MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

ENERGY EFFICIENCY REPORTS WEB APPLICATION

LICENSE THESIS

Graduate: **Adrian Lucian OROS**

Supervisor: **Senior lecturer Eng. Cosmina IVAN**

2011

MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

DEAN,
Prof. dr. ing. Sergiu NEDEVSCHI

VIZAT,
DEPARTMENT HEAD,
Prof. dr. ing. Rodica POTOLEA

Graduate: **Adrian Lucian OROS**

ENNERGY EFFICIENCY REPORTS WEB APPLICATION

1. **Project proposal:** *Create an easy to use, modern web application targeted to energy efficiency auditors whose main purpose is to calculate the energy efficiency rating of a building and to generate the energy efficiency certificate report based on the computed results.*

2. **Project contents:** *Table of contents*
   Chapters : Introduction, Project Objectives and specifications, Bibliographic researh, System Specifications and Architecture, Detailed design and implementations, User's manual, Conclusions
   Bibliography
   Annexis

3. **Place of documentation:** Technical University of Cluj-Napoca

4. **Consultants:** Eng. Marius Codarla, energy auditing expert

5. **Date of issue of the proposal:** November 1, 2010

6. **Date of delivery:** June 25, 2011

Graduate: _____

Supervisor: _____

MINISTERUL EDUCAŢIEI, CERCETĂRII, TINERETULUI ŞI SPORTULUI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

Declaraţie

Subsemnatul *Adrian Lucian OROS*, student al Facultăţii de Automatică şi Calculatoare, Universitatea Tehnică din Cluj-Napoca, declar că ideile, analiza, proiectarea, implementarea, rezultatele şi concluziile cuprinse în acest proiect de diplomă constituie efortul meu propriu, mai puţin acele elemente ce nu îmi aparţin, pe care le indic şi recunosc ca atare.

Declar de asemenea că, după ştiinţa mea, lucrarea în această formă este originală şi nu a mai fost niciodată prezentată sau depusă în alte locuri sau alte instituţii decât cele indicate în mod expres de mine.

Data: 25 iunie 2011

Absolvent: **Adrian Lucian OROS**
Număr matricol: 21010704
Semnătura:_____

Table of contents

Table of contents 4
Chapter 1 Introduction 6
1.1 Project context 6
1.2 Scope 6
Chapter 2 Project Objectives and Specifications 7
2.1 Problem domain 7
2.2 Project objectives 7
2.3 Motivation 7
2.4 Requirements 7
2.4.1 Functional Requirements 7
2.4.2 Non Functional requirements 9
Chapter 3 Bibliographic research 10
3.1 Similar systems 10
3.1.1 Certificate-energetic web application 10
3.1.2 AllEnergy Software 11
3.1 Http Protocol 12
3.2 Servlet Engines 12
3.3 Spring 13
3.3.1 Inversion of Control 13
3.3.2 Dependency Injection 15
3.3.3 AOP – Aspect Oriented programming 15
3.3.4 Spring MVC 16
3.3.4.1 Controllers 19
3.3.4.2 View resolvers 19
3.5 Apache POI 19
3.6 Jasper Reports 20
Chapter 4 System specifications and architecture 21
4.1 Energy efficiency report generator application 21
4.2 System block diagram 21
4.3 Use case Diagram 22
4.4 General system architecture 27
4.5 Detailed deployment diagram 31
Chapter 5 Detailed Design and Implementation 33
5.1 System structure 33
5.2 Web module 34
5.2.1 Configuring Spring MVC 35
5.2.2 Configuring Reports Generation 37
5.2.3 Web components 37
5.2.4 Question wizard Spring controller 37

5.2.5 Security 42
5.3 Model module 45
5.4 Business module 50
Chapter 7 User's manual 55
7.1 Installation description 55
7.2 User Manual 56
Chapter 8 Conclusions 60
8.1 Achievements 60
8.2 Future improvements 60

Chapter 1 Introduction

1.1 Project context

As of January 2007, when Romania has joined the EU, the law number 372/2005 relating the building energy efficiency has been put into practice which states that any building that is going to be sold, bought, rented needs to have an energy efficiency certificate.

The energy efficiency certificate is a technical document with an informative purpose, which shows the energy efficiency rating of a building by showing in a clearly, detailed manner the main characteristics of the building and its facilities as a result of an energetic and thermal expertise. The elaboration of the certificate is achieved by energy auditors for buildings, certified according to the romanian laws.

Energy Efficiency reports, called certificates, which valid for 10 years from the date of issue, are being elaborated for new or existing buildings that are constructed, sold or rented: single-family dwellings, blocks of flats, offices, educational buildings, hospitals, hotels and restaurants, gyms, trade services buildings, other types of buildings for energy. The certificate helps also to calculate future maintenance costs.

For the categories of dwellings that are rented or sold, the certificate shall be prepared by the owner and represents the contract of sale and purchase, that the lease contract. So every home owner, if he wants to sell or rent a home, he will need to have a building's energy performance certificate. The certificate is issued to the owner, the owner puts it, if necessary, the prospective buyer or tenant.

As a result of this new demand, and due to the huge market, approximately 8 million buildings need to be certified, the idea of having an automated system of generating this certificates has arrived.

As till now the law has not been applied, but as of 1 of January 2011 it is necessary to have such an energy certificate.

1.2 Scope

The purpose of this document is to define and describe the *Energy efficiency certificate reports application* by analyzing other similar systems available, the features that the new application comes with which differentiates it from the other ones, describing the technologies that will be used for the fulfillment of the project and then detailing the created application.

Chapter 2 Project Objectives and Specifications

2.1 Problem domain

The domain that this project covers is energy efficiency. Energy efficiency is measured by many factors and some of those are: the carbon dioxide emissions, annual consumption of energy, heating, warm water, air conditioning and lightning. Also, other factors that may decrease the energy efficiency are the so called penalty factors: which can be some of the following: penalties due to the exterior walls, due to the method the water is heated, buildings that don't have a independent heating system etc.

2.2 Project objectives

The purpose of this project is to provide a easy to use, modern web application that can generate **energy efficiency reports** based on calculations of the annual consumption of energy, water consumption, and lightning. The web application target are the accredited energy efficiency auditors whose main purpose is to calculate the energy efficiency of a building and generating the energy efficiency certificate report based on the calculated results.

The elements that will drive and define the application is to minimize the data being added by the auditors and provide an easy to use friendly UI that will lessen the total effort in calculating and creating the report. The main purpose of the application is nonetheless to be a solid, robust application that will calculate the data efficiently and that also will save users time by keeping record of their progress and validate data along completion not at the end when the calculations will be done. Also, another goal is to invest as little money as possible in acquiring hardware and software, and this is accomplished by the overall design of this application as only an internet connection is needed to run the program and generate the reports.

2.3 Motivation

The motivation for building an energy efficiency report web applications comes as from the need of simpler solutions for creating energy efficiency reports. It also lets auditors to focus on energy efficiency analysis rather than spending time on complicated formulas or by validating energy efficiency gathered data by themselves. There are other solutions that do the same thing, but the data gathering is different and does not bring additional values.

2.4 Requirements

After analyzing the project objectives the following functional and non-functional requirements specifications have been made:

2.4.1 Functional Requirements

The application must handle the following business operations:

a) **Must be accessible only by authenticated and authorized users and must not be available for other users**.

The developed application has many types of users: administrators, energy efficiency auditors and guests. Each type of users has a set of roles based on which they can view certain parts and perform certain actions. The administrator should be permitted all actions on the site.

By all actions it is meant that he can create reports, view other people reports and update their reports. An energy efficiency auditor can input data in the energy efficiency question wizard and generate reports based on the answers given. A guest can see the main page but does not have access to questions wizard, to the administration panel nor to the report generation component, thus has only read actions available.

b) Wizard-like input of technical parameters and question answers

The data and technical parameters needed for energy efficiency computation must have a wizard like flow and data should be validate as soon as it is available.

An authenticated energy efficiency auditor can input technical parameters by using a generic question wizard and thus not omit any data. The question wizard will need to take the questions from the database and display them in a generic way to the user; also the answers need to be taken from the database in case there are from a given range.

c) PDF Report generation

An authenticated energy efficiency auditor can generate a pdf report that will be the energy efficiency certificate. The energy certificate must have the format of the Romanian energy certificates and must have dynamic fields and images based on the computed results.

d) Correct energy efficiency computations

The formula used for the energy efficiency ratings must be as close as possible to the Romanian legislation from this domain. The formula must be able to be changed easily and the change should be transparent to the system, as in the system should not know about how the results are calculated, it only needs to collect the result.

e) The wizard contains relevant energy efficiency questions

The wizard must contain relevant questions according to the Romanian energy efficiency methodology. The wizard must also request as rare as possible data from the user, thus it should omit certain questions if those questions depend on other questions answers.

f) Generated reports and all answers should be stored in the database

The generated reports and the question answers must be saved in the database as the auditor can continue an unfinished work or they can update a given answer that was incorrect and generate a new energy certificate report.

g) Report and user administration

The application will also provide an administration component where users that have the administrator role can view the current reports and filter them based on certain criteria. Also, the administrators can perform create, read, delete, update operations on users. This web application section needs to be private and secured so that only administrators can access it.

h) Auditor multitasking

The energy efficiency auditors can work on many different reports at a time and the work is saved in the database as the auditors answer the wizard questions.

2.4.2 Non Functional requirements

a. Accessibility

The web application must be accessible from any location, with minimum effort and minimum technology knowledge. In order to do this, hints will be given to the energy eficiency auditor as they answer the wizard questions.

b. Security

The system must be very secure, as it will deal with important, sensible data regarding the auditors and if some other user will have access to the application work can be lost. For making sure that the security is very strong, container managed security should be used and also the passwords will be kept in the database using a strong hashing algorithm. Also, for ensuring that accounts are not stolen, the login page and register pages will work on a more secure protocol https.

c. Scalability

The system must be horizontal scalable because more questions can be added or update if the legislation changes. For assuring this, all the questions must be stored somewhere and the ystem takes the questions from that place. The system must also support many users at the same time and the must also support an increasing number lf wizard zuqestions. For this to happen, the qizard questions should be stored in such a way that multiple concurrency is supported, therefore a database persistence should be considered.

d. Availability 24/7

The system must be functional 24 hours per day, 7 days a week. In order to ensure thay both two zones described in the deployment diagram are always available, rendunant connection iwll be kept between the components ana also all the phisical devices will be connected to UPSs. Regarding the software part of it, the load balancer assures that the application is available even if a server drops by sending the requests just to the server that is still working.

e. Usability

The application can be learned very easily as the wizard displays hints for every questions, and also pictures that will display the object of inspection. An user can start create ane energy efficiency certificate report in less than 2-3 minutes and the actual steps of the wizard can be performed in less than 5 minutes if the auditors have done the thermic and energetic analysis and have the answers to the questions beforehand. Also, the wizard will have an intuitive user interface and a progress bar will display how many questions need to be answered.

f. Extensibility

The application will be extensble, as the system behavior must change at runtime without redeploying. This refers to the possibility to add new questions to the wizard and also updating the computing formula without having to restart the application.

g. Response time

The repsonse time of the aplication must be very low and the report generation must be very fast and thus the technology used for creating the pdf report must ensure that this is possible.

Chapter 3 Bibliographic research

As the developed project was a web application that had to generate reports based on the user inputs the following were studied:

- Http Protocol
- Servlet Engines
- Spring
- Jasper Reports
- Java Persistence API (JPA) and Eclipselink
- Apache POI

3.1 Similar systems

As energy efficiency report is a sought-of document, companies have developed their own system for generating energy efficiency reports.

The analyzed similar systems that were found and taken as a reference for the developed application are:

- Web application located at www.certificat-energetic.ro
- Software application described at www.allenergysystems.ro

3.1.1 Certificate-energetic web application

The web application located at certificate-energetic.com is similar to the proposed project as it is a web based application which is intended for energy efficiency auditors which can fill data in order to generate energy efficiency certificates. Their calculation program is accredited by the Romanian authorities therefore is a good reference application. The algorithm used for computing the energy efficiency is based on computing statistics data from the Romanian knowledge base and combining them with the inputted data which results in a high speed, efficient algorithm which enables the auditors to create energy certificates very fast. The calculation uses the latest techniques in computer science. From the technical point of view is similar to computer programs / PEC management used the world-renowned centers in Spain and the U.S., the same trend as in the Netherlands early in the migration to programming systems using XML-based programming language.

The program is a centralized computing, the introduction and running is done on dedicated servers solutions with good performance, co-located in Data Centers located outside the United States managed by reputable companies in the field.

Access to the user account is decentralized. The account can be accessed from any existing operating system today, using a simple browser available for free. The implemented system allows simultaneous access from multiple workstations for maximum flexibility.

The project elaboration can be supervised by another auditor and this is possible by assigning a given code to a given resource that can be given to another auditor.

Whether an auditor wants the help of colleagues to complete a complex project or one wants lower division (e.g. apartment certificates) to work in parallel, this system will allow the flexibility the auditors need.

This web application is done using php server side scripting deployed on a web server. The application is available for users as a public website and thus can be accessible by any user

having an internet connection. There are two types of users: guest users that can see the application description but with no functionality and users that have an account which also can be a student account or a paid account. The student account has access to all the features but has quantity restrictions (number of certificate constraints, square meters per building constraints) and the other type of account does not have any constraints and the application can be used any number of times for any type of building.

A nice feature that this application comes with is the security provided for the generated certificates. The certificates have 3 security keys that will protect and identify certificate changes.

The first key is the name of the program used to compute the final result which is written in the first page and in the second page the unique code of the certificate is written.

The second key is the fact that on each page of the certificate a key is printed which is a link to the online version of the document. In this way the certificate can be used online and if a user wants to see if a certificate is valid he can use the link and see the original version.

The third key is the fact that for each report a key is generated using a cryptography algorithm and stored in the database and can be used to identify changes from the original report.

3.1.2 AllEnergy Software

AllEnergy sooftware is a desktop application that calculates the annual energy consumptions according to the Romanian laws. The application is thus available as product that can be bought.

Using AllEnergy software application one can calculate the energy consumption profile of the building and identify energy saving measures.

AllEnergy ® Software V4.0.1, the latest version of this application contains many modules and some of them are:

- computing the annual consumption of heat for heating,;
- calculation of thermal resistance of construction elements of building envelop
- calculation temperatures unheated spaces (secondary area)
- calculation of the heat source for heating,
- calculation of auxiliary energy consumption for heating,
- determining the duration of the heating season graphical method / analytical,
- the annual consumption of heat for hot water consumption,
- annual consumption of electricity for lighting,
- calculation of electricity consumption for lighting customization for blocks of flats,
- annual energy consumption / cold air conditioning,
- specific annual energy consumption for mechanical ventilation,
- index of CO2 equivalent emissions, penalties awarded certified building,
- generating / printing of summary reports, interim and final results can be entered directly in the energy audit report,
- generating the final certificate report.

Thus the main purpose of this application is to provide solid calculations facilities and does not concentrate on user experience because the auditor must enter all the needed parameters and thus the system does not help their task.

The application is not available for free use. The update of the application also costs and also affects the user as the application is not self updating.
The user experience is not at a high level as he must input many parameters and moreover the design is not as good as it is not clear where everything fits in.

3.1 Http Protocol

The **Http Protocol** is a request response protocol. In http, the web browser acts as a client while the application running on a host on a computer acts as server. First, the client sends a request to the server, then the server, which stores content, or provides resources, such as html files, or performs other functions on behalf of the client, returns a response message to the client. A response contains completion status information about the request and may contain any date requested by the client in the message body.

The http protocol is platform independent but it generally runs on top of the TCP transport layer.

The request message is constructed of the request lines, headers, a blank line and the optional body. The request line is composed of three words: the first one is the method, the second is the resource link and the last word represents the http version.

The http protocol has 9 methods and the most important methods are **post** and **get**.

The GET method requests a representation of the specified resource. Requests using GET should not have the significance of taking an action other than retrieval. The W3C has published guidance principles on this distinction, saying, "web application design should be informed by the above principles, but also by the relevant limitations." [2].

POST Submits data to be processed to the identified resource. The data is included in the body of the request. This may result in the creation of a new resource or the updates of existing resources or both.

The http protocol is a stateless protocol and thus in order to keep user state a special object may be created on the server side for the user and it is called session. The session is kept using the additional headers. When the user first accesses the server a session is created for him and the session is transmitted to the user using a response header named set cookie along with an id which represents the session id. Then when the user accesses that resource again the session id will be transmitted in the cookie header by the browser and thus the state can be kept.

3.2 Servlet Engines

As the project is a web application and the technology used in implementing the application is java, there is a need to talk about the container that will host the application which in our case is a servlet engine. First of all, a servlet is an extension to the capabilities of server that are conformant to the request response type of communication. The servlet is not rooted to a specific protocol, but it is generally used over the http protocol. A servlet must apply the restrictions of the JAVA Enterprise Edition servlet API . The servlet has a specific structure that is used in the same way by all servlet engines. Among the servlet engines that can be used are, Apache Tomcat from Jakarta Apache, IBM's WebSphere or Google's app engine.

The servlet engine that is used in this web application is Apache Tomcat. The reason why is tomcat used is because it is open source and provides very good documentation regarding it's features and more it was built by many people from Sun Microsystems. Plus this servlet engine is the most popular one and hosts the majority of applications that need such a servlet container.

Apache tomcat has three main components: Catalina, Coyote and Jasper 2. Catalina is the servlet container which implements the Sun Microsystems servlet and jsp specifications. Coyote is the http connector which uses the http 1.1 protocol. The role of the connector is to listen to a port on a host and direct the request to the tomcat engine that will further process the request and send back data to requesting client. [2]

Jasper 2 is tomcat's jsp engine. It translates the jsp pages to servlets and compiles them as so. Jasper 2 has come with the following features that jasper did not have: custom tag pooling (the objects that were created for handling the tags can now be pooled and reused), background jsp compilation (the compilation of new jsp pages is done in the background and until the jsp is finished being compiled the old jsp is used until the first mentioned finishes when it will be replaced) and the jsp compiler is changed.

Tomcat also has security support integrated into it by using realms which represents a database of usernames, passwords and roles assigned to these users. There are various implementations of realm but those will be described later in the Realm security topic.

3.3 Spring

Spring is a free open source framework for the java technology.

The spring framework provides many modules that make available for developers the following services: inversion of control container, aspect oriented programming, transaction management and model view controller design.

3.3.1 Inversion of Control

Inversion of control is a design pattern that solves the problem of having loose coupling in an application. The inversion of control principle is explained in an article in the javaboutique website [4]. Loose allows one to change the implementations of two related objects without affecting the other object. Suppose one has a situation in which making a change to the implementation details in component A forces him to make a change in component B (which uses component A). This is what is called a strong coupling, and it can make life more difficult when it comes to the scalability of an application. Inversion of Control (IoC) is one approach can be used to achieve loose coupling between several interacting components in an application.

Inversion of control as the name says is all about inverting the control; or, how one object uses another object. Suppose class A wants to use class B. Traditionally, one would create and use an instance of class B within class A, like in the following example:

```
Class A {
  private B b;

  public A() {
      b = new B();
  }

  public void doSomething(){
     b.someMethod();
  }
```

One can easily spot the problems in creating dependency between two objects using this traditional approach will cause a tight coupling:

1. The creation of object B relies upon the availability of a default constructor.
2. Any change in the constructor implementation of class B will necessitate a change in the implementation of class A.
3. Suppose class A wants to use class C instead of class B. Then class A needs to change. What happens if class B and class C are two separate implementations of same service and the application will need to switch over from one implementation to the other?

All three of the problems in the code example are due the strong coupling of class A and class B. Class A first needs to know that it must use an instance of class B and then it needs to know how to construct an instance of class B.

The solution to this problem is to eliminate the dependency from class A. What follows is a modified class A, where strong coupling is removed:

```
Class A {
  private B b;

  public A(B b) {
      this.b = b;
  }

  public void doSomething(){
     b.someMethod();
  }
```

This example accepts an instance of class B via the constructor. This relieves class A from knowing how to instantiate class B. Now, in order to use class A, the caller class of A must also instantiate class B and pass it into class A. In this context, the caller class is also acting as an assembler of object A and object B. Object A does not explicitly look for B, but B is supplied to it. This is the Inversion of Control. The control of class B is taken out of class A and placed in the Assembler class.

The Inversion of Control container is the backbone of any spring based web application.

In spring the above inserted object that was given as a constructor parameter is called a bean.

The **`org.springframework.beans.factory.BeanFactory`** is the actual representation of the Spring IoC **container** that is responsible for containing and otherwise managing the aforementioned beans.

The *BeanFactory* interface is the central IoC container interface in Spring. Its responsibilities include instantiating or sourcing application objects, configuring such objects, and assembling the dependencies between these objects.

There are a number of implementations of the BeanFactory interface that come supplied straight out-of-the-box with Spring. The most commonly used BeanFactory implementation is the *XmlBeanFactory* class. This implementation allows you to express the objects that compose your application, and the doubtless rich interdependencies between such objects, in terms of XML. The XmlBeanFactory takes this XML **configuration metadata** and uses it to create a fully configured system or application.

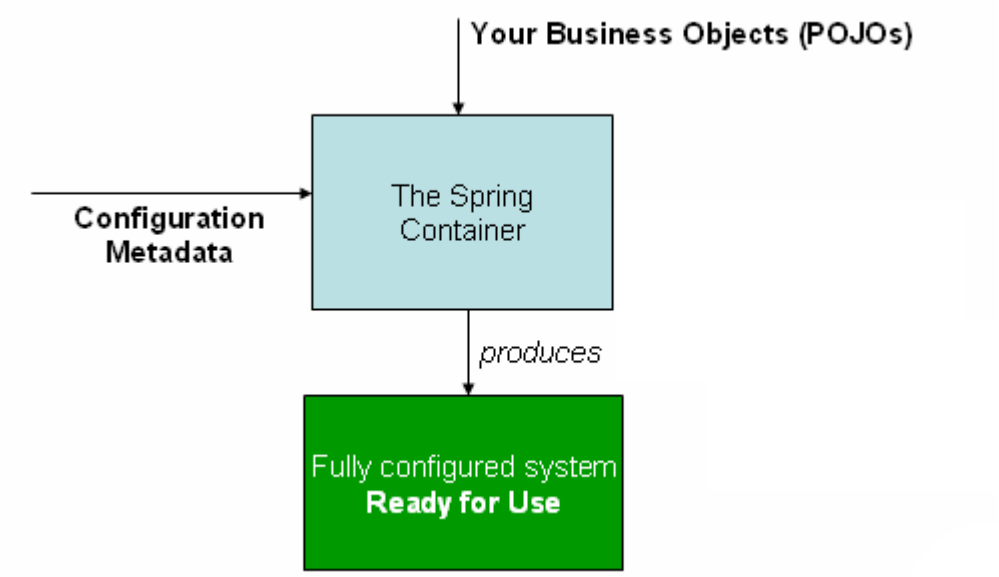

Figure 3.1 Spring Configuring based on xml written configuration metadata

So the beans can be defined in the form of xml. There is another way, much easier for developers of using annotation on classes and in this way the xml is not needed which is by far an easier solution.

Now that it is clear how to create beans we must now inject the dependencies to the objects that need them and this is done by the dependency **injection principle (DI).**

3.3.2 Dependency Injection

The basic principle behind *Dependency Injection* (DI) is that objects define their dependencies only through constructor arguments, arguments to a factory method, or properties which are set on the object instance after it has been constructed or returned from a factory method. Then, it is the job of the container to actually *inject* those dependencies when it creates the bean. This is fundamentally the inverse, hence the name *Inversion of Control* (IoC), of the bean itself being in control of instantiating or locating its dependencies on its own using direct construction of classes, or something like the *Service Locator* pattern.

For most applications the beans that are being injected are singletons, but can also have a prototype scope, request, session or global.

3.3.3 AOP – Aspect Oriented programming

Aspect-oriented programming, or AOP, complements object-oriented programming by allowing the developer to dynamically modify the static object-oriented model to create a system that can grow to meet new requirements, allowing an application to adopt new characteristics as it develops.

AOP provides a solution for abstracting cross-cutting code that spans object hierarchies without functional relevance to the code it spans. Instead of embedding cross-cutting code in classes, AOP allows you to abstract the cross-cutting code into a separate module (known as an aspect) and then apply the code dynamically where it is needed. You achieve dynamic application of the cross-cutting code by defining specific places (known as pointcuts) in your object model

where cross-cutting code should be applied. At runtime or compile time, depending on your AOP framework, cross-cutting code is injected at the specified pointcuts. Essentially, AOP allows you to introduce new functionality into objects without the objects' needing to have any knowledge of that introduction.

Spring AOP is implemented in pure Java. There is no need for a special compilation process. Spring AOP does not need to control the class loader hierarchy, and is thus suitable for use in a J2EE web container or application server.

Spring AOP currently supports only method execution join points (advising the execution of methods on Spring beans). Field interception is not implemented, although support for field interception could be added without breaking the core Spring AOP APIs. Spring AOP's approach to AOP differs from that of most other AOP frameworks. The aim is not to provide the most complete AOP implementation (although Spring AOP is quite capable); it is rather to provide a close integration between AOP implementation and Spring IoC to help solve common problems in enterprise applications.

3.3.4 Spring MVC

By far, the most important feature of Spring is the MVC framework which enables developer to create application using this architectural pattern more easily.

Before getting into Spring MVC's implementation particularities the MVC architectural pattern must be made clear. Model view controller architectural pattern is widely used because it isolates business logic (the application logic for the user) from the user interface (input and presentation), permitting independent development, testing and maintenance of each and thus having loose coupled components.

The MVC pattern as described at the Sun Microsystems site. [9]

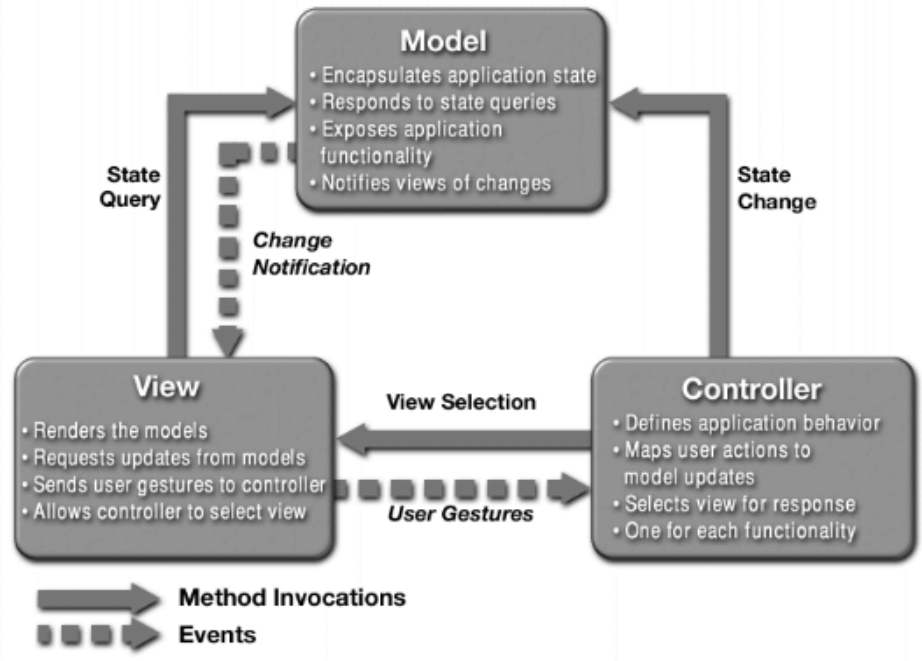

Figure 3.2 Model View Controller Architectural pattern diagram
http://java.sun.com/blueprints/patterns/images/mvc-structure-generic.gif

The model manages the data of the application domain and is mutable by responding to requests for information from the view and changing its state by requests coming generally from the controller.

The view renders the model in a form suitable for interaction.

The controller receives inputs from the view and sends the response by giving commands to the model and then making the model available to the view layer.

Spring's Web MVC framework is designed around a **DispatcherServlet** that dispatches requests to handlers, with configurable handler mappings, view resolution, locale and theme resolution as well as support for upload files. The default handler is a very simple Controller interface, just offering a ModelAndView handleRequest(request, response) method. This can already be used for application controllers, but you will prefer the included implementation hierarchy, consisting of, for example AbstractController, AbstractCommandController and SimpleFormController.

Spring's view resolution is extremely flexible. A **Controller** implementation can even write a view directly to the response (by returning null for the ModelAndView). In the normal case, a ModelAndView instance consists of a view name and a model Map, which contains bean names and corresponding objects (like a command or form, containing reference data). View name resolution is highly configurable, either via bean names, via a properties file, or via your own ViewResolver implementation. The fact that the model (the M in MVC) is based on the Map interface allows for the complete abstraction of the view technology. Any renderer can be integrated directly, whether JSP, Velocity, or any other rendering technology. The model Map is simply transformed into an appropriate format, such as JSP request attributes or a Velocity template model. [3]

The Spring MVC is illustrated in the diagram below. The dispatcher servlet is represented by the front controller.

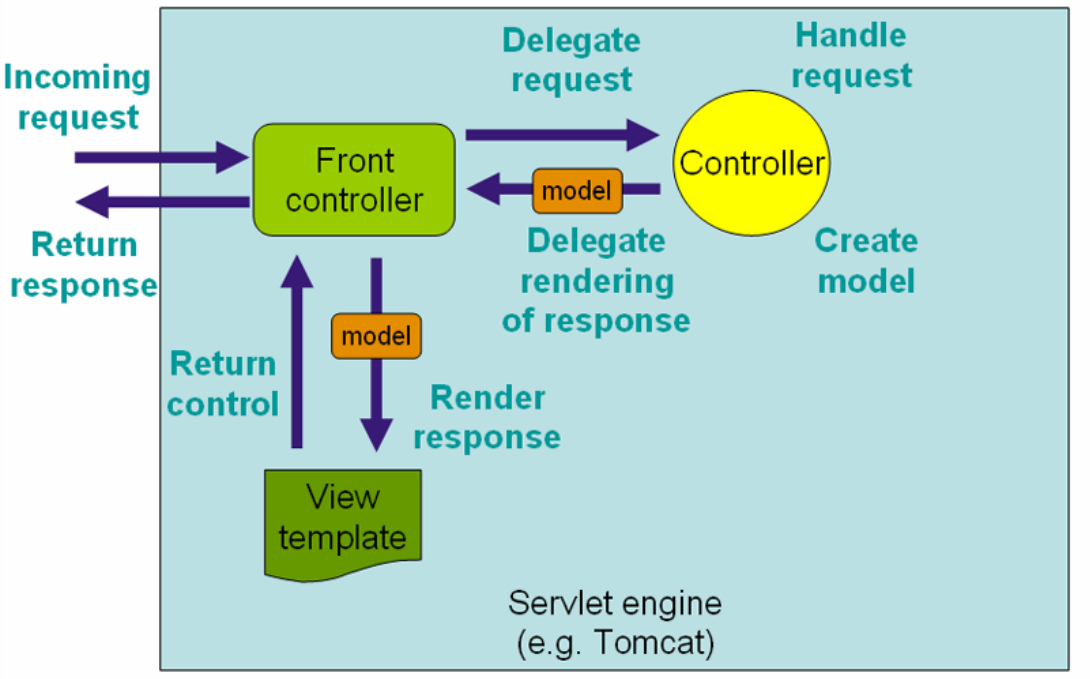

Figure 3.3 Spring MVC request handling mechanism high level view
http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html

The Dispatcher Servlet is actually a servlet and thus it needs to be mapped in the web xml and be accessible via a url which also needs to be given.

```
<web-app>
   <servlet>
      <servlet-name>spring</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

   <servlet-mapping>
      <servlet-name>spring</servlet-name>
      <url-pattern>*.do</url-pattern>
   </servlet-mapping>
</web-app>
```

In the above example all requested web pages that will end with .do will be handled by the spring servlet.

In a web MVC framework each Dispatcher servlet has an application context which inherits all the beans already defined in the root web application context.

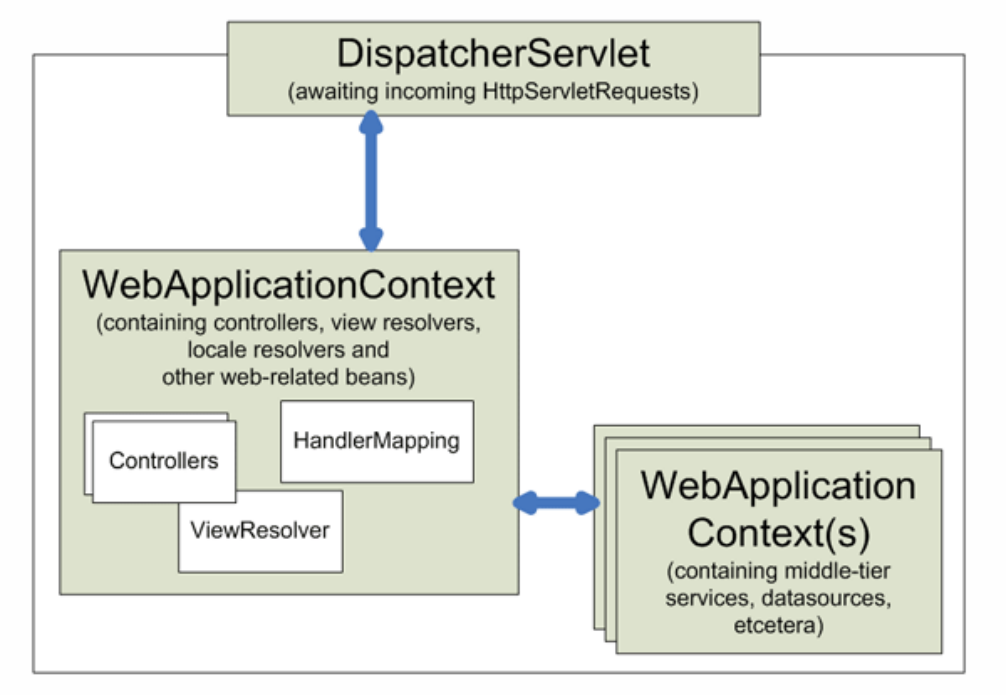

Figure 3.4 Spring context hierarchy
http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html

The most important components that build the application context are:

- Controllers which are the components that form the 'C' part of the MVC
- Handler mappings which handle the execution of a list of pre- and post-processors and controllers that will be executed if they match certain criteria (for instance a matching URL specified with the controller)
- View resolvers which are components capable of resolving view names to pages.

3.3.4.1 Controllers

Controllers take user input and convert such input into a sensible model which will be represented to the user by the view. Spring implements the notion of a controller in an abstract way enabling a wide variety of different kinds of controllers to be created. Spring contains form-specific controllers, command-based controllers, and controllers that execute wizard-style logic, to name but a few.

Spring's base contract for controllers is `org.springframework.web.servlet.mvc.Controller` interface, the source code for which is listed below.

```
public interface Controller {

    /**
     * Process the request and return a ModelAndView object which the DispatcherServlet
     * will render.
     */
    ModelAndView handleRequest(
        HttpServletRequest request,
        HttpServletResponse response) throws Exception;

}
```

3.3.4.2 View resolvers

View resolvers, as the names says, resolves the page that will be displayed after a certain action. Spring enables you to use JSPs, Velocity templates and XSLT views. The two interfaces which are important to the way Spring handles views are ViewResolver and View. The ViewResolver provides a mapping between view names and actual views. The View interface addresses the preparation of the request and hands the request over to one of the view technologies.

3.5 Apache POI

Apache POI (Poor Obfuscation Implementation**)** provides java libraries for reading and writing files in Microsoft Office formats, such as Word, PowerPoint and Excel. The name of this library reflects the

The part most important for the current project is the reading and writing of Excel files. HSSF is the POI Project's pure Java implementation of the Excel '97(-2002) file format.

Some of the features of HSSF are that it provides a way to read spreadsheets create, modify, read and write XLS, it provides an eventmodel api for efficient read-only access and it provides full user model api for modifying, reading and creating XLS files. [http://poi.apache.org/spreadsheet/index.html]

The API used is the event API. It's relatively simple to use, but requires a basic understanding of the parts of an Excel file. The advantage provided is that you can read an XLS with a relatively small memory footprint.

One important thing to note with the basic Event API is that it triggers events only for things actually stored within the file. With the XLS file format, it is quite common for things that have yet to be edited to simply not exist in the file. This means there may well be apparent "gaps" in the record stream, which you either need to work around.

3.6 Jasper Reports

Jasper reports is an open source java tool that provides an API for creating reports that can be written to many formats such as: PDF, html, Microsoft excel files, xml files and many others. [2][7]

JasperReports reports are defined in an XML file format, called JRXML, which can be hand-coded, generated, or designed using a tool. The file format is defined by a Document Type Definition (DTD), providing limited interoperability.

As a jasper report can become very complex, tools for creating and editing jasper reports file have been created to ease the development by having a better UI, and enabling developers to work more on the design of reports which is translated to xml automatically by the tool used.

One such tool, that was used in the creation of this application is iReport which an open source standalone graphical program that provides report designer capabilities, and is able to run reports using all data source supported by the JasperReports engine.

As described in JasperReports for Java Developers, by David R. Heffelfinger, Jasper is also integrated in Spring MVC which has a custom view resolver which will enable showing a compiled jasper report as a spring controller result view. The Spring Framework contains five different `View` implementations for JasperReports, four of which correspond to one of the four output formats supported by JasperReports, and one that allows for the format to be determined at runtime:

- JasperReportsCsvView for csv files
- JasperReportsHtmlView for html files
- JasperReportsPdfView for pdf files and
- JasperReportsXlsView for Microsoft Excel documents.

JasperReports has two distinct types of report file: the design file, which has a .jrxml extension, and the compiled report file, which has a .jasper extension. With the Spring Framework one can map either of these files to a report file and the framework will take care of compiling the .jrxml file on the fly. After a .jrxml file is compiled by the Spring Framework, the compiled report is cached for the lifetime of the application. To make changes to the file one thus needs to restart your application.

Chapter 4 System specifications and architecture

4.1 Energy efficiency report generator application

An energy efficiency report is a technical document with an informative purpose that which atests the energy efficiency rating of a building, that declares and computes the energy efficiency rating presented in a concise, formal manner by detailing the main characteristics of the building as result of the thermic and energetic analysis.

The scope of the application is to give specialized energy efficiency auditors to input data by answering a set of questions through a wizard based on the thermic and energetic analysis and generate the energy efficiency reports. The application must be a web application and the full fuctionalities must only be available to auditors. The web application will contain quite a number of web pages with the most important and fundamental pages housing the set number of questions that the energetic auditor is required to fill in.

The auditors should be able to use the application without installing a software because the application requires only an internet connection for the auditors to access it.

4.2 System block diagram

The web system application is composed of two main parts:

- the demilitarized zone and
- secured intranet zone which is protected by an internal firewall.

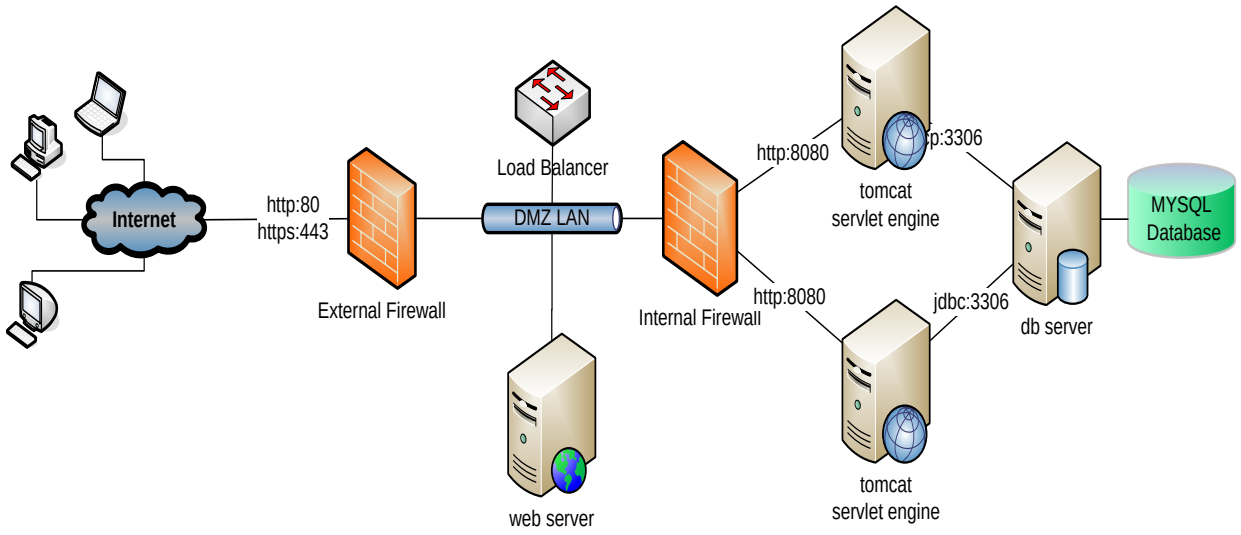

Figure 4.1 System block diagram

DMZ

The demilitarized zone is a middle ground between an organization's trusted internal network and an untrusted, external network such as the Internet. Also called a "perimeter network," the DMZ is a subnetwork (subnet) that may sit between firewalls or off one leg of a

firewall. DMZ is firstly a military term that refers to the area between two enemies therefore it has the same meaning in the nowadays web context.

In this application the demilitarized zone is not left unprotected but has a small shield, an external firewall that accepts only requests for a given protocol, in our case because it is a web appliation, the http and https protocols will be enabled thus two ports will be opened 80 for http and 443 for https respectively. As the internal firewall is needed when having a DMZ it can be pointed out that the DMZ has a dual **firewall architecture**. This architecture is more secure then the other popular alternative, single firewall architecture. The first firewall assures more security as it only allows requests fro the DMZ only. The second firewall (also called "back-end" firewall) allows only traffic from the DMZ to the internal network.

A firewall can be a dedicated hardware or an software application that can block certain requests for an IP area or ports. Such an application can be an apache web server that can be configured to block certain IPs and certain ports.

The demilitarized network will contain a set of Apache Web Servers that are used as proxy and load balancers. The load balancing functionality can also be achieved using a hardware load balancer. Load balancers assures that the application will be available even if a server that hosts the application drops down and thus gives a higher availability to the application.

The intranet

Intranet users will be able to access the Certificate Report Wizard web site servers directly using specified ports. The intranet LAN will provide access to the energy efficiency reports web site tomcat servlet engines and the database server.

The two servlet engines get requests from the load balancer and process the request.

4.3 Use case Diagram

Based on the above requirements, three types of users have been identified: administrators, energy efficiency auditors and guest users. Every type of users will have associated roles based on which their actions are authorized. The authentication part is done using their email address wich will be used as username and their password.

After authentication certain sections of the web application will be configured to be accessed only by users with specific roles. This part is knwon as the authorization phase.

The user roles specific authorized actions are:

a. Administrator:

- Adding, editing and locking auditors
- Adding / editing/ removing guests users
- Viewing and filtering all reports based on certain filters
- Answering wizards questions
- Generating reports
- Downloading reports
- Browse read only content (pages)

It can be seen in figure 4.2 that all the actions depend on the login action. This is happening because no part of the application is available if the administrator is logged in, this being due to the security constraints.

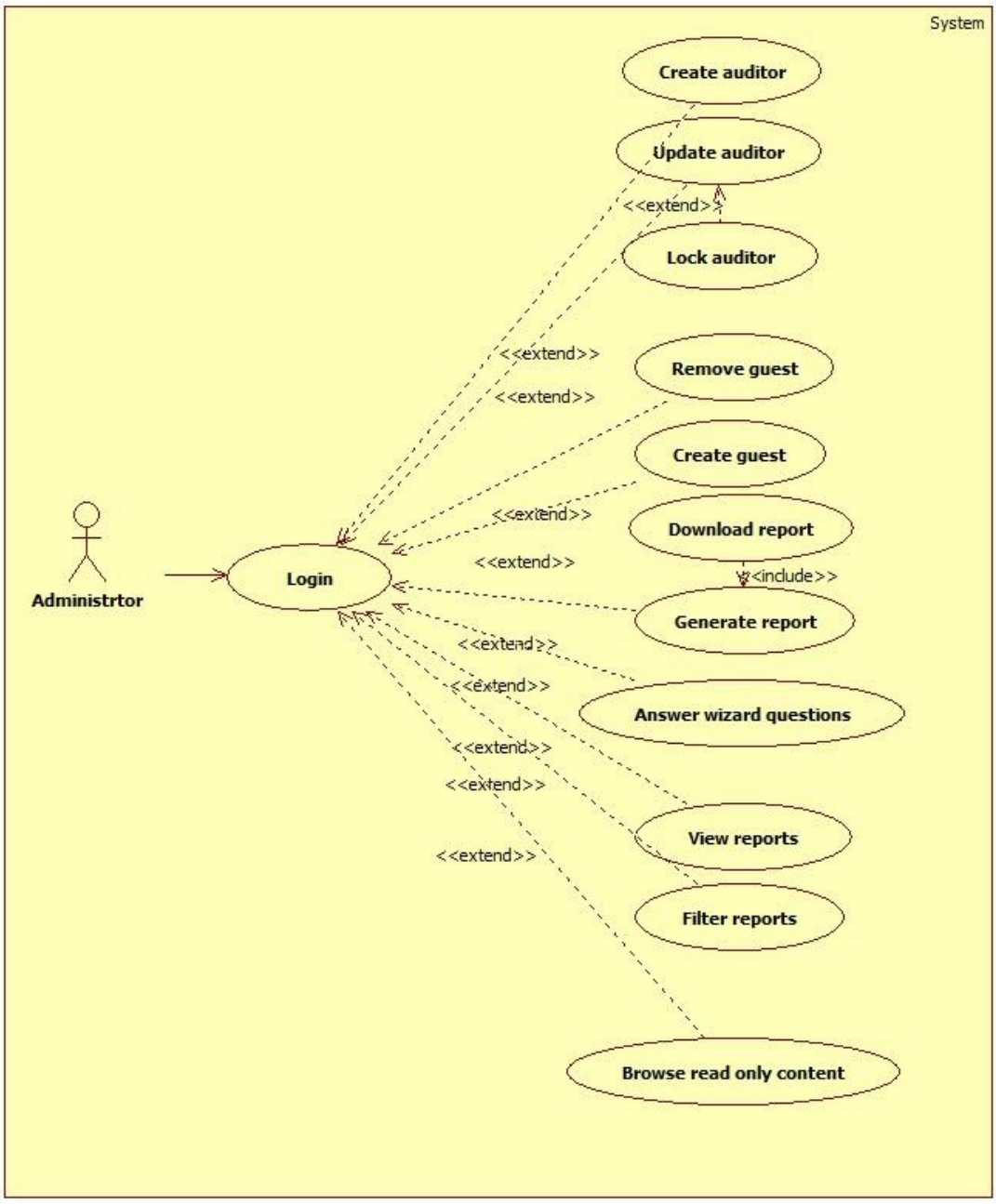

Figure 4.2 Administrator Role Use case diagram

b. Energy efficiency auditors

An energy efficiency auditor rights are a subset of the administrator rights. The auditor does not have access to administration specific actions such as viewing all reports and administrating web application users.

Available use cases:

- Viewing his reports
- Updating his reports

- Answering wizards questions
- Generating reports
- Downloading reports
- Browse read only content (pages)

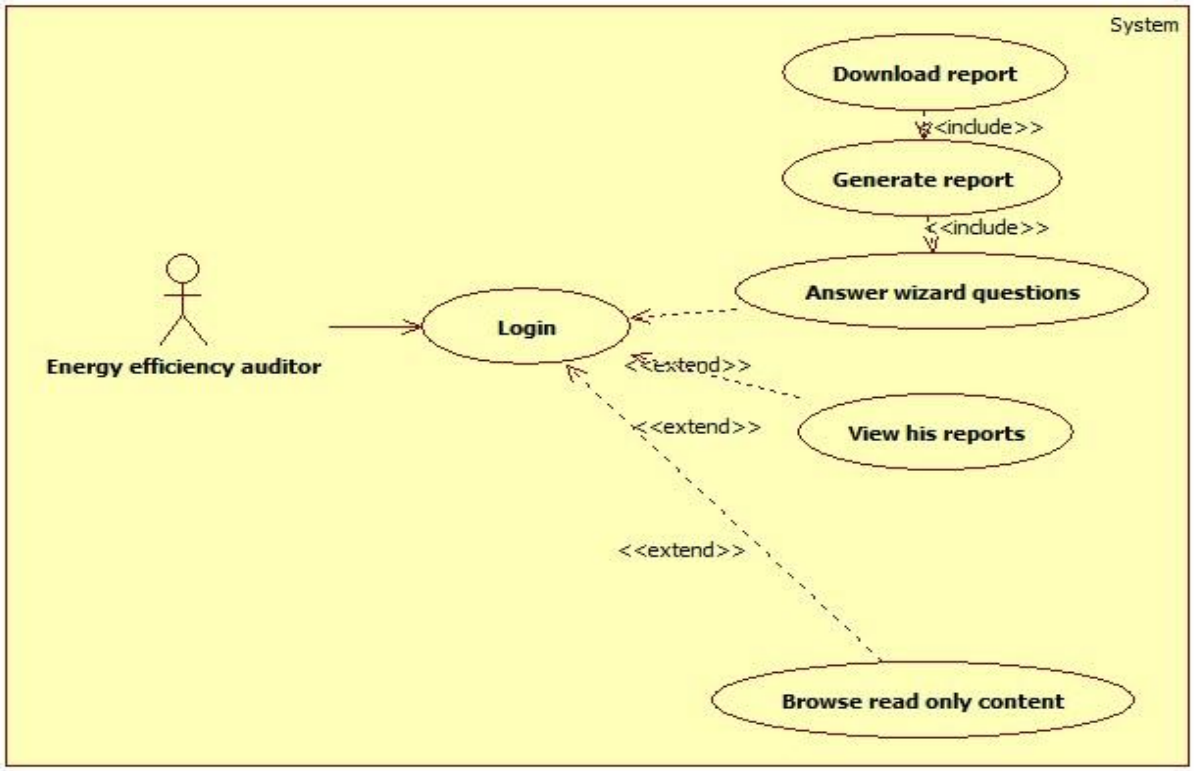

Figure 4.3. Energy efficiency auditor use case diagram

c. Guest

- Log in
- Browse read only content

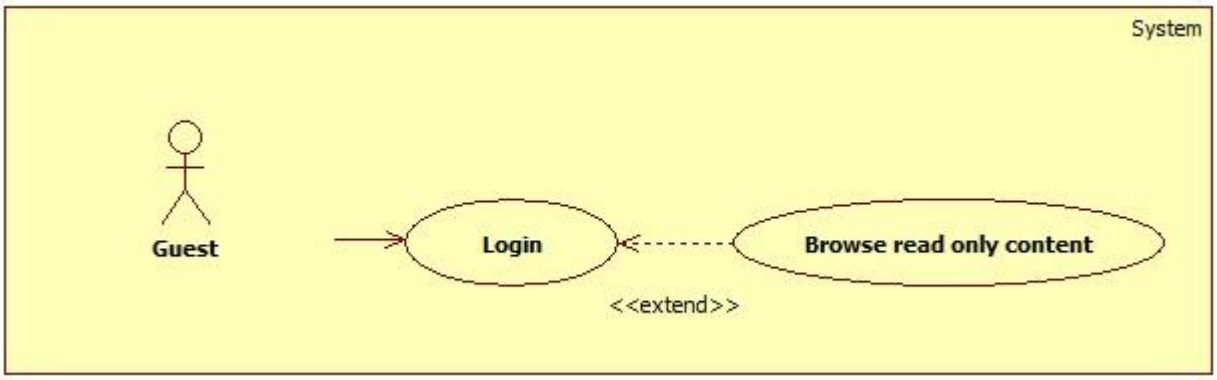

Figure 4.4 Guest use case diagram

The previous specifications are derived use cases that define the basic scenarios of the system. The main purpose of creating the use cases is to identify the main application and not to

define all possible combinations of events in the system. Diagram shown in Figure 4.2 is the main use case for an energy efficiency auditor, Figure 4.1 represents the main use case diagram for an auditor and in Figure 4.3 is the use case diagram for a user with full rights. For the proposed scheme were identified following use cases:

Use case number.1: Login

Description: User authentication based on his email and password. During this login also the roles of the user will be fetched, based on which other future use cases depend.
Primary actors: Administrator, Energy efficiency auditor, guest
Preconditions: the web application must be up and running and a web browser must be opened at any page from the energy efficiency certificate report web site.
Postconditions: The user is authenticated and based on his roles other features will be put available or disabled.
Success scenario:

1. The authentication page is displayed and the user email and password are requested.
2. The system verifies the given data and forwards the user to the first requested page.

Extensions:
*a. From technical reasons (such as not working internet connection) the browser cannot open the requested page

1. The "Page not found" error page will be displayed

*2a. The authentication credentials are invalid.

1. The configured, not authenticated error page is displayed.

Use case number.2: Answer wizard questions

Description: User answers the energy efficiency wizard questions that will be used as data for computing the final energy efficiency rating of the building
Primary actors: Administrator, Energy efficiency auditor
Preconditions: the web application must be up and running and a web browser must be opened at any page from the energy efficiency certificate report web site. Also the user must be logged in must have the role of an administrator or of an auditor. The user must go to the start wizard page.
Postconditions: The user answered the questions and the next question is .
Success scenario:

1. The authentication page is displayed and the user email and password are requested.
2. The system verifies the given data and forwards the user to start wizard page.
3. The user answers the questions by filling in a text or choosig from o multiple choice list
4. The next question is displayed/ The generate report link is displayed if all the wizard questions heve been answered

Extensions:
*a. From technical reasons (such as not working internet connection) the browser cannot open the requested page

1. The "Page not found" error page will be displayed

*2a. The authentication credentials are invalid.

1. The configured, not authenticated error page is displayed.

*2b. The authentication credentials are correct but the user does not have any of the needed roles, because he is a guest user

1. The configured, not authorized page is displayed showing that the user does not have the right to view that page.

*3a. The user does not fill in the requested answer

1. The same page is displayed once again, showing that the user must respond to the wizard question

Use case number 3: Generate Report

Description: The users generates the energy efficiency certificate report

Primary actors: Administrator, Energy efficiency auditor

Preconditions: the web application must be up and running and a web browser must be opened at any page from the energy efficiency certificate report web site. Also the user must be logged in must have the role of an administrator or of an auditor.

Postconditions: The energy certificate report is displayed in the browser.

Success scenario:

1. The authentication page is displayed and the user email and password are requested.
2. The system verifies the given data and forwards the user to start wizard page.
3. The user answers all the wizard questions by filling in a text or choosing from a multiple choice list
4. The generate report link is displayed after all the questions have been aswered.
5. The energy efficiency certificate report is displayed in the browser

Extensions:

*a. From technical reasons (such as not working internet connection) the browser cannot open the requested page

1. The "Page not found" error page will be displayed

*2a. The authentication credentials are invalid.

1. The configured, not authenticated error page is displayed.

*2b. The authentication credentials are correct but the user does not have any of the needed roles, because he is a guest user

1. The configured, not authorized page is displayed showing that the user does not have the right to view that page.

*5a. The PDF report is not displayed because the user does not have adobe reader installed

1. The user must install the adobe reader in order to see the report.

Use case number 4: Create energy efficiency auditor

Description: The administrator creates a new energy efficiency auditor

Primary actors: Administrator

Preconditions: the web application must be up and running and a web browser must be opened

Postconditions: A new energy efficiency auditor will be added to the system

Success scenario:

1. The authentication page is displayed and the user email and password are requested.
2. The system verifies the given data.

3. The user goes to the administration panel
4. The user chooses to create a new energy efficiency auditor.
5. The user inputs the new auditor data and submits the data
6. A message that a new auditor was created is dispplayed

Extensions:

*a. From technical reasons (such as not working internet connection) the browser cannot open the requested page

1. The "Page not found" error page will be displayed

*2a. The authentication credentials are invalid.

1. The configured, not authenticated error page is displayed.

*3a. The authentication credentials are correct but the user does not have the administrator role

1. The configured, not authorized page is displayed showing that the user does not have the right to view that page.

*6a. The new auditor cannot be created because an auditor with the same email already exists

1. The page containing the form for the given user is displayed once again showing the above mentioned error message.

Use case number 5: Browse read only website content

Description: On this application there are some pages that do not have any functionality, such as feedback, contact and description of the application.

Primary actors: Administrator, guest user, energy efficiency auditor

Preconditions: the web application must be up and running and a web browser must be opened

Postconditions:

Success scenario:

1. The authentication page is displayed and the user email and password are requested.
2. The system verifies the given data.
3. The user browses the read only website content

Extensions:

*a. From technical reasons (such as not working internet connection) the browser cannot open the requested page

1. The "Page not found" error page will be displayed

*2a. The authentication credentials are invalid.

1. The configured, not authenticated error page is displayed.

4.4 General system architecture

At the beginning of the development process, when the system architecture was thought of two architectures emerged: the three-tier web application architecture and the model-view-controller (MVC) architecture.

The three-tier web application architecture (Figure 4.4) is composed of three layers: presentation layer, business layer and data layer. There is another part of this architecture, which is the cross-cutting layer which is meant to hold the aspect oriented driven part of the architecture.

Presentation layer

This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing, and shopping cart contents. It communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network.

Business layer (business logic, logic tier, data access tier, or middle tier)

The logic tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing.

Data layer

This tier consists of database servers. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance.

In the energy efficiency certificate report wizard the custom implementation of this architecure would have been:

1. Client layer: any browser that has support for javascript.
2. Presentation Layer: the presentation layer consists of jsp pages and servlets, along with cascading style sheets and other resources such as images, javascript files. The jsp pages would be used only for displaying data and servlets would be used only for accessing the business layer that will perform the actual business logic. Also, the presentation layer would be the entry point of end user requests which will fill in forms or will request certain pages.
3. Business layer: the business layer would contain services, managers that will perform custom logic based on the requests that come from the presentation layer. The business layer will perform the business logic on the model data which will be taken by accessing components from the data layer.
4. The data layer will contain repositories, data access objects that will contain methods/functionalities for accessing the underlying database using queries or other ORMs (object relational mapping).

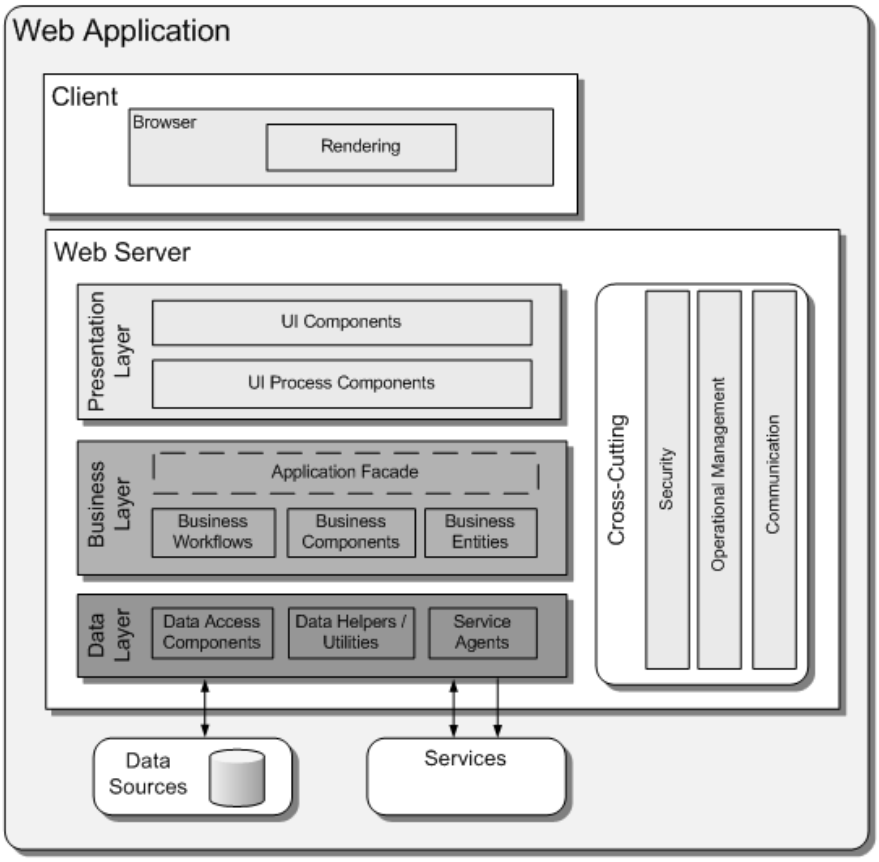

Figure 4.5 Layerd web application architecture diagram

The MVC Architecture

The **model** manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller). In event-driven systems, the model notifies observers (usually views) when the information changes so that they can react.

The **view** renders the model into a form suitable for interaction, typically a user interface element. Multiple views can exist for a single model for different purposes. A viewport typically has a one to one correspondence with a display surface and knows how to render to it.

The **controller** receives user input and initiates a response by making calls on model objects. A controller accepts input from the user and instructs the model and viewport to perform actions based on that input.

An **MVC** application may be a collection of model/view/controller triads, each responsible for a different UI element.

The MVC architecture (Figure 4.4) was decided to be used because there are various frameworks that apply to this pattern that bring greater development speed and greater overall robustness of the system. Some frameworks that apply this MVC architectural pattern is Spring Framework, Stripes Framework or Tapestry.

The framework that will be used for this project is Spring. The reason why Spring is chosen over the others is that it is also an inversion of control container and also because it is integrated with Jasper reports and thus reports can be generated in a clear, elegant manner.

The detail about how Spring MVC framework which is a **JAVA** based framework will be used for creating the energy efficiency report wizard application will be covered later in the detailed system architecture.

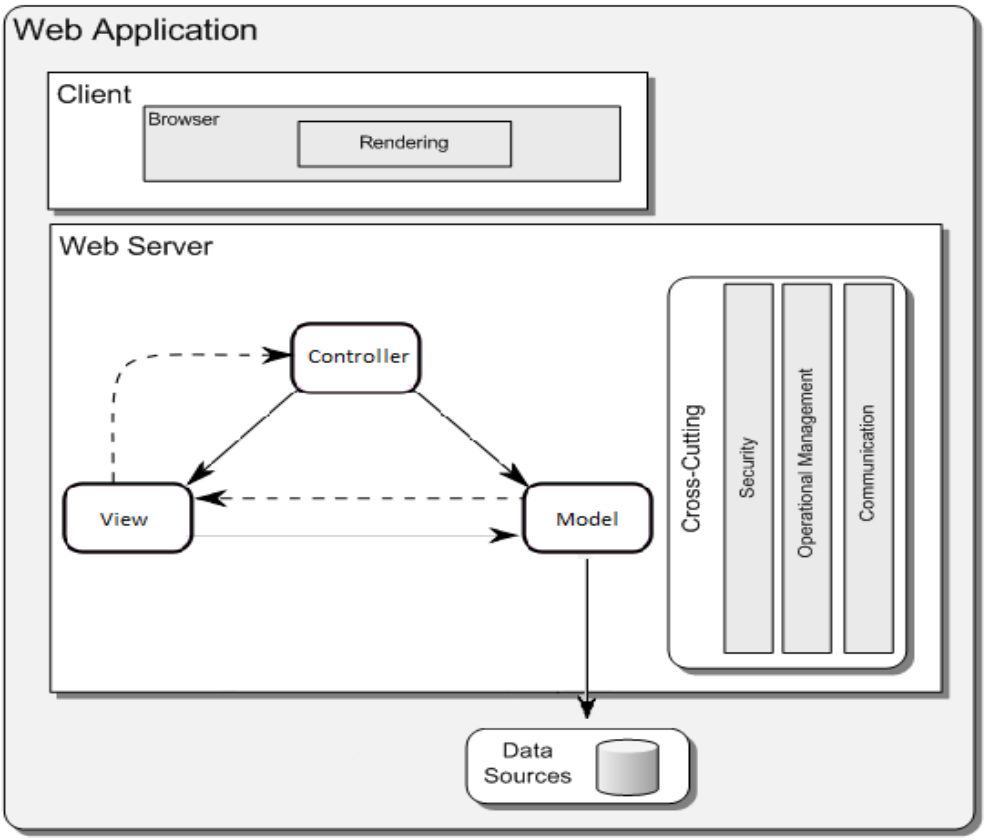

Figure 4.6 Model-view-controller web application architecture diagram.

In the energy efficiency reports web application the cross cutting component consists only of security.

The security of a web application's resources can be controlled either by the container or by the web application itself. As stated in the Tomcat, Definitive Guide book from O'Reilly [5], the J2EE specification calls the former container managed security and the latter application-managed security. Tomcat provides several approaches for handling security through built-in mechanisms, which represents container-managed security. On the other hand, if one has a series of servlets and JSPs with their own login mechanism, this would be considered application-managed security. In both types of security, users and passwords are managed in groupings called realms. In order to use Tomcat's container-managed security, you must set up a realm. A realm is simply a collection of users, passwords, and roles. Web applications can declare which resources are accessible by which groups of users in their web.xml deployment descriptor.

Then, a Tomcat administrator can configure Tomcat to retrieve user, password, and role information using one or more of the realm implementations.
Tomcat comes in with more realm implementations which are UserDatabaseRealm, JDBCRealm, JNDIRealm, and JAASRealm. Choosing which realm should be used, insert a Realm element

into your server.xml file, specify the realm to use through the className attribute, and then provide configuration information to the realm through that implementation's custom attributes:

```
<Realm className="some.realm.implementation.className"
customAttribute1="some custom value"
customAttribute2="some other custom value"/>
```

As stated in **Pro Apache Tomcat 6 book** [6] the UserDatabaseRealm is loaded into memory from a static file and kept in memory until Tomcat is shut down. In fact, the representation of the users, passwords, and roles that Tomcat uses lives only in memory; in other words, the permissions file is read only once, at startup. This is a real disadvantage because if users and roles need to be added the whole application needs to be restarted, so this type of realm is not very lucrative for serious real work production.

The **JDBCRealm** provides more flexibility than a UserDatabaseRealm, as well as dynamic access to data. It is essentially a realm backed by a relational database; users, passwords, and roles are stored in that database, and JDBCRealm accesses them as often as needed. If your existing administrative software adds an account to a relational database table, for example, the JDBCRealm will be able to access it immediately. The JDBC realm is configured by giving the JDBC driver class, the connection URL, connection user and password and also specifying the user and role table names and the name of the column that are needed: the username column, the password column and the role name column.

Container-managed authentication methods control how a user's credentials are verified when a protected resource is accessed. There are four types of container-managed security that Tomcat supports, and each obtains credentials in a different way: basic authentication when user's password is required via HTTP authentication as base64-encoded text, Digest authentication - when the user's password is requested via HTTP authentication as a digest-encoded string, Form authentication - when the user's password is requested on a web page form and client-cert authentication when the user is verified by a client-side digital certificate.

The authentication method that will be used in this project is the form based authentication method, which is going to be described in more detail.

To implement form-based authentication, one needs a login form page and an authentication failure error page in your web application, a security-constraint element and a login-config element in the **web.xml** file just like in the example below.

```
<login-config>
<auth-method>FORM</auth-method>
<realm-name>Real Example</realm-name>
<form-login-config>
<form-login-page>/login.html</form-login-page>
<form-error-page>/error.html</form-error-page>
</form-login-config>
</login-config>
```

4.5 Detailed deployment diagram

As the general system diagram presents there are two servlet engines (two Tomcats) that will both have a developed web application. This type of topology is called a farm topology.

The energy certificate report wizard application will be assembled as a web archive that will be deployed on each tomcat instance. The web application will be based upon the Spring MVC framework.

As it can be seen in Figure 4.5 there is an apache web server in front of those two tomcats which will have the role of a load balancer.

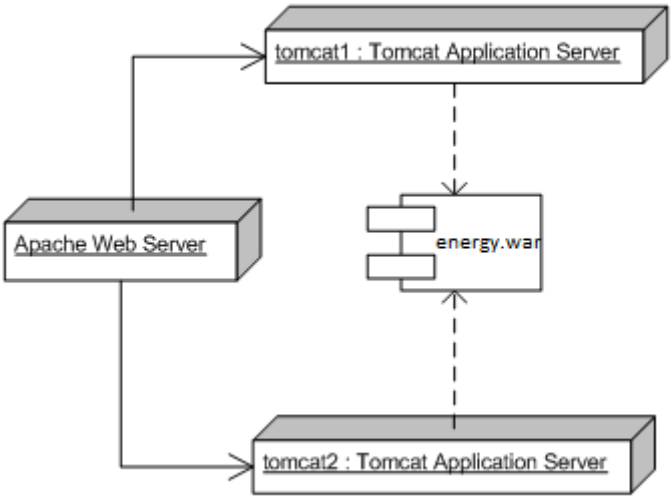

Figure 4.5 Two tomcats in a web farm topology hosting the energy web archive

Chapter 5 Detailed Design and Implementation

For implementing the energy certificate report wizard, a programming language that fits the necessities of the application was used. The language must be a cross operating system one, which is fully object oriented and also open source. The language that has all this necessities is JAVA.

For data persistence, the MySql Server was used as it is open source and it is a owned by Oracle (which also owns JAVA). As a tool for creating the database tables, the MySql Workbench was used.

5.1 System structure

5.1.1 Detailed system block diagram

Based on the Spring implementation of the MVC architectural pattern and of the used servlet container a new more detailed can be architectural diagram can be described (Figure 5.1).

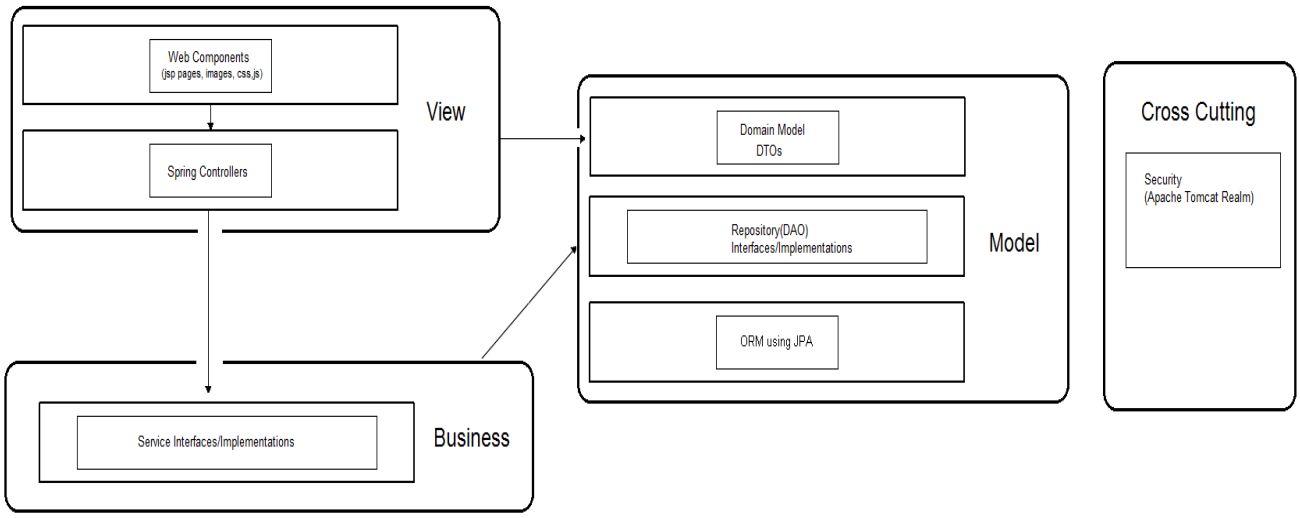

Figure 5.1 Detailed System Block Diagram

Based on the above detailed application architecture, the application has been divided into four modules:

- Web (represents the View) - the web module will contain the web components: jsp pages, static html files and other resources; and the spring controllers. The role of the spring controllers will be detailed later.
- Business - will contain java classes that will hold the application business logic.
- Model (part of Model) - module will contain database transfer objects (DTOs) represented as JPA entities
- Repository (part of Model) - module will contain java classes that will have the role of database access object (DAOs).

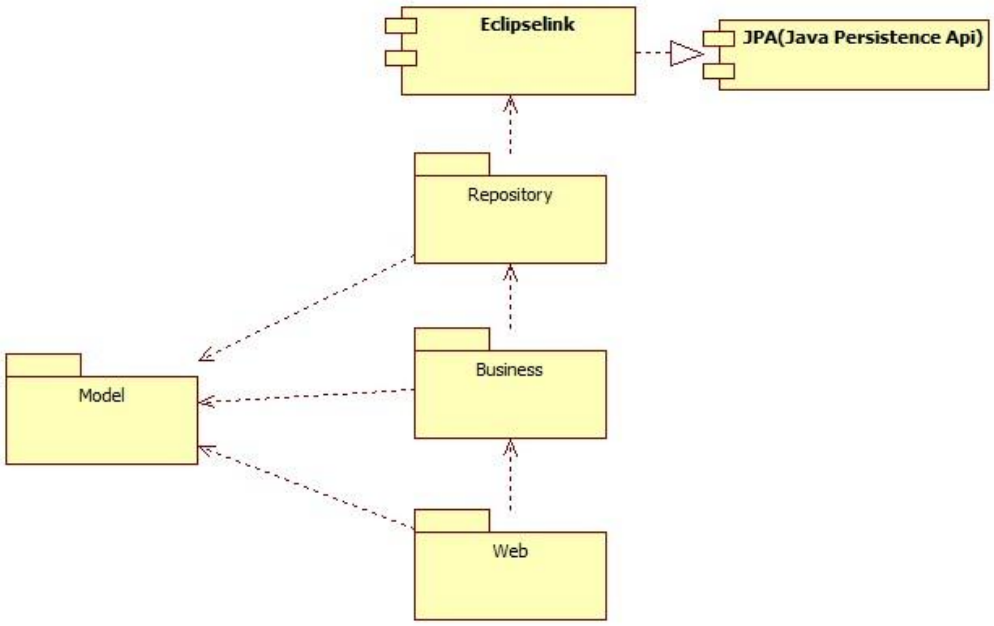

Figure 5.2 Modules diagram and their dependencies

5.2 Web module

The web module contains the web components and spring controllers.
Web components refer to:

- Jsp pages
- Tag files
- Images
- Cascading style Sheets (CSS)
- Javascript files (js)

A spring **controller** can be seen as Spring's MVC analogous to J2EE servlet specification. Therefore, the controller is used for handling a certain command. Spring controllers comes with a build in feature for **data binding and validation** which makes form submitted data to be binded to domain model objects. This feature will be covered in greater detail in a successing chapter.

In figure 5.3 the main classes from the web module are displayed.

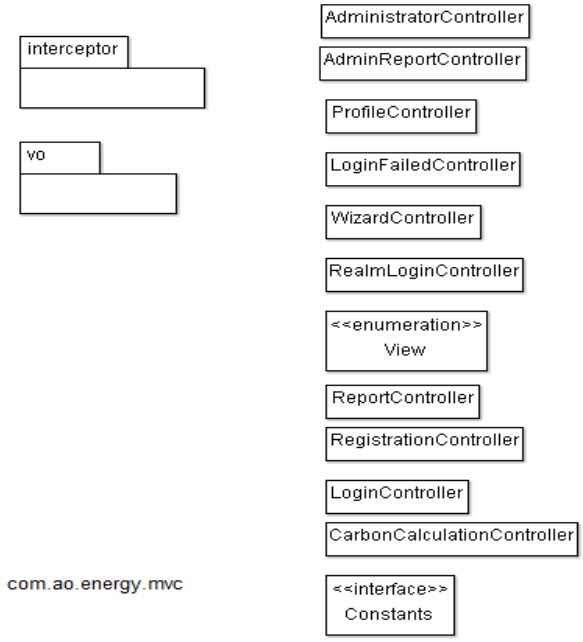

Figure 5.3 com.ao.energy.mvc package containing all the classes in the web module

5.2.1 Configuring Spring MVC

Spring MVC is itself a container, and like most containers it has a dispatcher servlet that will dispatch the request/command to the correct controller. The dispatcher servlet is configured just like any other servlet in the **web.xml** by specifying the url pattern and the servlet mapping from a given name to the dispatcher servlet class.

```
<servlet>
<servlet-name>appServlet</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
<init-param>
<param-name>contextConfigLocation</param-name>
<param-value>
/WEB-INF/spring/appServlet/servlet-context.xml, classpath:business.xml,
classpath:META-INF/repository.xml
</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>appServlet</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>
```

Code fragment: Spring dispatcher servlet configuration

One must notice that when declaring the dispatcher servlet the load-on-startup parameter is set to 1 this meaning that the servlet is loaded first when the servlet context is created. In this way we make sure that the dispatcher servlet will be loaded first and all future requests to the given url mapping is handled by it.

One must also notice that the **contextConfigLocation** parameter is set. When dispatcher servlet is loaded it also tries to load configuration files from a file ending with –servlet.xml in the WEB-INF classpath. This file usually contains bean definitions and other spring specific configuration. The contextConfigLocation changes that location and permits specifying the location of the configuration files. There is one for every module except model: one for business module where business bean definitions will be put, one for repository where repository beans will be put and one for general configurations.

Now the dispatcher servlet is ready to dispatch requests to controllers. The problem is that it does not know where the controllers are situated. The dispatcher servlet has to be told where to look for controllers in the classpath. This is done in the servlet-context.xml file which imports another configuration files controllers.xml.

<beans:import resource="controllers.xml"/>
In controllers.xml the packages that will hold controllers will be set in the following way:
<context:component-scan base-package="**com.ao.energy.mvc**"/>

To ease the development process by putting annotation on controllers such that the spring container will recognize them the annotation driven programming model needs to be enabled.

<annotation-driven/>

Now the dispatcher servlet knows that it has to look in the given package to search for controllers. Thus, as till now, the request comes to a controller which will delegate the business logic to a service but then it must display the service result. This is done by returning a view. Most controllers in the energy certificate report wizard application will return a string from their handler methods. That string will be mapped to a given view which can be any template page type: jsp pages, velocity codes pages, jasper reports files etc.

The above mentioned mapping is done by registering view resolvers. As the energy certificate report wizard will use jsp pages for displaying content and also needs to display jasper pdf reports which are in xml format, two **view resolvers** will be registered: XmlViewResolver and InternalResourceViewResolver.

```
        <beans:bean id="viewResolver"
class="org.springframework.web.servlet.view.XmlViewResolver">
            <beans:property name="location" value="/WEB-
INF/spring/appServlet/jasper.xml"/>
        </beans:bean>

        <!-- Resolves views selected for rendering by @Controllers to .jsp resources in
the /WEB-INF/views directory -->
        <beans:bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
            <beans:property name="prefix" value="/WEB-INF/views/"/>
            <beans:property name="suffix" value=".jsp"/>
```

```
</beans:bean>
```

By using the InternalResourceViewResolver view resolver whenever a string is returned from a controller, the resolver will try to find a jsp page with that name to display located at path: WEB-INF/views under the deployed web archive.

5.2.2 Configuring Reports Generation

The energy certificate report will be generated in PDF format. The global standard for generating PDFs has become Jasper Reports.

Spring comes integrated with Jasper Reports. The integration is done by registering an XmlViewResolver view resolver and then giving mapping between names and jasper reports xml files.

```
<bean id="report"
class="org.springframework.web.servlet.view.jasperreports.JasperReportsPdfView">
  <property name="url" value="/WEB-INF/reports/report.jrxml"/>
</bean>
```

After this configuration whenever a controller returns the view name “report” the report at the above written path is displayed.

5.2.3 Web components

The web application will be deployed in a web application archive under the webapps directory of tomcat.

The web components are can be split into two categories:

- Web components under WEB-INF directory – that cannot be seen from outside the application
- Resources outside of WEB-INF that can be seen

Under the WEB-INF the jsp pages are put as a measure of security. This assures that nobody can access jsp pages without going through a controller.

The resources outside the WEB-INF directory are put in the **resource** folder of the web archive. Under resources there are three folders: one for javascript files – js folder, one for images – images and one for cascading style sheets (css) – css folder.

5.2.4 Question wizard Spring controller

As detailed in the analysis and design part, the energy efficiency report can be generated only after the user – energy efficiency auditor has answered a set of questions that will gather data which will be supplied to a formula that will compute the necessary data to be put in the energy efficiency report.

In this subchapter the fetching and displaying of the wizard questions is made. The entry point for this operation is the WizardController class.

```
@Controller
```

```
public class WizardController {

  /** Repository that handles operations with Countys. */
  @Autowired
  private CountyRepository countyRepository;
```

It can be seen that the controller is annotated with **@Controller** which is used by the spring container to recognize a controller class type. When a new wizard is created, the county of the building needs to be taken from the DB therefore a county repository is needed. We can see in this example how the county repository is injected by the IOC container int the countyRepository object. The county repository that will be injected is a singleton which is made available for injection by putting another spring annotation **@Component** on the CountyRepository class.

```
@Component
public class CountyRepository {
```

This is an alternative way for creating a bean and making it available for injection. The other way is to define a bean through xml.

Displaying the create new wizard page.

The first step in starting a question wizard is displaying the first page. This is done using a method in this Controller class annotated with @RequestMapping:

```
@RequestMapping(value = "/audit/wizard/start", method = RequestMethod.GET)
public String prepareStartNewWizard(HttpSession session) {
User user = (User)
session.getAttribute(Constants.LOGGED_USER_SESSION_ATTRIBUTE);
if (user == null) {
return View.REDIRECT_REGISTER.getInternalViewName();
}
return VIEW_WIZARD_DETAILS;
}
```

Two other parameters are given to the annotation, the first one is the url pattern where this method can be found and then it is the http request method. The method simply checks if the user is logged and if so id displays the page where a new wizard can be created page. If not it displays the login page. The page that is displayed is the jsp page under WEB-INF/views which has the name of the returned string in this method. This is done by the internal view resolver that was configured earlier.

Starting the new wizard

```
@RequestMapping(value = "/audit/wizard/start", method =
RequestMethod.POST)
  public String startNewWizard(@ModelAttribute("wizard") @Valid
WizardResult wizard, BindingResult bindingResult,
                HttpSession session) {
    User user = (User)
session.getAttribute(Constants.LOGGED_USER_SESSION_ATTRIBUTE);
    if (user == null) {
      return View.REDIRECT_REGISTER.getInternalViewName();
```

```
        }
        if (bindingResult.hasErrors()) {
          return VIEW_WIZARD_DETAILS;
        }

        LOG.info("Starting a new wizard ");
        wizardRepository.createWizard(wizard, user);

        List<WizardResult> incompleteWizards =
wizardRepository.getIncompleteWizards(user);
        if ((incompleteWizards != null) && (!incompleteWizards.isEmpty())) {
          session.setAttribute(Constants.PENDING_REPORTS_ATTRIBUTE, true);
        } else {
          session.setAttribute(Constants.PENDING_REPORTS_ATTRIBUTE, false);
        }

        return "redirect:" + getQuestionPath(wizard.getId(),
Question.FIRST_QUESTION_INDEX);
      }
```

This method expects to get a wizard result composed of post request parameters. This is done by using the @ModelAttribute @Valid wizard in the method declaration.

ModelAttribute("wizard") @Valid WizardResult wizard

This method fetches the incomplete wizards, puts them on the session and then redirects to the controller that displays with the first wizard question. The result of this action, a jsp page can be seen in Figure 5.4.

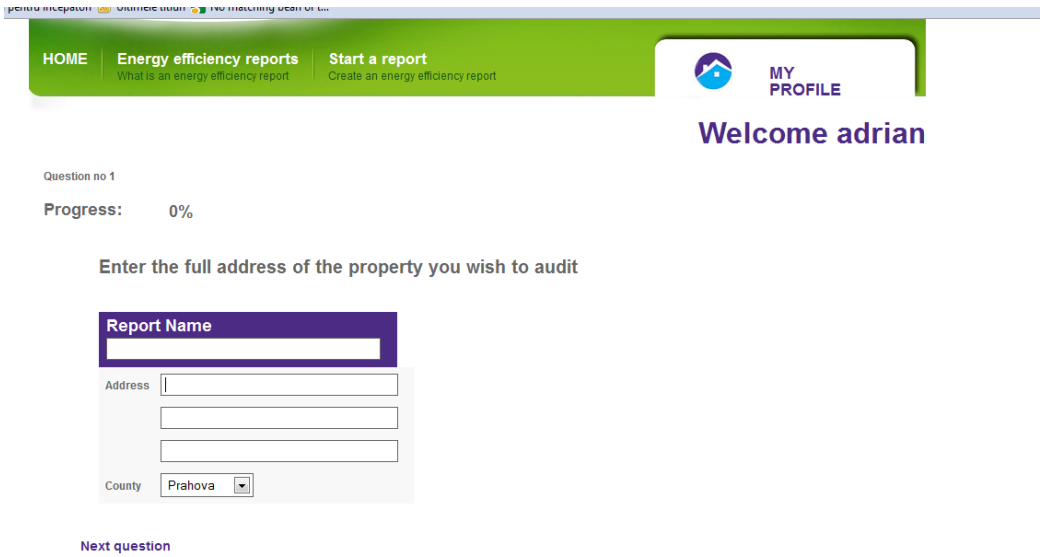

Figure 5.4 the first wizard question page

Fetching and displaying a question

The wizard is a set of controller operations: first step viewing the questions, then answering the question, then the answer is saved and the next question is fetched and displayed.

This step shows a question number questionNo from a given wizard report (the wizard with the given id).

It can be seen in the url value that a parameter can be given based on the url so when the link audit/wizard/1/q/2 the second question from the wizard with the given id is displayed. The wizard fetches the question with the given id and first checks if the given question is necessary in the current wizard, and if so it displays it by populating the model which is done in the populateQuestionView(model, question, wizard, false) method.

```
@RequestMapping(value = "/audit/wizard/{wizardId}/q/{questionNo}", method =
RequestMethod.GET)
public String viewQuestion(@PathVariable("wizardId") int wizardId,
@PathVariable("questionNo") int questionNr,
Model model, HttpServletResponse response, HttpSession
session) {
WizardResult wizard = validateWizardExistence(wizardId, response, session);
if (wizard == null) {
return null;
}

Question question = wizardRepository.getQuestionByNumber(questionNr);

if (question != null) {
if (skipConditionService.checkSkipConditions(wizardId, question)) {
return "redirect:" + getQuestionPath(wizardId, questionNr + 1);
}
populateQuestionView(model, question, wizard, false);
return VIEW_QUESTION;
}

QuestionsGroup group =
wizardRepository.getQuestionsGroupByNumber(questionNr);
if (group != null) {
populateQuestionsGroupView(model, wizard, group, false);
for (Question groupQuestion : group.getEntries()) {
if (groupQuestion.isVisible()) {
return VIEW_GROUP;
}
}
return "redirect:" + getQuestionPath(wizardId, questionNr + 1);
}

response.setStatus(HttpServletResponse.SC_NOT_FOUND);
return null;
}
```

A question can be skipped if a previous question had a specific answer. For example what kind of boiler do you have question can be skipped if the question of having a boiler is answered

with no. It can be seen on the last two lines that if the question is not found the not found response status is set in case a page not found page is shown (Figure 5.5)

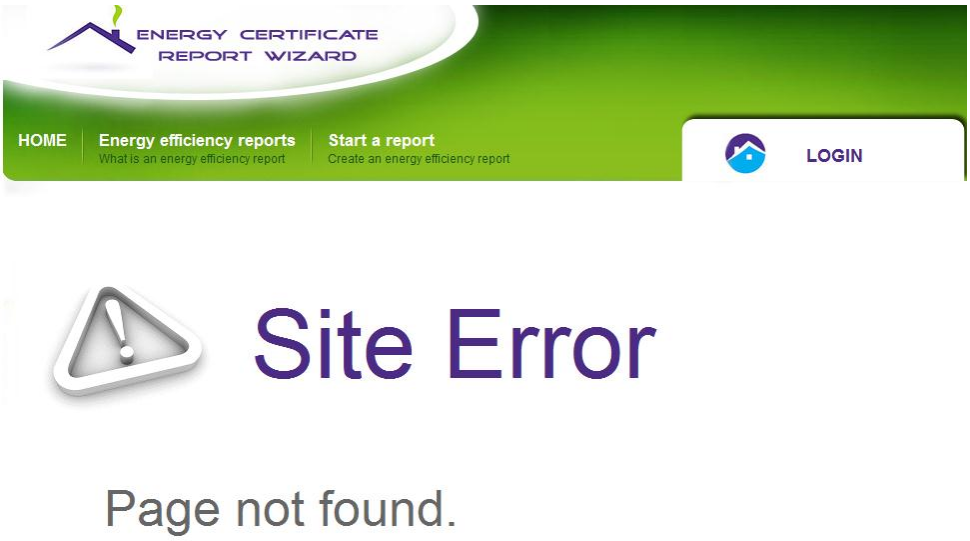

Figure 5.5 Page not found error page

Answering a wizard question

A question in the energy efficiency report wizard can be of 2 type: input number questions (questions that are answered if a given number) or a multiple choice type of question (choosing the answer from a given answer list). This logic is done on the answer question method and depending on them the answer is being validated. If the result is valid, the answer is saved by using the wizard repository.

```
public String answerQuestion(@PathVariable("wizardId") int wizardId,
@PathVariable("questionNo") int questionNr, @ModelAttribute
(ATTR_ANSWER) @Valid Answer answer, BindingResult bindingResult,
HttpServletResponse response, Model model, HttpSession session) {

  WizardResult wizard = validateWizardExistence(wizardId, response, session);
  if (wizard == null) {
    return null;
  }
  Question question = wizardRepository.getQuestionByNumber(questionNr);
  if (question == null) {
    response.setStatus(HttpServletResponse.SC_NOT_FOUND);
    return null;
  }
wizardRepository.addAnswerToQuestion(answer, question.getId(), wizardId);
  if (questionNr == TOTAL_FLOOR_AREA_QUESTION_NO) {
    computeDefaultArea(wizard, GROUND_FLOOR_AREA_QUESTION_NO);
    computeDefaultArea(wizard, LIVING_FLOOR_AREA_QUESTION_NO);
  }
  wizard.setLastModified(new Date());
```

```
            wizardRepository.updateWizard(wizard);
            if (questionNr == wizardRepository.getTotalWizardQuestionsNo()) {
              wizardRepository.completeWizard(wizard);
              User user = (User)
session.getAttribute(Constants.LOGGED_USER_SESSION_ATTRIBUTE);
              if (user != null) {
                List<WizardResult>                    incompleteWizards                    =
wizardRepository.getIncompleteWizards(user);
                if ((incompleteWizards != null) && (!incompleteWizards.isEmpty())) {
                  session.setAttribute(Constants.PENDING_REPORTS_ATTRIBUTE, true);
                } else {
                  session.setAttribute(Constants.PENDING_REPORTS_ATTRIBUTE,
false);
                }
              }
              return "redirect:" + VIEW_REPORT + "?id=" + wizardId;
            }
            return "redirect:" + getQuestionPath(wizardId, questionNr + 1);
          }
```

Also here extra care is done if for a question a certain case should be applied. For example when calculating the total floor area the input number must be modified and another value should be computed based on it.

Also, in the wizard controller question groups are processed. A question group is a type of question where many answers must be given to a give question. For example the question of the number of windows that are double glazed or single glazed. This custom business logic will not be covered but it is present in the wizard controller.

5.2.5 Security

The security of the application is cross cutting and its trigger point is any request that is done in the site. The security is done in two phases:

- Authentication
- Authorization

The security provider is the servlet engine itself. As described in the analysis and design phase the servlet container that is used is Apache Tomcat. It provides a generic way of applying security constraints based on the requested url patterns and roles.

This is done using realm security. The realm used for the energy efficiency report web application is the **JDBCRealm**. This realm is flexible and allows the authentication and authorization to be done via records in the database. The jdbc realm configuration is done in the server.xml file within the tomcat configuration folder. The way the jdbc realm is configured in the application is presented below.

```
<Realm className="org.apache.catalina.realm.JDBCRealm"
            debug="5"
            driverName="com.mysql.jdbc.Driver"
            connectionURL="jdbc:mysql://localhost:3306/energywizard"
            connectionName="root"
```

```
connectionPassword=""
userTable="user" userCredCol="password"
userRoleTable="roles" roleNameCol="role_name"
userNameCol="email" digest="MD5"/>
```

It can be observed that the mysql jdbc driver is used and the energy wizard database, via the connection url, is used for getting the users and roles.

The JDBCRealm needs to be configured with the user table and the roles table from where it will take the users and the roles associated for each user.

The **userTable** parameter is used as the user table, the **userCredCol** parameter is used as the column name of the password and the **userNameCol** is used as the column of the username which is used both in the roles and users table.

The digest parameter is an extra parameter that is used for security issues. It tells the realm that the password needs to be encoded with the md5 hashing algorithm before comparing it with the password stored in the database.

Now that the jdbc realm is configures the security must be put on the different sections of the web application. This is done by adding security constraints in the web.xml web application descriptor.

```
<security-constraint>
    <web-resource-collection>
       <web-resource-name>Secure Content</web-resource-name>
       <url-pattern>/protected/admin/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
       <role-name>admin</role-name>
    </auth-constraint>
  </security-constraint>
```

The above code fragment applies a security constraint on all the url pattern starting with /protected/admin. The authentication part is composed of checking if there is a user with the given email and password. The authorization phase is done only if the authentication succeeded and it is done by getting all the roles of the user and checking if it has the admin role. If the authentication is not successful e.g. there is no user with the given email and password a custom page will be displayed. If the authentication is successful but the authorization phase is not successful e.g. the user does not have the proper roles in order to see the page a not authorized page with 403 forbidden page is to be displayed.

When the user first requests a protected page a given login page is displayed and this is done by adding a configuration in the web.xml descriptor. A new login-config tag is added in the web.xml file just like below:

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>Energy Wizard</realm-name>
    <form-login-config>
       <form-login-page>/public/login</form-login-page>
       <form-error-page>/public/login</form-error-page>
    </form-login-config>
```

```
</login-config>
```

The form based login configuration is used because it allows displaying a custom page for login. The login page displayed is the page located at /public/login which is actually a controller method. The other page (form-error-page) is the page to be displayed if the authentication failed. This method is better than using a browser based version where a popup is displayed when the login is needed.

It can be seen that the not authorized page is cannot be configured in the login-config part. This can be done by adding an error page configuration by displaying the authentication page when the 403 status code is received just like below:

```
<error-page>
    <error-code>403</error-code>
    <location>/WEB-INF/views/errors/notauthorized.jsp</location>
 </error-page>
```

The controller mapped method located at /public/login simply returns the name of the jsp page where the login page is. The login page must contain a form with the action set to j_security_check and needs to have two fields: ne for username which needs to have the name : j_username and one for the password which needs to have the fields: j_password. The login page code is displayed below and the actual result can be seen in figure 5.6.

```
<form method="post" id="userLoginForm" action="j_security_check">
  To login please enter the email address and the password you registered with.
  <br/><br/>
  <span class="required">*</span> indicates required information
  <br/><br/>
  <table cellpadding="5" cellspacing="5" border="0">
    <tr><td><span class="required">*</span></td>
      <td style="font-size:12px; font-weight:bold;"><label
for="emailAddress"><s:message code="emailAddress"/></label>
      </td><td>
       <input type="text" id="emailAddress" class="textfield-main"
name="j_username"/>
      </td></tr><tr>
      <td class="error-message" colspan="3"><form:errors path="email"/></td>
     </tr><tr><td><span class="required">*</span></td>
      <td style="font-size:12px; font-weight:bold;"><label
for="passwordInput"><s:message code="password"/></label></td>
      <td><input type="password" id="passwordInput" class="textfield-main"
name="j_password"/>
      </td></tr> <tr>
      <td class="error-message" colspan="3"><form:errors
path="password"/></td></tr><tr><td> </td>
      <td valign="top"></td>
      <td align="right"><input type="submit" class="button-go-small-01"
value="Login" /></td></tr></table><br /></form>
```

Figure 5.6 Login page

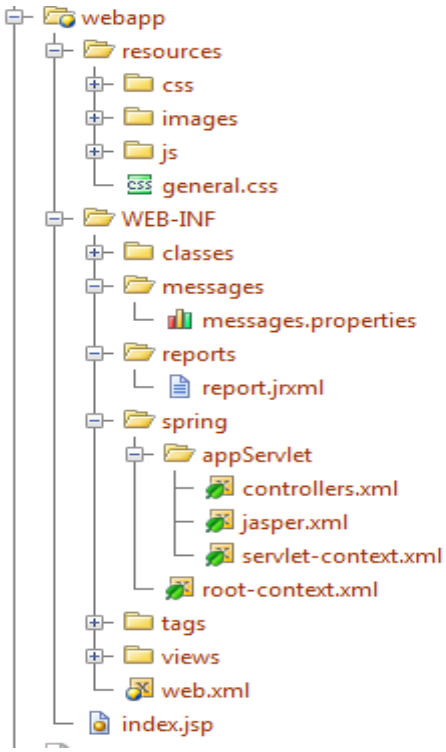

Figure 5.6 Web application archive structure

5.3 Model module

The model module is composed only of **JPA entity classes** which apply the pattern of database transfer objects (DTO).

The Java Persistence Architecture API (JPA) is a Java specification for accessing, persisting, and managing data between Java objects / classes and a relational database. JPA is now considered the standard industry approach for Object to Relational Mapping (ORM) in the Java Industry.

JPA itself is just a specification, not a product; it cannot perform persistence or anything else by itself. JPA is just a set of interfaces, and requires an implementation. There are open-source and commercial JPA implementations to choose from and any Java EE 5 application server should provide support for its use. JPA also requires a database to persist to. [2]

The core part of the JPA is the Entity. An entity class models an object that will be persisted in the database. An entity class may make use of auxiliary classes that serve as helper classes or that are used to represent the state of the entity. The entity class must be annotated with the Entity annotation or denoted in the XML descriptor as an
entity.

The entity class must have a no-argument constructor. The entity class may have other constructors as well.

The no-arg constructor must be public or protected. The entity class must be a top-level class. An enum or interface should not be designated as an entity. The entity class must not be final. No methods or persistent instance variables of the entity class may be final. If an entity instance is to be passed by value as a detached object (e.g., through a remote interface), the entity class must implement the **Serializable** interface. Entities support inheritance, polymorphic associations, and polymorphic queries.

For working with these entities in relationship with the database a new query language is used, Java Persistence API query language that will be used from now on as JPAQL.

According to the JPA specification from the enterprise java beans specification, [1] the Java Persistence query language is used to define queries over entities and their persistent state. It enables the application developer to specify the semantics of queries in a portable way, independent of the particular database in use in an enterprise environment. The Java Persistence query language can be compiled to a target language, such as SQL, of a database or other persistent store. This allows the execution of queries to be shifted to the native language facilities provided by the database, instead of requiring queries to be executed on the runtime representation of the entity state. As a result, query methods can be optimizable as well as portable.

The query language uses the abstract persistence schemas of entities, including their relationships, for its data model, and it defines operators and expressions based on this data model. It uses a SQL-like syntax to select objects or values based on entity abstract schema types and relationships among them. It is possible to parse and validate queries before entities are deployed. The JPAQL as the following statements: select, update, delete along with the from, where, group by, having and order by.

An **alternative** to the query language which is in fact a written representation of a query , a more OOP approach can be taken by using Criteria API, but which is not a JPA standard but it I s present in almost all JPA implementations as Hibernate **or Eclipselink**.

EclipseLink is an implementation of the JPA specification and provides sophisticated and high performance object-relational mapping services and complete support for the JPA specification. JPA objects are mapped through the package javax.persistence annotations and JPA persistence.xml and orm.xml. EclipseLink also provides extended annotations through the package org.eclipse.persistence.annotations. JPA provides a runtime API through the Entity Manager class in the javax.persistence package. EclipseLink also provides extended runtime API through the package org.eclipse.persistence.jpa.

EclipseLink supports any relational database that is compliant with SQL and has a compliant JDBC driver and the most important ones are : Oracle, MySql, Sybase, Microsoft SQL and PostgreSQL. EclipseLink has support for stored procedures and triggers.

The entity classes were identified based on the application persistence needs which are: the user and their roles need to be persisted, also the questions needed to be modeled along with all specific logics such as question groups, skipped questions and multiple choice answers. Also wizard results and user answers must be saved in the database.

The persistence provider used in this application was **Eclipselink.** The configuration file holding the eclipselink specific configuration must be put in the META-INF directory from the classes web application folder. The file name must be persistence.xml and as the extension sais is in xml format.

```
<persistence-unit name="energy">
  <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
  <jta-data-source>/com/ao/EnergyDS</jta-data-source>
  <class>com.ao.model.User</class>
  <class>com.ao.model.County</class>
  <class>com.ao.model.Answer</class>
  <class>com.ao.model.Option</class>
  <class>com.ao.model.Question</class>
  <class>com.ao.model.InputNumberAnswerQuestion</class>
  <class>com.ao.model.MultipleOptionsQuestion</class>
  <class>com.ao.model.WizardResult</class>
  <class>com.ao.model.SkipQuestionCondition</class>
  <class>com.ao.model.QuestionsGroup</class>
  <class>com.ao.model.EnergyRating</class>
  <class>com.ao.model.UserRole</class>
  <exclude-unlisted-classes>false</exclude-unlisted-classes>
  <shared-cache-mode>ALL</shared-cache-mode>
  <properties>
   <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver"/>
   <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost:3306/energywizard"/>
   <property name="javax.persistence.jdbc.user" value="root"/>
   <property name="javax.persistence.jdbc.password" value=""/>
   <property name="eclipselink.target-database" value="MySQL"/>
  </properties>
 </persistence-unit>
```

In the above code it can be seen that a persistence unit element is defined. The name attribute is used for making the database configuration available for persistence unit dependency injection needed in the repository module. The provider element configures the provider of the persistence api and in this application is the **eclipselink provider**.
The class elements inform the persistence provider which classes should be considered as JPA entity classes.

An entity class must be POJO (plain old java object) with setters and getters for properties, with a default empty constructor and annotated with the @Entity annotation and also must implement the Serializable markup interface.

```
@Entity
@Table(name = "USER")
public class User implements Serializable {

  /** The id for this entity, representing the id of the table. */
  @Id
  @Column(name = "ID")
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Integer id;

  /** The email address of the user. */
  @Column(unique = true, length = 100, nullable = false)
  @Email
  @Size(max = 100)
  @NotEmpty
  private String email;

  /** The first name of the user. */
  @Column(length = 50, nullable = false)
  @NotEmpty
  @Size(max = 50)
  private String firstName;

   /** The password of the user. */
  @Column(length = 50, nullable = false)
  @NotEmpty
  @Size(min = 6, max = 50)
  private String password;
```

The above code is a snippet of the User entity class. Along with the @Entity annotation, the **@Table** annotation serves for specifying the name of the MySql table on which the entity is mapped. Also it can be seen that on the class attributes that the **@Column** annotation is used where validation constraints can be put on that field such as maximum length, if it is optional or unique. Also the name of the column can be put, but the name of the field is used by default in this case as no name attribute is given to the @Column annotation.

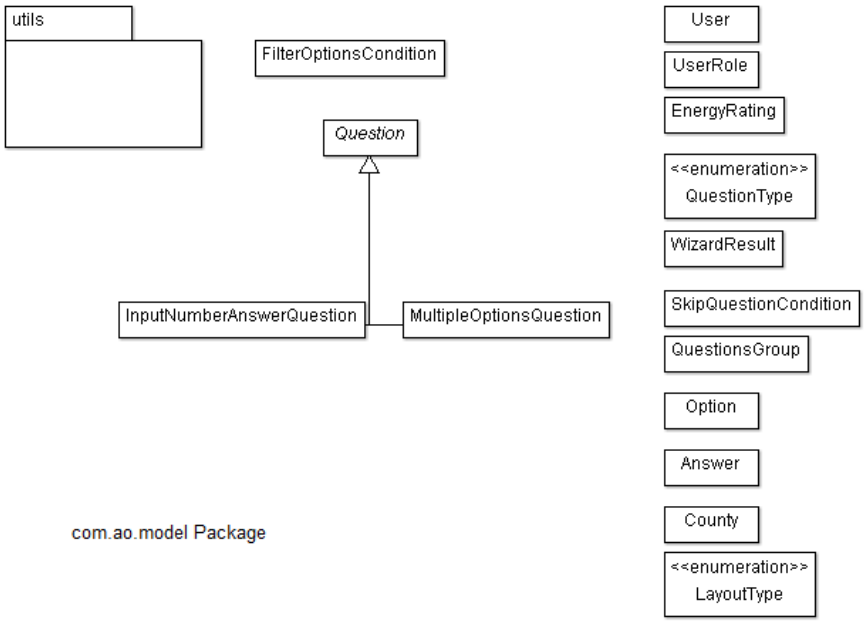

Figure 5.7 com.ao.model package containing all the entity classes

Based on the above entity classes the MySql tables were generated.

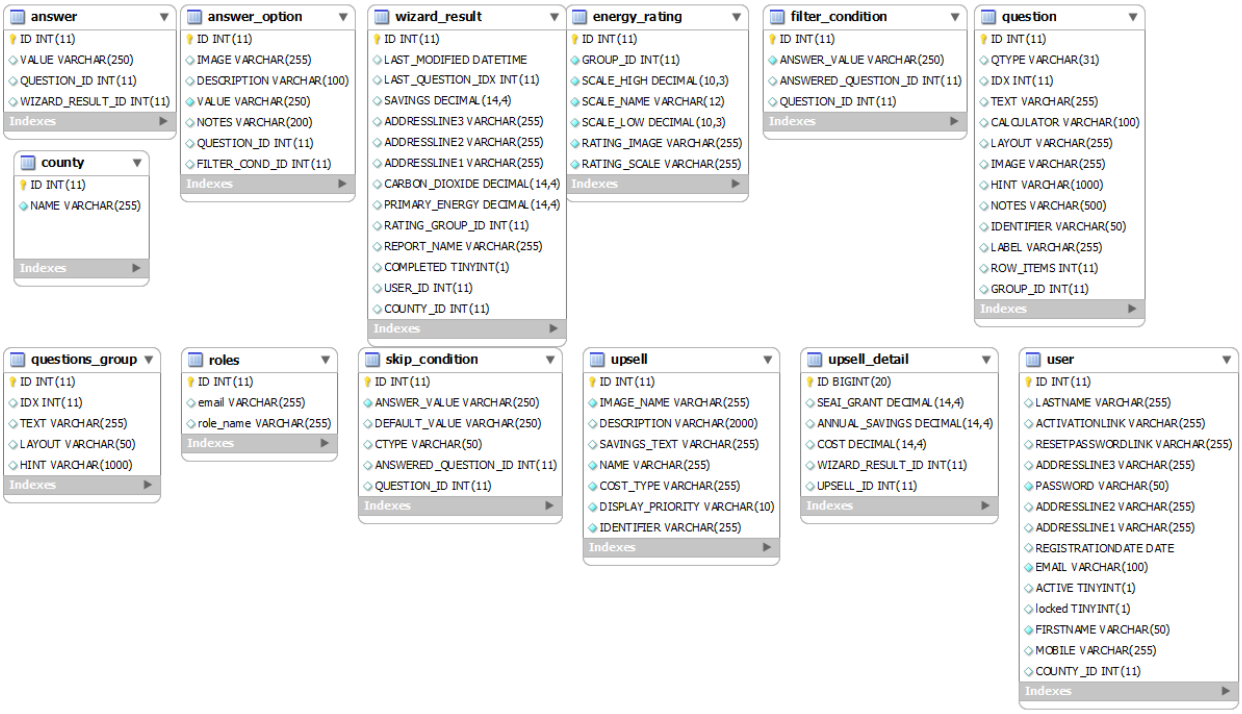

Figure 5.8 Database tables diagram

Based on the entity classes that were defined in the persistence.xml files the MySql schema was generated as shown in figure 5.8.

5.4 Business module

The business module holds business logic services and other handlers needed for computing the energy efficiency report wizard. The wizard web part only takes data and delivers them to the services within this layer which will process the data by persisting, updating and fetching data using the needed repositories.

In figure 5.8 the business main package is displayed. As it can be seen it contains 5 subpackages: calculator, skip conditions, workbook, report and utils.

The **calculator package** contains four different calculators: floor area calculator, total floor calculator, ground floor calculator and default calculator. These are utility classes that processes certain question answers and calculates the needed values.

The **report package** holds business logic classes used for computing the energy efficiency certificate report data such as carbon dioxide footprint, the energy used for heating water, the energy used for ventilation, the energy used for heating.

The most important class in the reports package is the ReportCalculator class which holds methods for calculating the final results that will be displayed in the generated energy efficiency reports. The formula used for computing the energy efficiency certificate results is contained in an excel file where all the **calculations** are done. The job of the application is to put the user answers in the excel file which will then make all the results available in a well known set of cells. The reading and writing of cells is done using POI which lets user create/update/read excel files.

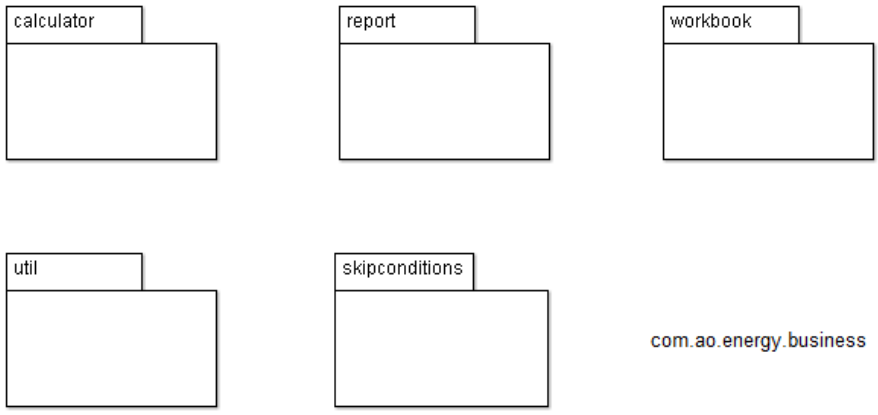

Figure 5.9 com.ao.energy.business package diagram

The file that will compute the final results is calculations.xls and must be placed in the web application classpath.

Remedial Measures	Typical Installation Cost (Lei)		Installation cost less grant	Annual Saving (€)	Payback Period (Years)	Carbon Emissions reduction (Tonnes of CO_2 /annum)	New Energy Score	New Rating	Applicable
Upgrade Ceiling Insulation to 300mm	€1,155.00	€250.00	€905.00	€28.88	31.3	0.22	826.43	G	TRUE
External Wall insulation	€27,315.40	€4,000.00	€23,315.40	€854.13	27.3	6.38	513.92	G	TRUE
Cavity Wall Insulation	€2,091.06	€400.00	€1,691.06	€718.49	2.4	5.37	565.29	G	FALSE
Internal dry lining	€12,846.30	€2,500.00	€10,346.30	€854.13	12.1	6.38	743.71	G	TRUE
91% efficient gas boiler & Controls	€2,995.00	€700.00	€2,295.00	€1,335.34	1.7	12.07	339.74	E1	FALSE
91% Oil Boiler & Controls	€3,277.00	€700.00	€2,577.00	-€332.42	-7.8	11.28	339.74	E1	FALSE
91% LPG boiler & controls	€2,955.00	€700.00	€2,255.00	-€1,840.52	-1.2	11.80	339.74	E1	FALSE
Heating controls	€1,512.14	€500.00	€1,012.14	€0.00	0.0	0.00	837.37	G	TRUE
Double Glazing	€14,170.00	€0.00	€14,170.00	€123.36	114.9	0.92	823.83	G	TRUE
Energy Efficient lights	€0.00	€0.00	€0.00	€0.00	0.0	0.00	837.37	G	FALSE
Lagging Jacket	€20.00	€0.00	€20.00	€246.62	0.1	1.25	803.13	G	TRUE
Full Central Heating Retrofit	€99,999.00	€99,999.00	€99,999.00	€99,999.00	999999.0	0.00	0.00	G	TRUE
Current Carbon Emissions	19.01891693	Tonnes CO2 Per annum			0.0	A1			
Current Energy Score	837.3667348	Kwh/m2 per annum			25	A2			
Current Rating	G				50	A3			

Figure 5.10 Snippet of the calculations.xls formula

One of the most important functionalities of this web application is to get the formula results. The below detailed method does that.

```
public FormulaResult getFormulaResults(WizardResult wizardResult) throws Exception {
    FormulaResult formulaResult = null;
    WorkbookHandler workbookHandler = null;
    try {
      workbookHandler = (WorkbookHandler) workbookPool.borrowObject();
      logger.info("Computing formula results for wizard with id " + wizardResult.getId() +
          ". After borrowing object, pool status is: BORROWED: " + workbookPool.getNumActive() + " IDLE: " +
          workbookPool.getNumIdle());
      UserAnswers userAnswers = getUserAnswers(wizardResult);
      setUserAnswers(workbookHandler, userAnswers);
//      workbookHandler.saveToFile("e:/workspace/results.xlsx");
      formulaResult = getFormulaResults(workbookHandler);
      logger.info("Results computed successfully for wizard with id " + wizardResult.getId());
    } finally {
      if (workbookHandler != null) {
        workbookPool.returnObject(workbookHandler);
        logger.info(
            "After returning object for wizard with id " + wizardResult.getId() + ", pool status is: BORROWED: " +
            workbookPool.getNumActive() + " IDLE: " + workbookPool.getNumIdle());
      }
    }
    return formulaResult;
  }
```

The business logic that is performed here is: the answers to the wizard questions are extracted from the database using the getUserAnsers() method. The answers are then passed. Then the user answers are put in the excel file that will compute the final results. The answers are wrote in the excel file with the setUserAnswers method. After writing the user answers the result can be taken from the excel file and this is done by calling the getFormulaResult method.

POI API usage

In the below example the usage of the POI api is displayed. The method getEvaluationResult gets the main result of the energy efficiency certificate, the result that states in which energy class the building is.

```
private Object getEvaluationResult(WorkbookHandler workbookHandler, String
absoluteCellReference)
    throws ReportCalculatorException {
  int indexOfDelimiter = absoluteCellReference.indexOf("!");
  String sheetName = absoluteCellReference.substring(0, indexOfDelimiter);
  String cellReference = absoluteCellReference.substring(indexOfDelimiter + 1);
  XSSFSheet sheet = workbookHandler.getSheet(sheetName);
  if (sheet == null) {
    throw new ReportCalculatorException("Sheet named " + sheetName + " not
found");
  }
  CellReference cellRef = new CellReference(cellReference);
  XSSFRow row = sheet.getRow(cellRef.getRow());
  XSSFCell cell = row.getCell(cellRef.getCol());
  return workbookHandler.evaluate(cell);
}
```

The method gets as parameter a reference to the excel formula file in the workbook handler object and also the position from where the result should be taken.

First the excel sheet is fetched and then a cell reference in fetched based on the input absolute path parameter. Then the value from that given cell is taken when calling workbookHandler.evaluate(cell).

The evaluate method from the workBookHandler object is detailed below:

```
public Object evaluate(XSSFCell cell) {
  CellValue cellValue = evaluator.evaluate(cell);
  Object returnValue = null;
  switch (cellValue.getCellType()) {
    case Cell.CELL_TYPE_BOOLEAN:
      logger.info("Cell " + cell.getReference() + " in sheet " +
cell.getSheet().getSheetName() +
          " evaluated to Boolean value " + cellValue.getBooleanValue());
      returnValue = cellValue.getBooleanValue();
      break;
     ….
    case Cell.CELL_TYPE_FORMULA:
    break;
```

```
        }
        return returnValue;
    }
```

The method simply uses an evaluator that evaluates the cell which returns a CellValue object that is just a wrapper for a type which can be a Boolean, a number or a string. Then it checks the type of the cell value and based on it the wrapped value is returned as a result.

Generating the energy efficiency certificate report

The energy efficiency report is generated using jasper reports. As explained in chapter 5.2 in the web module, the report will be generated when a string with the name of the japser xml file is returned from a spring controller. The jasper report xml file was build using **iReports** which is a tool for designing reports using a friendlier UI which is an alternative to writing the reports in xml. The jasper report also has references to data that is being passed as a model from the controller. That model contains data that is dynamic in the reports such as the overall energy efficiency result, the name of the owner, the building's address and paths to images that are to be displayed in the report.

The method below details the way the map that will be passed to the report is populated.

```
public Map<String, Object> getWizardModel(WizardResult wizardResult, FormulaResult formulaResult, String contextPath, EnergyRating energyRating) {
        Map<String, Object> model = new HashMap<String, Object>();
        BigDecimal thresholdScore = new BigDecimal(upgradeThresholdScore);
        JRDataSource jrDataSource = null;
        model.put("addressLine1", wizardResult.getAddressLine1());
        model.put("addressLine2", wizardResult.getAddressLine2());
        model.put("addressLine3", wizardResult.getAddressLine3());
        model.put("county", wizardResult.getCounty().getName());
        model.put("carbonDioxide", formulaResult.getCurrentCarbonFootprint());

        // this drives the top text in the 3B section, if under threshold score show 'Above average' copy
        model.put("showEfficient3B", (formulaResult.getCurrentEnergyScore().compareTo(thresholdScore)) <= 0 ? "yes" : "no");

        // get energy rating properties to determine what scale to use
        model.put("ratingLineImage", energyRating.getRatingImage());
        model.put("ratingScaleImage", energyRating.getRatingScale());

        model.put("averageCarbonDioxide", averageCarbonEmission);
        model.put("energyRating", energyRating.getScaleName());

        //jasper needs absolute path for images in report
        StringBuilder imageContextPath = new StringBuilder(contextPath);
        imageContextPath.append(StringUtils.isNotEmpty(imageContext) ? imageContext : "");
        imageContextPath.append("/");

        model.put("imageContext", imageContextPath.toString());
```

```
        // pass an empty data source if no upsells
        jrDataSource = new JREmptyDataSource();
        model.put(ReportDetailService.ATTR_JASPER_DATASOURCE, jrDataSource);
        return model;
    }
```

As it can be seen a map is constructed and the energy rating results are being put as values to the map keys. The map keys will be used in the jasper xml files in order to display the energy efficiency report results. It can be seen that the address line is sent along with the results such as energy rating and average carbon dioxide emissions.

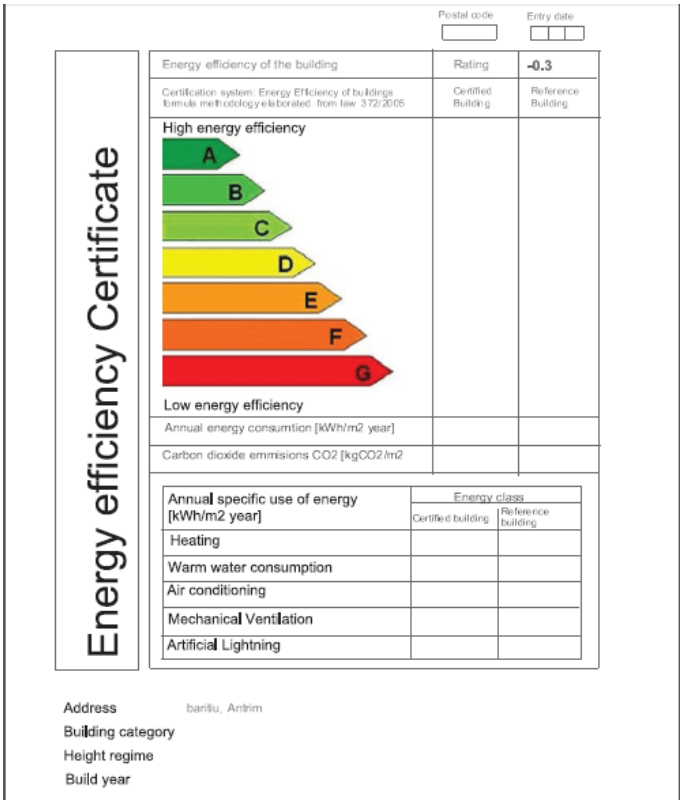

Figure 5.11 Snippet of the energy efficiency report

Chapter 7 User's manual

7.1 Installation description

The application will be build using the create artifact tool from the Intelij Idea IDE. The artifact that will be choosed from the artifact list is energy.war.

According to the deployment diagram, if the application should be put in production for general use, 4 machines should support the application:

- One machine that will be in the demilitarized zone which will host the load balancer and the webserver
- Two machines that will both host a tomcat servlet container each with it's own web archive (energy.war)
- One machine that wil hold the MySql database server

The machine in the DMZ (demilitarized zone) must have the apache web server installed and configured as a load balancer and as a firewall using the custom apache module. The hardware configuration is the following:

Windows platforms (2000, XP, Server 2003, Vista, 7, Server 2008, 32- & 64-bit versions)

- 512 MB of RAM (2000, XP), 1024 MB (others)

Linux platforms (32- & 64-bit versions)

- 512 MB of RAM

The two machines holding each one a **tomcat servlet container** must have java 1.5 or above installed, and also must have the apache tomcat 6.0 servlet container installed. The energy.war web archieve must be put in the configured webapps folder of tomcat (by default webapps). The hardware configuration for the two machines:

Windows platforms (2000, XP, Server 2003, Vista, 7, Server 2008, 32- & 64-bit versions)

- 2GB of RAM (2000, XP), 4GB MB (others)
- Minimum 2GHz processor single CPU processor
- 50MB of free space for the energy efficiency application

Linux platforms (32- & 64-bit versions)

- 2GB of RAM
- Minimum 2GHz processor single CPU processor
- 50MB of free space for the energy efficiency application

CentOS platform:

- 2GB of RAM
- Minimum 2GHz processor single CPU processor
- 50MB of free space for the energy efficiency application

The **machine holding the MySql Server** should have MySql 5.0 or above installed and must have the following system requirements:

- 1GB of RAM
- Minimum 2GHz processor single CPU processor

- 500B of free space for the MySql server and it's indices

The energy.sql which creates the energy efficiency needed database and it's table must be run in the MySql server console.

7.2 User Manual

When any user first accesses the site the login page is displayed.

Figure 7.1 Login page

If the login page the initial requested page is dispayed. If the login fails the login form will be displayed once again.

Based on the user role some sections are visible or not. An admnistrator can access any page within the site, the auditor can access anything but the adminstration section and the guest can access only read only content.

If the user tries to access a page that cannot be seen because it does not have a proper role the access denied page below will be displayed.

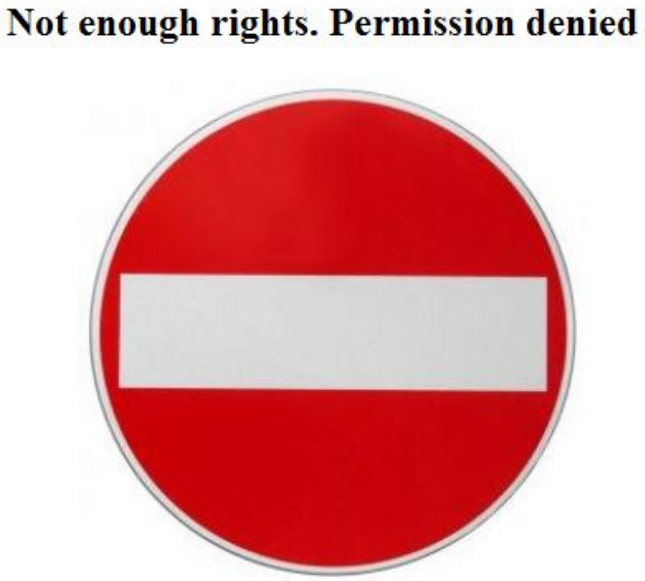

Figure 7.2 Access denied page.

The home page has links to the energy wizard start page and to a description page where it is described what the purpose of this applciation.

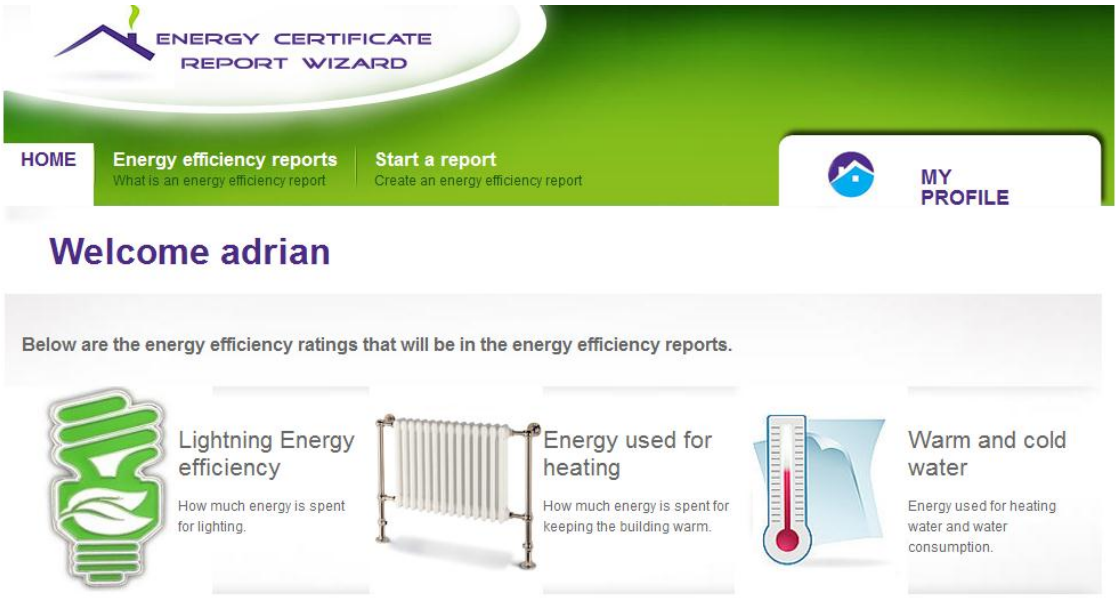

Figure 7.3 Home page

If the logged user is an admniistrator it can access the administration panel at the link **/protected/admin/home.** There are two pages for the administrator: one for reports administration and one for user administration.

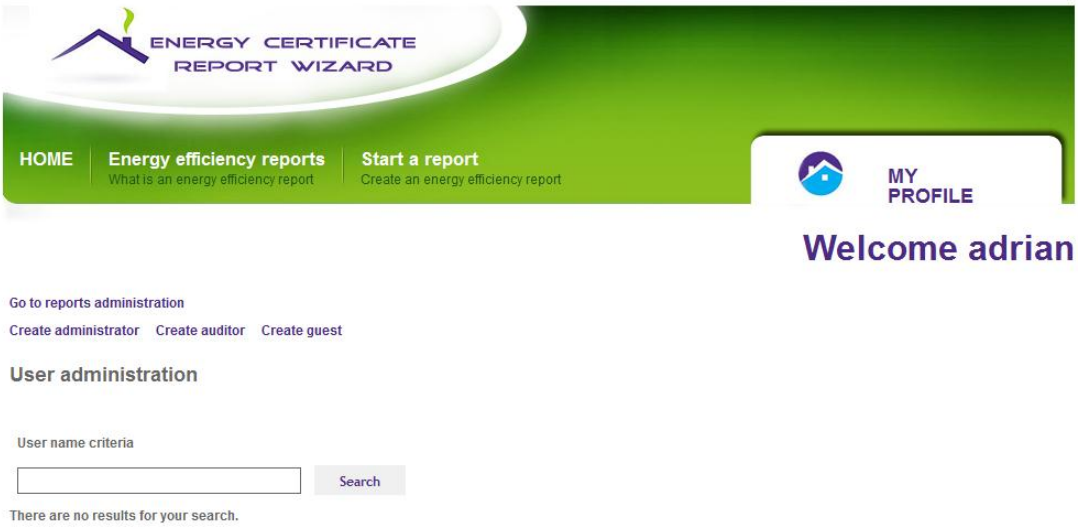

Figure 7.4 User Administration page

From the user administration page the user can access the reports page and can choose to create different users. If the search criteria returns result the found users can be locked, removed, updated.

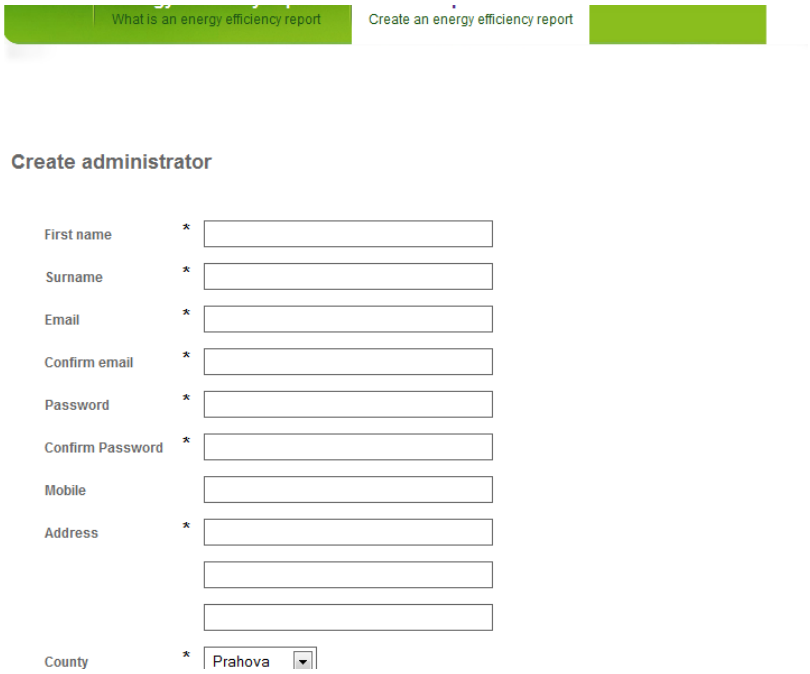

Figure 7.5. The create administrator page

When accessing the third link from the menu a new energy efficiency report can be started. Below the first question of the energy efficiency report wizard is displayed.

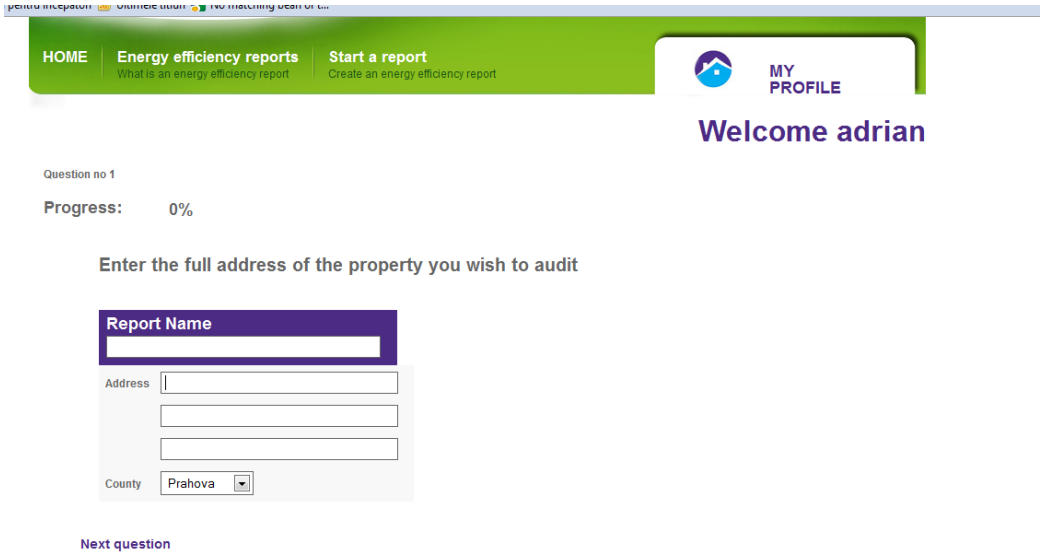

Figure 7.6. The first wizard question

After answering all the wizard questions instead of the next question link a **Generate report link** will be displayed. This link will generate the report and send the user to the my profile page where all his reports can be managed (update, delete, create). In the auditor reports page the completed reports can be seen in PDF format ad downloaded.

Chapter 8 Conclusions

8.1 Achievements

The objective of this project was to implement an energy efficiency report tool that will be give energy efficieny auditors the means to create energy efficiency reports by having to fill in as less data as possible. The tool should have also been implemented as an web application.

The energy efficiency reports should contain the following computed energy rating attributes:

- The energetic class the building is in (A,B,C,D etc)
- The annual energy consumption
- The anual carbon dioxide emmisions
- The annual specific use of energy for: heating, warm water consumption, air conditioning, air ventilation and artificial lightining

From these requirements only the **carbon dioxide emissions** is calculated by the application and the other ones leave space for future improvements.

All other specification functional requirements were implemented and validated.

All the non functional requirements were met:

- **Usability**: the application user interface was designed to be intuitive and friendly, also there is a clear separation of responsabilities and thus the primary user: the auditor can create reports easily in less then 15 minutes if all the data was gathered previously
- **Accessability:** the application can be accessed by anyone who has an internet connection and has an account
- **Security:** the application is secured from outside ill willed users but also against users that have an account
- **Availability:** the application deployment ensures that if a server fails the other server will be available and thus the application will be working
- **Extensibility**: the application design supports adding new questions in the wizard and also the formula can be changed easily by updating the excel file containing the formula

8.2 Future improvements

The reports aplication must be updated regularly because the energy efficiency laws can be changed and the formula must be updated to

The application leaves space for future improvements and some of them are:

- implementing the calculation of all the needed data that ought to be present in the energy efficiency report (heating, warm water consumption, air conditioning, air ventilation and artificial lightining)
- update the energy efficiency report by improving the design

Bibliography

[1] **Enterprise JavaBeans JPA –** JSR 220: Enterprise JavaBeans TM, Version 3.0

[2] **Wikipedia**:

a. Jasper Reports http://en.wikipedia.org/wiki/JasperReports
b. Java Persistence API - http://en.wikibooks.org/wiki/Java_Persistence/What_is_JPA%3F
c. Apache Tomcat - http://en.wikipedia.org/wiki/Apache_Tomcat
d. Http Protocol - http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
e. Grasp principles - http://en.wikipedia.org/wiki/GRASP_%28object-oriented_design%29

[3] **Spring Framework homepage** Spring MVC architecture - http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html

[4] GRASP - Loose Coupling - **Inversion of Control principle article** http://javaboutique.internet.com/tutorials/loose/

[5] Apache Tomcat-**Tomcat: The Definitive Guide Second edition, Jason Brittain and Ian F. Darwin, O'Reilly Media October 2007**

[6] Apache Tomcat -**Pro Apache Tomcat 6, Mathew Moodie, APress 2007**

[7] JasperReports for Java Developers, by **David R. Heffelfinger, PACKT Publishing 2007**

[8] HTML, CSS, javascript - **http://w3schools.com/**

[9] **Java sun homepage -** http://java.sun.com/blueprints/patterns/MVC-detailed.html

Acronyms

POI – Poor Obfuscation Implementation
JPA – Java Persistence API
IDE – Integrated Development Environment
IoC – Inversion of control
DI – dependency injection
AOP – aspect oriented programming
J2EE - Java 2 Platform, Enterprise Edition
MVC – model view controller
JSP – java server pages
XSLT – XML (Extensible Markup Language) Stylesheet Language for Transformations
JPAQL – java persistence language query language
API – application programming interface
SQL – structured query language
XLS – Microsoft excel sheet file
HSSF - Horrible Spreadsheet Format
DTD – document type definition
DMZ – demilitarized zone

List of figures:

Page 15: Figure 3.1 Spring Configuring based on xml written configuration metadata
Page 16: Figure 3.2 Model View Controller Architectural pattern diagram
Page 17: Figure 3.3 Spring MVC request handling mechanism high level view
Page 18: Figure 3.4 Spring context hierarchy
Page 21: Figure 4.1 System block diagram
Page 22: Figure 4.2 Administrator Role Use case diagram
Page 24: Figure 4.3. Energy efficiency auditor use case diagram
Page 24: Figure 4.4 Guest use case diagram
Page 29: Figure 4.5 Layerd web application architecture diagram
Page 30: Figure 4.6 Model-view-controller web application architecture diagram
Page 32: Figure 4.7 Two tomcats in a web farm topology hosting the energy web archive
Page 33: Figure 5.1 Detailed System Block Diagram
Page 34: Figure 5.2 Modules diagram and their dependencies
Page 35: Figure 5.3 com.ao.energy.mvc package containing all the classes in the web module
Page 39: Figure 5.4 The first wizard question page
Page 41: Figure 5.5 Page not found error page
Page 45: Figure 5.6 The login page
Page 45: Figure 5.7 Web application archive structure
Page 49: Figure 5.8 com.ao.model package containing all the entity classes
Page 49: Figure 5.9 Database tables diagram
Page 50: Figure 5.10 com.ao.energy.business package diagram
Page 51: Figure 5.11 Snippet of the calculations.xls formula
Page 54: Figure 5.12 Snippet of the energy efficiency report

Page 56: Figure 7.1 The login page
Page 57: Figure 7.2 The permission denied page
Page 57: Figure 7.3 The home page
Page 58: Figure 7.4 The user profile page
Page 58: Figure 7.5 The create administrator page
Page 59: Figure 7.6 The last wizard question