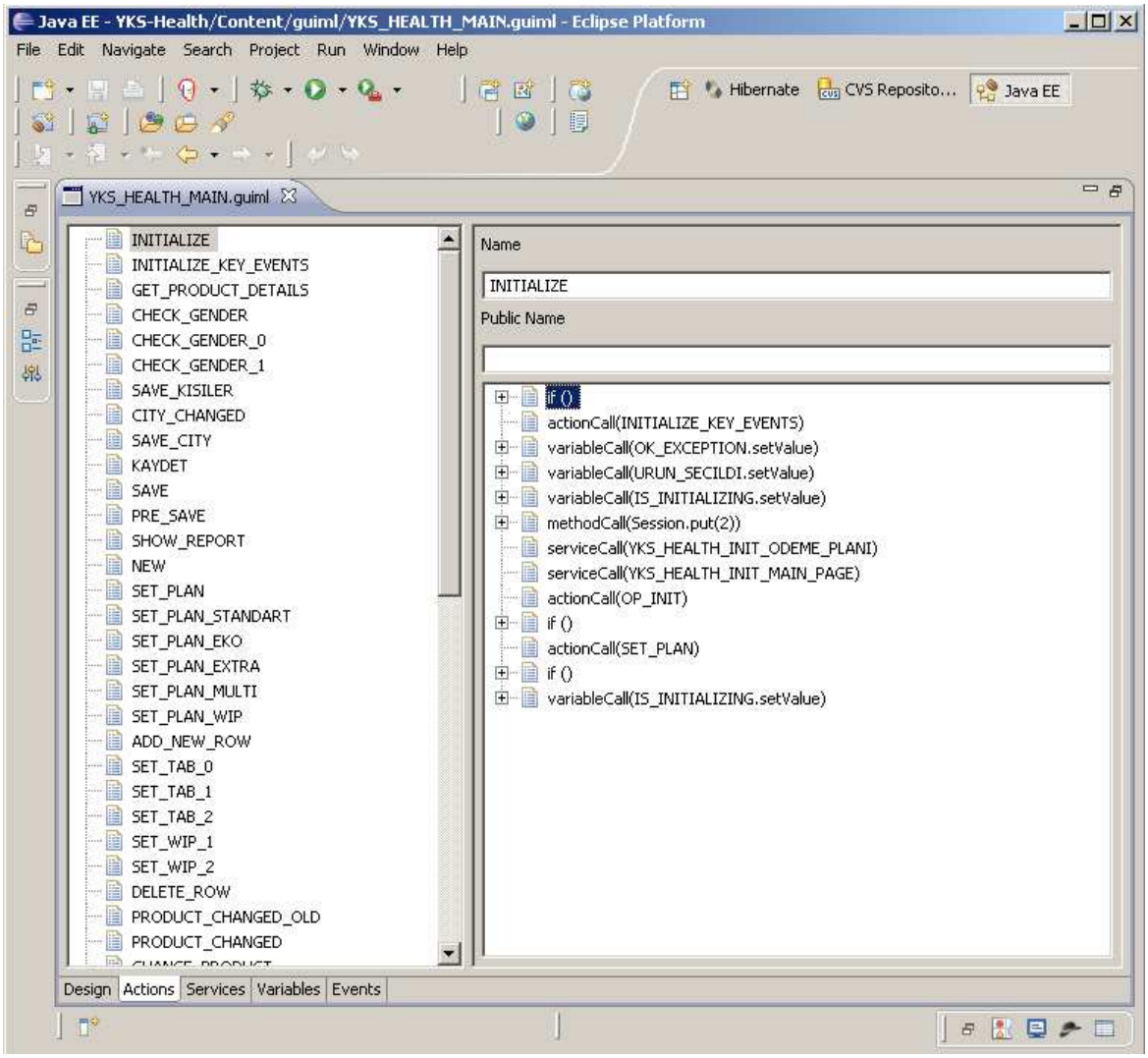

Figure 5.3 Actions Tab to Show Client Side Method Development

Services are the core communication model of this system. A service name is the key and server side must have a corresponding tag starting with an annotation @GraymoundService("Service_name"). The application automatically receives this annotation and marks this piece of code underneath as a service. The service has inputs and outputs. Inputs are the set of parameters sent from client to the server. These can either be variables, static text or parameters received from the client side beans. Outputs are the response of the services. These services can be methods to call a bean function with this parameter. For example, when a service response is returned like NAME, this can be called as txtbox.setText method thus this parametric result will be automatically set to text on the textbox. Figure 5.4 shows a sample services tab.

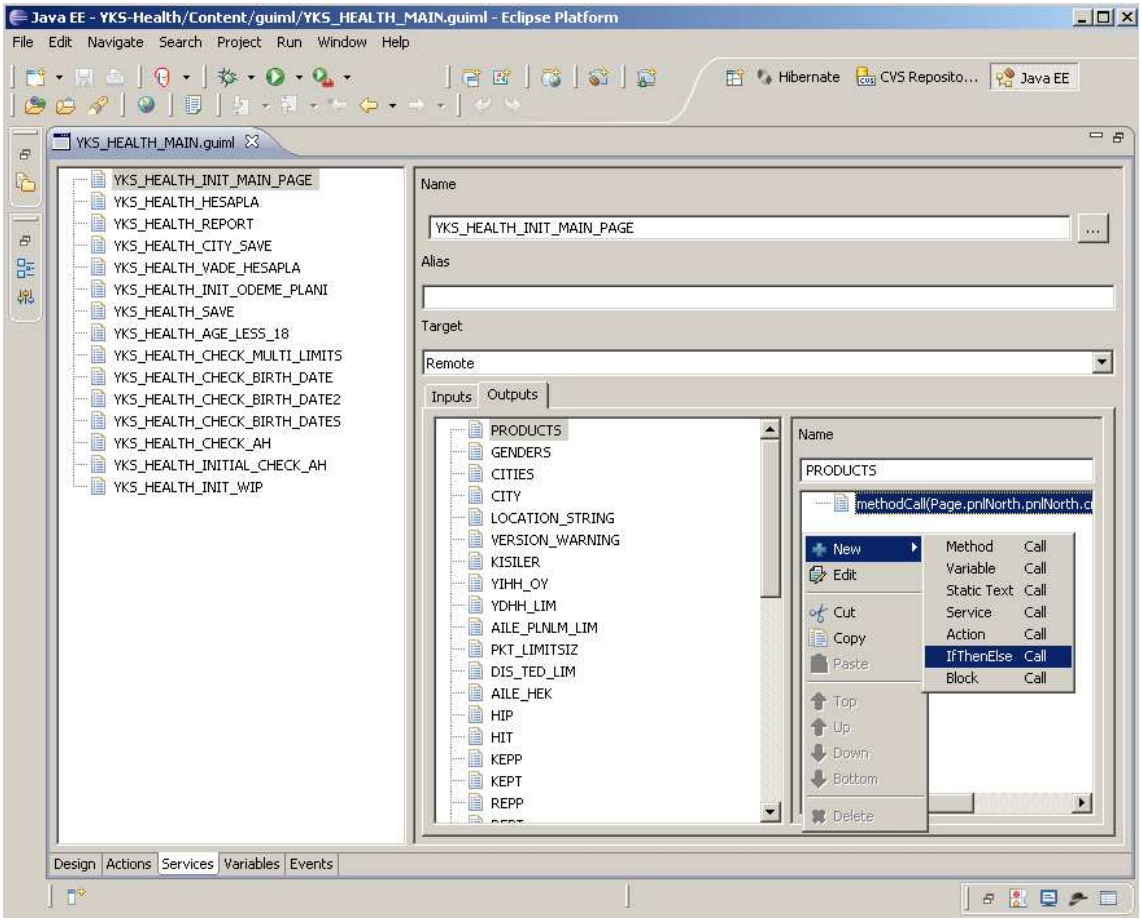

Figure 5.4 Services Tab to Show How a service is Implemented

Outputs can also be set to variables on a page. This way, variables are updated and hold on page and can be used when necessary. Another parameter can be static text, thus the input or the return value of a service is converted into a static text where users can insert whatever they want. Service return is also another option where a service can be called directly from an output. Then output of a service becomes the input of another service. Output parameters can be Actions, which are the functions and when an output parameter is set to an action, it will be a parameter passed to this action. If-then-else is used for a primitive logic during this output values, for example users can control this input and it required value, then call some action of call some another action etc. The last one is the block. Block is the set of code executed. Variables tab on eclipse plugin are for storing the variables on a page. Variable has the type of object, and can be all primitive type objects like String, Integer or Boolean. These values are casted and used on server side. Figure 5.5 shows a sample variables tab.

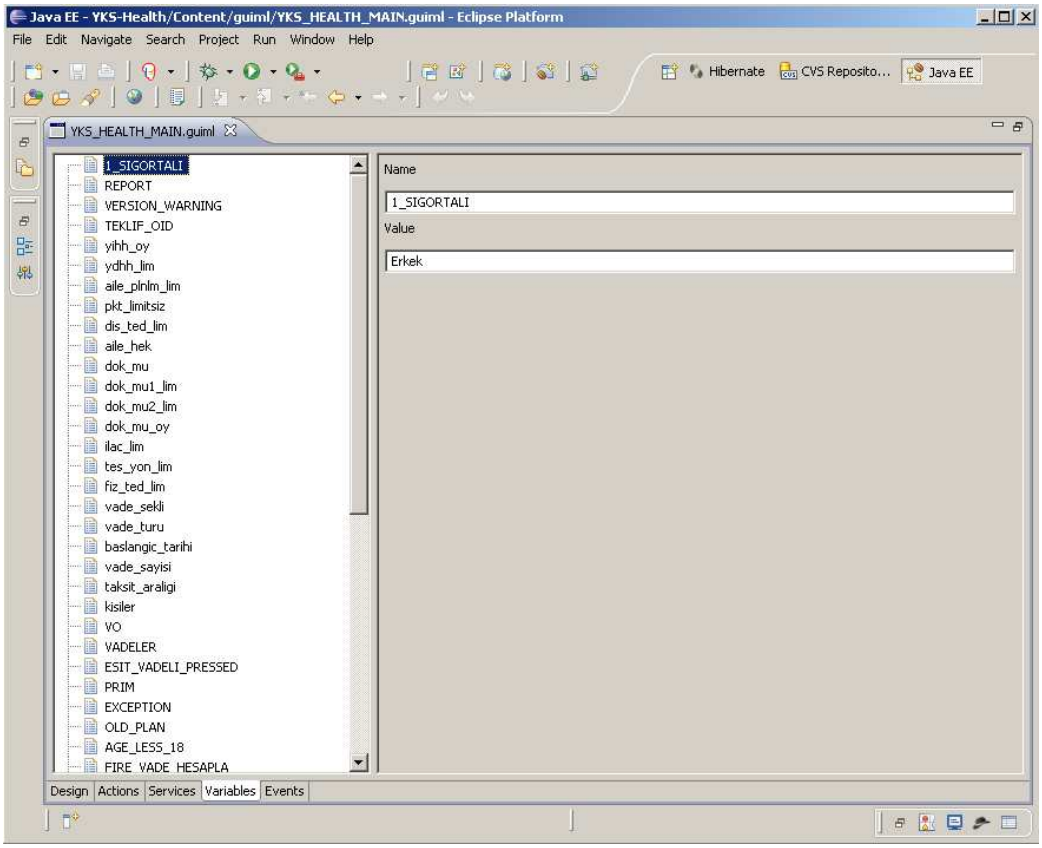

Figure 5.5 Variables tab to show the list of the variables

The events tab on the eclipse plugin shows the list of the events triggered by specific actions. Figure 5.6 has a screenshot of events tab. The lists of events are configured on bean xml files which are explained on chapter 4.4 Configuration files part. On design studio users right click on specific bean, select the events button and list of events triggered by that bean is listed, like table cell editing stopped or row inserted. Users can bind specific actions to the events when necessary.

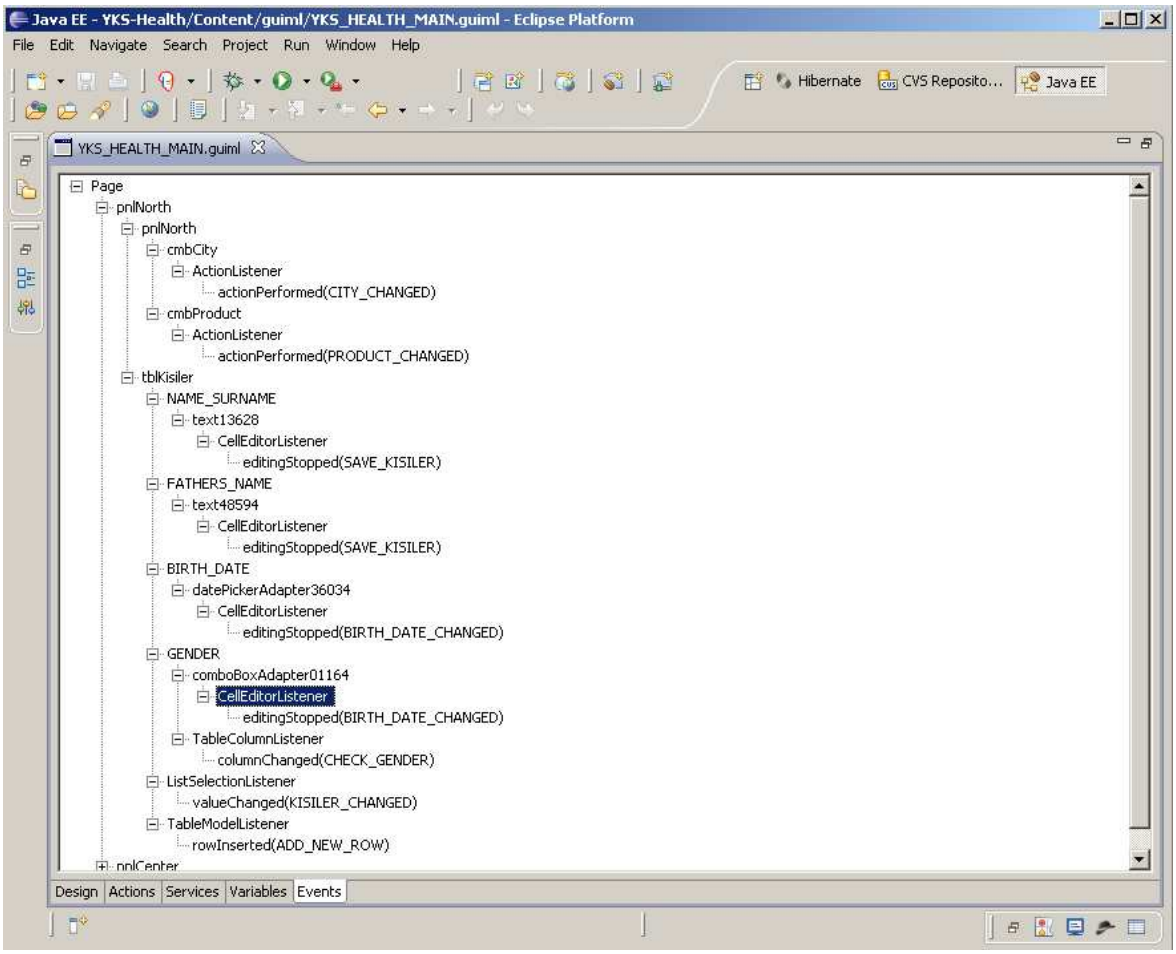

Figure 5.6 Events Tab to Show the List of Events on a Page

5.2 Sample Application Tutorial

Sample application tutorial shows a step by step demo of developing a “Hello World” type application communication with server side. Prior to this tutorial users shall install the application development framework on their computers. The tutorial begins with running Eclipse from desktop shortcut which the installer generated. We will use existing Graymound data source for this tutorial. Data source is the connection of database inside the application server.

First of all user should change eclipse perspective to Graymound. Eclipse plugin installed comes with its own perspective. First thing is creating a new Graymound project. This is done by the following steps below. Figure 5.7 shows eclipse page, after a project is created.

File -> New -> Other -> Graymound -> Graymound Project -> Next

Project Name: “Tutorial” -> Finish .

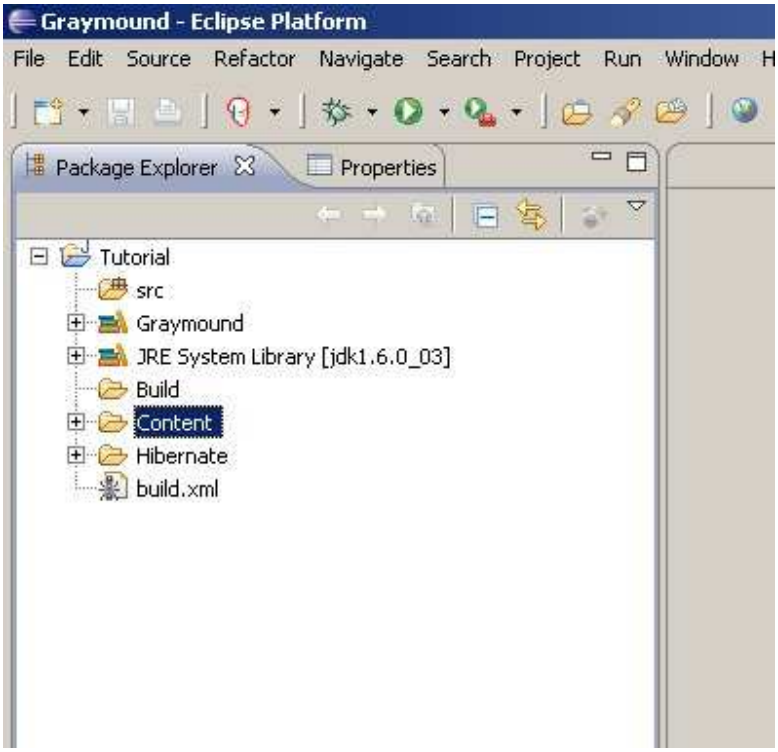

Figure 5.7 Creating a New Project

After creating a new project, some files automatically appear. These are the set of libraries or necessary folders for this specific project. Src holds the source files of the application. Graymound is the set of libraries on client side necessary during development or build. JRE system library is the eclipse generated java client library for client side. Build is the build folder where build script results are stored here. For example when build.xml is executed, the xml makes a jar file from the sources and also generates a hibernate archive for object relation mapping for the database side. These files are put under the Build folder. Content folder includes three sub folders. Guiml, reports and images. This folder is generated to give an order to the application files. Images are suggested to put under images file, reports are the jasper report database connected report files and guimls are the xml files the framework lies on. Next step is creating a new package. Users now should create a new package which is given on Figure 5.8. The name of the package is suggested to be tr.com.obss.tutorial.

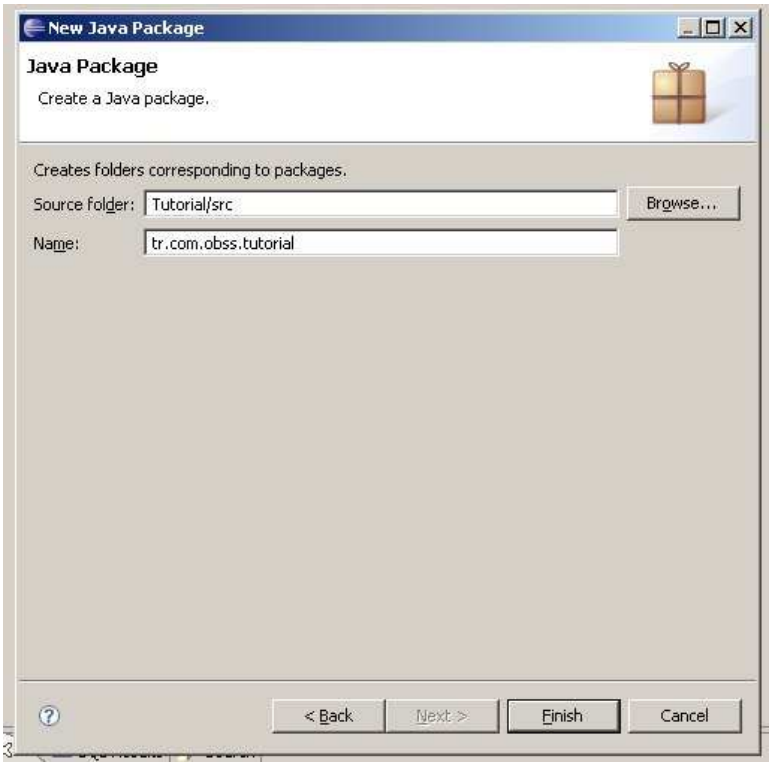

Figure 5.8 Creating a New Package

Now it is time to generate a new GUIML page for development. This is done by file-> new-> other-> guiml page. This page will be an empty framework page which is ready to be coded. This page opened in the GUIML editor is shown on Figure 5.9. The tutorial will have a combo box on the application. You can drag and drop a combo box from the palette to the application screen. After generation, you shall change its name to a readable one, like cmbLanguage. Next step is generating the service code for filling this combo. This code is given on Figure 5.10.

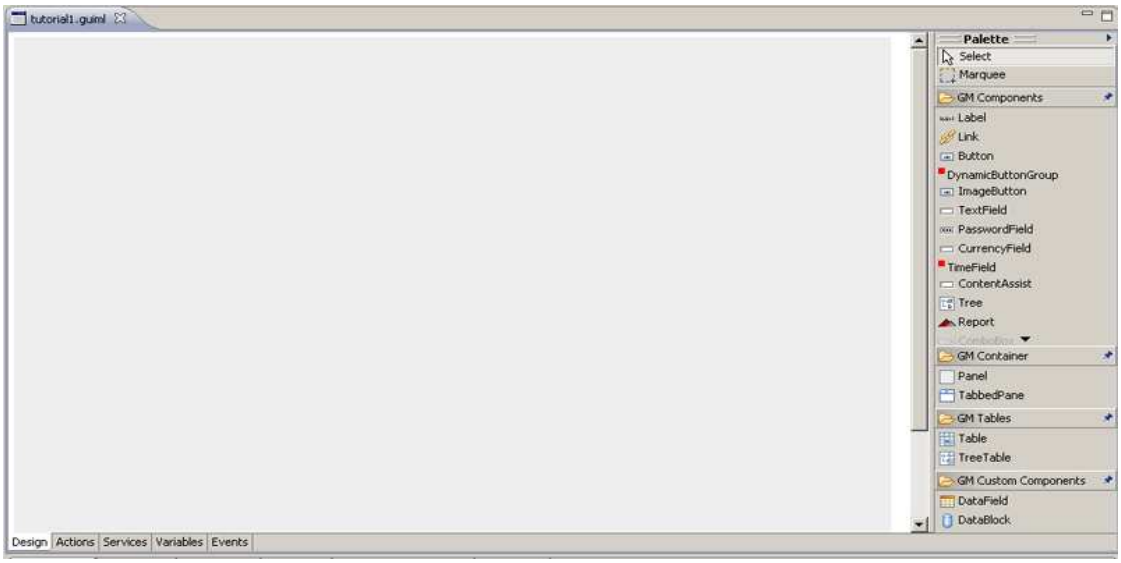

Figure 5.9 A New Page, After It is Created

```
@GraymoundService("FILLCOMBO")
public static GMMap fillCombo(GMMap iMap) {
    GMMap oMap = new GMMap();
    Connection conn = null;
    CallableStatement stmt = null;
    ResultSet rSet = null;
    String query = "";

    try {
        conn = GMServerDatasource.getConnection("java:/GraymoundDS");

        query = "select * from graymound.gm_adm_language_data";

        stmt = conn.prepareCall(query);
        rSet = stmt.executeQuery();

        int i = 0;

        while (rSet.next()) {
            oMap.put("LIST", i, "OID", rSet.getString("OID"));
            oMap.put("LIST", i, "NAME", rSet.getString("TEXT"));
            i++;
        }

        conn.close();
    } catch (Exception e) {

    }

    return oMap;
}
```

Figure 5.10 Fill Combo Service Code

After filling out the Combo code it is necessary to set the result of this code to the combo box. This is done by using the setModelData method of combo box. On Figure 5.11 setModelData method and how to select it on the combo box is shown.

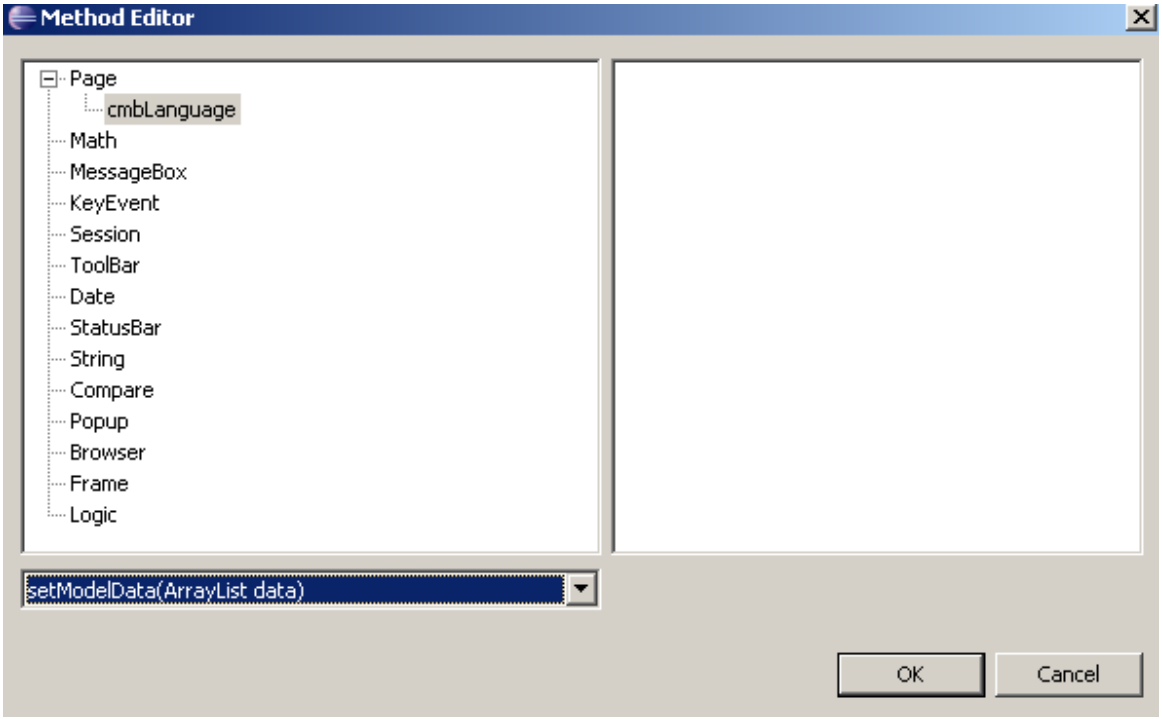

Figure 5.11 Combo Box Set Model Data

The last step of this application is putting a button on page, generating an action for combo and running the page to test the application. These last steps are illustrated on Figures 5.12, 5.13 and 5.14.

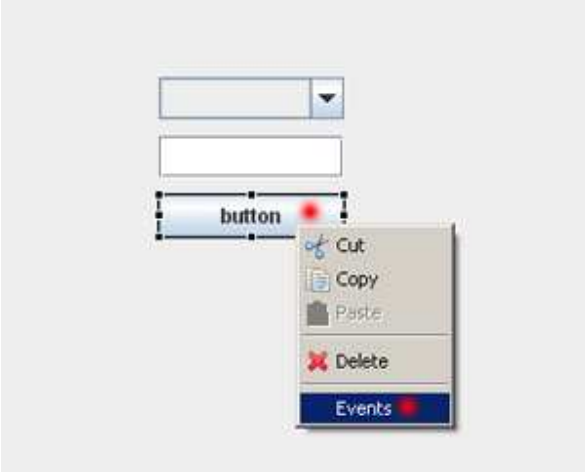

Figure 5.12 Button and Click Event

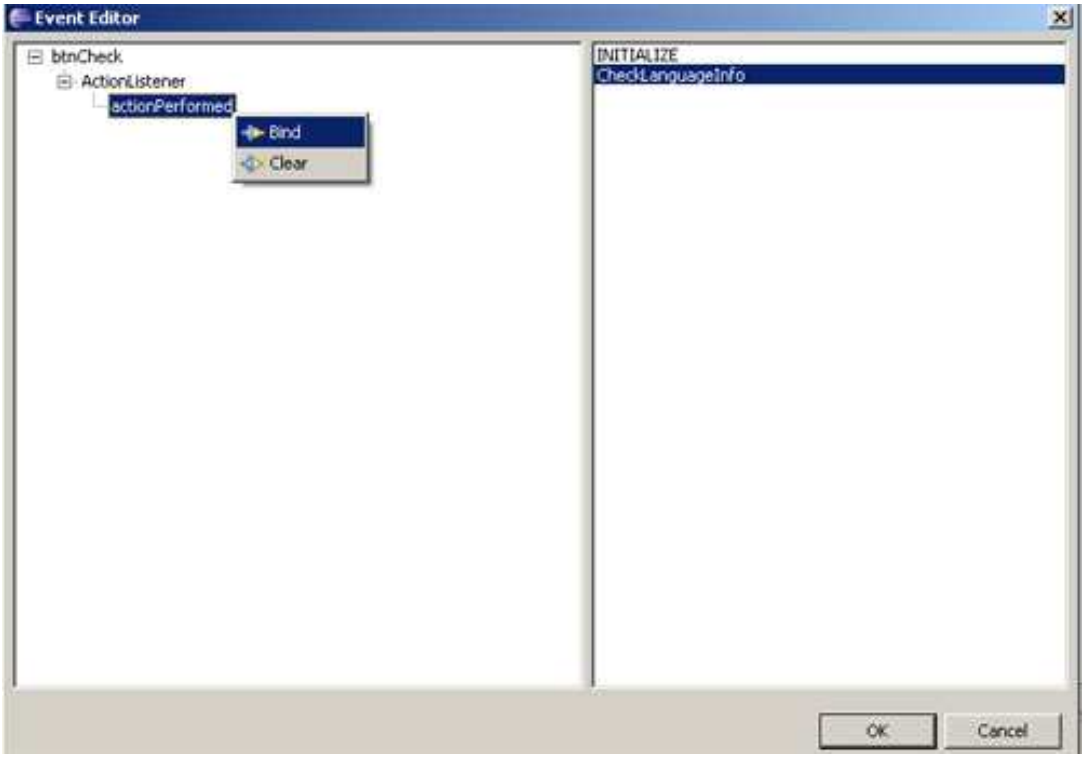

Figure 5.13 Binding the Event to the Action

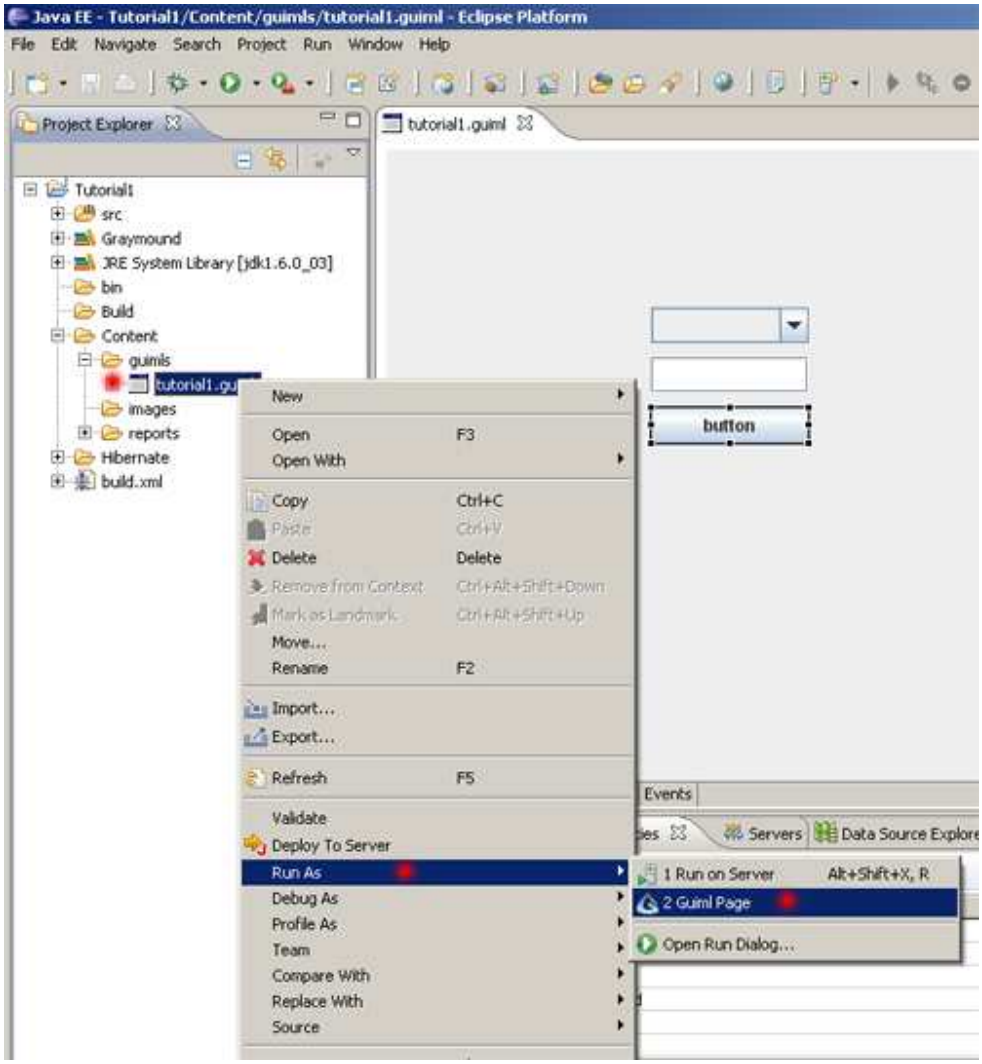

Figure 5.14 Running the Application for Test

Chapter 6
Conclusions and Recommendations for Future Work

The system has a high capacity of desktop like web application development. During the development phase system has also started in building an enterprise application, internet banking. The system completely fulfills the requirements of security, development and performance. Besides programmatic requirements, nowadays enterprises require management and monitoring capabilities over the application. The next step in development of the framework is, adding Business Process Management and Business Process Monitoring on top of this system.

The contributions of the project are unique xml mechanism such that it is able to generate a form application. Normally there are user interface standards to define on xml. These systems are successful in designing pages or even generating forms for settings of an application etc. However the xml of this system is dedicated for form applications which has extendable bean and layout capability, language independent system so that can be parsed by different languages, service calling and action, variable and event based system to generate a complete application. Another contribution of this system is AJAX mechanism. There are varieties of AJAX frameworks on the market. They are generally either for transforming swing code to AJAX or libraries for extendible use. However the AJAX mechanism of this framework has form generation capability and desktop application like structure which is unique in its category. The system uses an AJAX framework in the background and it acts as a high level language of Ajax based form generation.

Currently the system is used in an internet banking project of a Bank in Turkey and an Internet Banking project started in Albania. The system has high performance and capacity thus have a great potential getting used in Banks and large it projects. The next steps are Business Process Management inside this framework. A business process is a collection of related, structured activities that produce a service or product that meet the needs of a client. These processes are critical to any organization as they generate revenue and often represent a significant proportion of costs. Queuing the tasks and generating a process on top of those require standalone services. The BPM technology uses a SOA structure underneath and this framework has sufficient capabilities to integrate with BPM. This way it is possible to define processes inside the company using this framework. Another system required to integrate with this framework is Business Rules. A business rules engine is a software system that executes one or more business rules in a runtime production environment. Most Java-based rules engines provide a technical call-level interface, based on the application programming interface (API) standard, in order to allow for integration with different applications, and many rule engines allow for service-oriented integrations through Web-based standards such as WSDL and SOAP. Thus it will be possible to embed a rule engine to this system as one of the next steps.

References

[1] http://www.hibernate.org/, 2008

[2] http://qooxdoo.org/, 2008

[3] http://jasperreports.sourceforge.net, 2008

[4] http://www.oracle.com/technology/products/forms/index.html, 2008

[5] Erl, T., http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1044083,00.htm , 2005

[6] Curbera, F. and Duftler, M. J. C., *Lightweight middleware for service-oriented computing*., *IBM Research*.. http://www.research.ibm.com/journal/sj/444/curbera.html, 2005

[7] Potts, M., *Find Bind and Execute: Requirements forWeb Service Lookup and Discovery*, http://www.talkingblocks.com/resources.htm, 2005

[8] McGovern, J., *et al*., *Java Web Services Architecture.* s.l. : Sun Microsystems, 2003

[9] Lee, B., *IBM Research* http://www-128.ibm.com/developerworks/podcast/dwi/cm-int082206.txt, 2006

[10] Dobrzanski, J., *Social Semantic Information Sources for eLearning, 2007*

[11] http://delicious.com/, 2008

[12] http://www.flickr.com/, 2008

[13] http://www.wikipedia.org/, 2008

[14] http://www.last.fm/, 2008

[15] http://technorati.com, 2008

[16] http://en.wikipedia.org/wiki/JavaScript, 2008

[17] Sun Microsystems, *Enterprise JavaBeans Specification, Version 2.1*

http://java.sun.com/products/ejb, Nov. 12, 2003

[18] http://www.hibernate.org/118.html, 2008

[19] http://www.eclipse.org/, 2008

[20] http://www.softwareag.com/, 2008

[21] http://en.wikipedia.org/wiki/PL/SQL, 2008

[22] http://www.softwareag.com/Corporate/products/natural/rich_appl/default.asp, 2008

[23] http://java.sun.com/javaee/javaserverfaces/, 2008

[24] http://www.springframework.org/, 2008

[25] http://code.google.com/webtoolkit/, 2008

[26] http://www.oracle.com/technology/products/adf/index.html, 2008

[27] http://www-01.ibm.com/software/awdtools/developer/application/, 2008

[28] http://www.zkoss.org/, 2008

[29] http://www.icefaces.org/main/home/index.jsp, 2008

[30] http://www.acegisecurity.org/, 2008

[31] http://www.jgroups.org/, 2008

Appendix A. List of Configuration Files

```
- <beans>
- <!--
Util
  -->
  <bean name="Frame" infoPath="configuration/bean/util/GMFrameBeanInfo.xml" />
  <bean name="MainOutlook"
infoPath="configuration/bean/main/GMMainOutlookBeanInfo.xml" />
  <bean name="MainInternet"
infoPath="configuration/bean/main/GMMainInternetBeanInfo.xml" />
  <bean name="MainSimple"
infoPath="configuration/bean/main/GMMainSimpleBeanInfo.xml" />
  <bean name="Browser" infoPath="configuration/bean/util/GMBrowserBeanInfo.xml" />
  <bean name="Compare" infoPath="configuration/bean/util/GMCompareBeanInfo.xml" />
  <bean name="Date" infoPath="configuration/bean/util/GMDateBeanInfo.xml" />
  <bean name="KeyEvent" infoPath="configuration/bean/util/GMKeyEventBeanInfo.xml" />
  <bean name="Logic" infoPath="configuration/bean/util/GMLogicBeanInfo.xml" />
  <bean name="Math" infoPath="configuration/bean/util/GMMathBeanInfo.xml" />
  <bean name="MessageBox"
infoPath="configuration/bean/util/GMMessageBoxBeanInfo.xml" />
  <bean name="Popup" infoPath="configuration/bean/util/GMPopupBeanInfo.xml" />
  <bean name="Session" infoPath="configuration/bean/util/GMSessionBeanInfo.xml" />
  <bean name="StatusBar" infoPath="configuration/bean/util/GMStatusBarBeanInfo.xml"
/>
  <bean name="String" infoPath="configuration/bean/util/GMStringBeanInfo.xml" />
  <bean name="ToolBar" infoPath="configuration/bean/util/GMToolBarBeanInfo.xml" />
  <bean name="Flash" infoPath="configuration/bean/GMFlashBeanInfo.xml" />
- <!--
 Bean
  -->
  <bean name="Button" infoPath="configuration/bean/GMButtonBeanInfo.xml" />
  <bean name="DynamicButtonGroup"
infoPath="configuration/bean/GMDynamicButtonGroupBeanInfo.xml" />
  <bean name="CheckBox" infoPath="configuration/bean/GMCheckBoxBeanInfo.xml" />
  <bean name="ComboBox" infoPath="configuration/bean/GMComboBoxBeanInfo.xml" />
  <bean name="ContentAssist"
infoPath="configuration/bean/GMContentAssistBeanInfo.xml" />
  <bean name="CurrencyField"
infoPath="configuration/bean/GMCurrencyFieldBeanInfo.xml" />
  <bean name="DatePicker" infoPath="configuration/bean/GMDatePickerBeanInfo.xml" />
  <bean name="DynamicPart" infoPath="configuration/bean/GMDynamicPartBeanInfo.xml"
/>
  <bean name="File" infoPath="configuration/bean/GMFileBeanInfo.xml" />
  <bean name="ImageButton" infoPath="configuration/bean/GMImageButtonBeanInfo.xml"
/>
  <bean name="Label" infoPath="configuration/bean/GMLabelBeanInfo.xml" />
  <bean name="Link" infoPath="configuration/bean/GMLinkBeanInfo.xml" />
  <bean name="List" infoPath="configuration/bean/GMListBeanInfo.xml" />
  <bean name="MaskField" infoPath="configuration/bean/GMMaskFieldBeanInfo.xml" />
  <bean name="Page" infoPath="configuration/bean/GMPageBeanInfo.xml" />
```

```
  <bean name="Panel" infoPath="configuration/bean/GMPanelBeanInfo.xml" />
  <bean name="PasswordField"
infoPath="configuration/bean/GMPasswordFieldBeanInfo.xml" />
  <bean name="RadioButton" infoPath="configuration/bean/GMRadioButtonBeanInfo.xml"
/>
  <bean name="Report" infoPath="configuration/bean/GMReportBeanInfo.xml" />
  <bean name="TabbedPane" infoPath="configuration/bean/GMTabbedPaneBeanInfo.xml"
/>
  <bean name="Table" infoPath="configuration/bean/GMTableBeanInfo.xml" />
  <bean name="TableColumn" infoPath="configuration/bean/GMTableColumnBeanInfo.xml"
/>
  <bean name="TextArea" infoPath="configuration/bean/GMTextAreaBeanInfo.xml" />
  <bean name="TextField" infoPath="configuration/bean/GMTextFieldBeanInfo.xml" />
  <bean name="TimeField" infoPath="configuration/bean/GMTimeFieldBeanInfo.xml" />
  <bean name="CheckBoxTree"
infoPath="configuration/bean/GMCheckBoxTreeBeanInfo.xml" />
  <bean name="Tree" infoPath="configuration/bean/GMTreeBeanInfo.xml" />
  <bean name="TreeTable" infoPath="configuration/bean/GMTreeTableBeanInfo.xml" />
- <!--
Adapters
  -->
  <bean name="DataFieldAdapter"
infoPath="configuration/bean/adapter/GMDataFieldAdapterBeanInfo.xml" />
  <bean name="CheckBoxAdapter"
infoPath="configuration/bean/adapter/GMCheckBoxAdapterBeanInfo.xml" />
  <bean name="ComboBoxAdapter"
infoPath="configuration/bean/adapter/GMComboBoxAdapterBeanInfo.xml" />
  <bean name="ContentAssistAdapter"
infoPath="configuration/bean/adapter/GMContentAssistAdapterBeanInfo.xml" />
  <bean name="CurrencyFieldAdapter"
infoPath="configuration/bean/adapter/GMCurrencyFieldAdapterBeanInfo.xml" />
  <bean name="DatePickerAdapter"
infoPath="configuration/bean/adapter/GMDatePickerAdapterBeanInfo.xml" />
  <bean name="TextFieldAdapter"
infoPath="configuration/bean/adapter/GMTextFieldAdapterBeanInfo.xml" />
  <bean name="TimeFieldAdapter"
infoPath="configuration/bean/adapter/GMTimeFieldAdapterBeanInfo.xml" />
- <!--
 Custom Beans
  -->
  <bean name="DataField"
infoPath="configuration/bean/custom/GMDataFieldBeanInfo.xml" />
  <bean name="DataBlock"
infoPath="configuration/bean/custom/GMDataBlockBeanInfo.xml" />
  </beans>
```

Figure A.1 GMBeans Configuration File

```
<?xml version="1.0" encoding="UTF-8" ?>
- <configuration>
- <properties>
  <property name="size" value="1024,768" />
  <property name="resizable" value="true" />
  <property name="lookAndFeel" value="com.graymound.uiml.browser.util.GMLookAndFeel"
/>
  <property name="loginPage" value="GM_ADM_LOGIN.guiml" />
  <property name="firstPage" value="0" />
  <property name="debug" value="true" />
  </properties>
  </configuration>
```

Figure A.2 GMBrowser Configuration File

```
- <toolbar>
  <action name="New" actionName="NEW" icon="New.gif" />
  <action name="Save" actionName="SAVE" icon="Save.gif" />
  <action name="Delete" actionName="DELETE" icon="Delete.gif" />
  <action name="Search" actionName="SEARCH" icon="Search.png" />
  <action name="Clear" actionName="CLEAR" icon="Refresh.gif" />
  </toolbar>
```

Figure A.3 GMBrowserToolBar Configuration File

```
<?xml version="1.0" ?>
- <connections>
- <connection name="Default" url="http://localhost:8080/GMServer/Server/JAVA">
  <property name="authenticate-service" value="GM_ADM_USER_AUTHENTICATE" />
  <property name="language" value="tr" />
  <property name="username" value="Admin" />
  <property name="password" value="4e7afebcfbae000b22c7c85e5560f89a2a0280b4" />
  </connection>
  </connections>
```

Figure A.4 GMConnection Configuration File

```
- <layouts>
  <layout name="XYLayout" class="com.graymound.uiml.bean.layout.XYLayout"
constraintClass="java.awt.Rectangle" />
- <layout name="BorderLayout" class="java.awt.BorderLayout"
constraintClass="java.lang.String">
  <property name="vgap" displayName="Horizontal Gap" class="int" />
  <property name="hgap" displayName="Vertical Gap" class="int" />
  </layout>
  <layout name="FlowLayout" class="java.awt.FlowLayout" constraintClass="java.lang.String"
/>
- <layout name="GridLayout" class="java.awt.GridLayout" constraintClass="java.lang.String">
  <property name="vgap" displayName="Horizontal Gap" class="int" />
  <property name="hgap" displayName="Vertical Gap" class="int" />
  <property name="rows" displayName="Rows" class="int" />
  <property name="columns" displayName="Columns" class="int" />
  </layout>
  </layouts>
```

Figure A.5 GMLayouts Configuration File

```
- <listeners>
  <listener name="ChangeListener"
class="com.graymound.uiml.bean.listener.GuimlChangeListener" />
  <listener name="MouseListener"
class="com.graymound.uiml.bean.listener.GuimlMouseListener" />
  <listener name="ActionListener"
class="com.graymound.uiml.bean.listener.GuimlActionListener" />
  <listener name="FocusListener"
class="com.graymound.uiml.bean.listener.GuimlFocusListener" />
  <listener name="CellEditorListener"
class="com.graymound.uiml.bean.listener.GuimlCellEditorListener" />
  <listener name="ListSelectionListener"
class="com.graymound.uiml.bean.listener.GuimlListSelectionListener" />
  <listener name="TableColumnListener"
class="com.graymound.uiml.bean.listener.GuimlTableColumnListener" />
  <listener name="TableModelListener"
class="com.graymound.uiml.bean.listener.GuimlTableModelListener" />
  <listener name="TreeSelectionListener"
class="com.graymound.uiml.bean.listener.GuimlTreeSelectionListener" />
  <listener name="DocumentListener"
class="com.graymound.uiml.bean.listener.GMDocumentListener" />
  </listeners>
```

Figure A.6 GMListeners Configuration File

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <resource-factory>
- <resource name="Remote" class="com.graymound.resource.GMResourceRemote"
default="true">
  <property key="url" value="http://localhost:8080/GMServer/Server/JAVA" />
  </resource>
- <!--

        <resource name ="Local"
        class="com.graymound.resource.GMResourceLocal" default="true">
                  <property key="url"
        value="C:\GMDevelopment/Graymound/Server/Content/Root"/>
                  <property key="har-path"
        value="C:\GMDevelopment/Graymound/Server/Hibernate" system="true"/>
        </resource>
  -->
  </resource-factory>
```

Figure A.7 GMResourceFactory Configuration File

```
<?xml version="1.0" encoding="iso-8859-9" ?>
- <page>
- <i18n>
- <language name="tr">
<text key="Close" value="Kapat" />
<text key="Close All" value="Hepsini Kapat" />
<text key="Close Others" value="Diğerlerini Kapat" />
<text key="File" value="Dosya" />
<text key="Exit" value="Çıkış" />
<text key="Modules" value="Modüller" />
<text key="Save" value="Kaydet" />
<text key="OK" value="Tamam" />
<text key="Cancel" value="Vazgeç" />
<text key="Select All" value="Hepsini İşaretle" />
<text key="Deselect All" value="İşaretleri Kaldır" />
<text key="Hide All" value="Hepsini Gizle" />
<text key="Show All" value="Hepsini Göster" />
<text key="Username" value="Kullanıcı Adı" />
<text key="Password" value="Şifre" />
<text key="Language" value="Dil" />
<text key="Login" value="Giriş" />
<text key="Find" value="Bul" />
<text key="Shortcut" value="Kısa Yol" />
<text key="Error" value="Hata" />
<text key="INVALID_DATE" value="Geçersiz tarih girdiniz" />
</language>
</i18n>
- <i18n>
- <language name="en">
<text key="INVALID_DATE" value="Invalid Date" />
</language>
</i18n>
</page>
```

Figure A.8 GMToolkit Configuration File

Appendix B. CD Containing Code Listing and the Installation Package

CD will be provided within the thesis containing framework code.

Curriculum Vitae