Struts and Hibernate in practice

Shixing Wang

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

PARIS

_____

MEMOIRE

présenté en vue d'obtenir

le DIPLOME d'INGENIEUR CNAM

SPECIALITE : informatique

OPTION :Architecture et Ingénierie des systèmes et des logiciels

par

WANG Shixing

_____

Struts and Hibernate in practice

Soutenu le 12 décembre 2011

_____

JURY

PRESIDENT : Nicole Levy

MEMBRES : Olivier Pons

Pierre Courtieu

Abstract

In order to investigate the advantages and disadvantages of the Hibernate and Struts frameworks, this thesis was performed through a case study including two web applications to make a comparing for it.

The first case is a Bank Management System which is implemented during my internship, using Hibernate 3.2 and Struts 2. In the second case Java EE 5 and EJB3 were used without any frameworks. There is a need for studying these technologies, apart from Struts and Hibernate. The report contains a short introduction of these.

Then there‘s a description of the found advantages and disadvantages of the frameworks and the conclusion will be made.

Finally the thesis will make some improvements on the Bank Management System, add a new framework called ―Spring", constitute SSH (Spring + Struts + Hibernate) which is also a popular structure we usually used, and analyses its advantages and disadvantages.

Key words: Struts 2, Hibernate, Spring.

Résumé

Afin de trouver les avantages et les inconvénients de Hibernate et de Struts frameworks. Ce thème est prouvé à travers d'une comparaison entre deux Web applications.

Le premier cas est un « Bank Management System », qui a été réalisé pendant mon stage, en utilisant Hibernate 3.2 et Struts 2 ; quant au deuxième cas, Java EE 5 et EJB3 sont appliqués sans aucun framework. Sauf Struts et Hibernate; nous avons aussi besoins de saisir les technologies appliqués dans ces deux programmes. Nous verrons queleques explications dans ce thème.

Ensuite, je vais vous expliquer les avantages et les inconvénients découverts de ces frameworks, et puis, je vous donnerai une conclusion.

Enfin, il me paraît que nous puissions améliorer le « Bank Management System » en ajoutant un nouveau framework—« Spring », nous pouvons les constituer SSH (Spring + Struts + Hibernate) qui est un framework commun et populaire, et analyser leurs avantages et inconvénients.

Mots clés: Struts 2, Hibernate, Spring.

Acknowledgement

I would like to express my gratitude to all those who helped me during the writing of this thesis. I gratefully acknowledge the help of my supervisor, Prof. Pierre Courtieu, who has offered me valuable suggestions in the academic studies. In the preparation of the thesis, he has spent much time reading through each draft and provided me with inspiring advice. Without his patient instruction, insightful criticism and expert guidance, the completion of this thesis would not have been possible.

High tribute shall be paid to my manager in Groupe HN, whose profound knowledge of J2EE programming and whose earnest attitude tells me how to develop the project.

I am also deeply indebted to all the other tutors and teachers in CNAM, from whose devoted teaching and enlightening lectures I have benefited a lot and academically prepared for the thesis.

Finally, I owe much to my friends and parents for their continuous support and encouragement.

Table of contents

Table of Figures

1. BACKGROUND AND PURPOSE

This paper describes an investigation of the benefits and disadvantages of using the Hibernate and Struts frameworks compared to using none of the frameworks but only Java EE 5, Servlet and JSP technologies. The thesis includes developing a system for GCE Technologies when we take an internship in Groupe HN. The section –Two cases' design and solution" will explain more about the specification of the system.

1.1 Company

The group HN is a software company founded in 1983 which now has 400 employees in the sectors of banking, insurance, industry and services. It was founded by Michael HOCHBERG and Lionel NIQUET in Paris, and two letters "HN" are an abbreviation of the name of these two creators (HOCHBERG NIQUET).

The main service of the company is software development, it consists 3 major areas of activity:

- Engineering for technical studies, assistance with project management.
- Product information engineering, such as system engineering expertise and networks.
- Engineering training, which is called –HN Institute", the training center of the Groupe HN offers two main training programs: IBM Mainframe (Client / Server) and JAVA/J2EE.

70% of Groupe HN revenues are performed with the main actors in financial services (banking, insurance and pension funds).

In recent years, the Company intends to set up a Chinese subsidiary in Shanghai. But the time difference existed between China and France is very interesting. China has 6-7 hours ahead of France, and this period may sometimes be a challenge for project delay, which is unavoidable in the service sector. And as everybody knows, the cost of labor is quite low in China, for this purpose, the company hired many Chinese students to give them computer training and then send them to the subsidiary after the internship.

1.2 Customer

GCE Technologies ensures the activities of IT project management of Saving Banks (Caisses d'Epargne in France). It provides technical solutions for GCE Business Services which can offer specific requirements of Saving Banks.

GCE Technologies develops and operates the information system of Savings Banks and Subsidiaries, in compliance with regulations, current standard, service agreements and IT strategy within the framework. It is responsible for the sustainability of the information system for various subsidiaries of Saving Banks.

The company brings together 2,100 employees across 15 sites: Lille, Strasbourg, Nancy, Rouen, L'Isle Adam, Bagnolet, Montrouge, Orleans, Rennes, Nantes, Bordeaux, Toulouse, Aix-en-Provence, Toulon and Lyon.

GCE Technologies and Groupe HN have a long-term relationship, In order to manage their control, now they want us to develop a system which should have many functions like penalties management, incident management, report generation, etc.

2. THEORY

2.1 JAVA EE 5

Java SE (Java Platform Standard Edition) is known well by most Java programmers, it is the platform used to develop different kinds of Java desktop applications. The difference between Java SE and Java EE (Java 2 Platform, Enterprise Edition) is that Java EE adds libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server. It is easy for developing enterprise applications, which often includes a web application. In fact Java EE is a specification of rules which programmers must obey when implementing enterprise software. The goal of Java EE is to be platform-independent, secure and standardized. It is also meant for server-side development. [1]

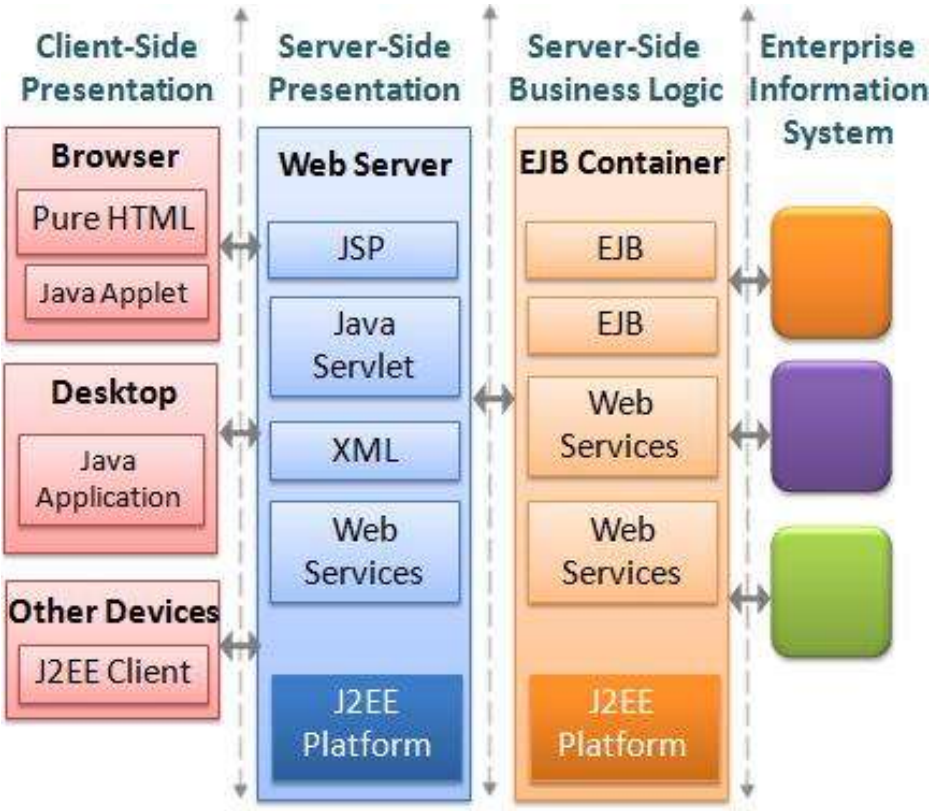

Figure 2-1 J2EE Architecture Diagram

Java EE includes several API specifications, such as JDBC, RMI, e-mail, JMS, web services, XML, etc., and defines how to coordinate them. Java EE also features some specifications unique to Java EE for components. These include Enterprise JavaBeans, Connectors, servlets, portlets (following the Java Portlet specification), JavaServer Pages and several web service

technologies. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies. A Java EE application server can handle transactions, security, scalability, concurrency and management of the components that are deployed to it, in order to enable developers to concentrate more on the business logic of the components rather than on infrastructure and integration tasks.

A brief history of Java EE is that when the need for dynamic web pages increased, the Java Community Process developed the J2EE platform, which was built on the foundation of Java SE, but added functionalities for all the interaction between a client and a server among other things.

The first version of Java 2 Platform Enterprise Edition was called J2EE 1.2 and this was followed by version 1.3 and 1.4 beta, but along with the next version came a name change from J2EE 1.5 to Java EE 5. The purpose of renamed is to let everyone know that J2EE is just for enterprise applications and the Java Community Process experts wanted to simplify the name as well. Then the platform had gone through several changes (the current version is Java EE 6) which made web application development even easier.

2.1.1 BROWSER/SERVER

The most important thing to understand about web applications is that they are based on client and server comunication. A client can be a web browser or an application that uses a remote interface to communicate with a server, but to web applications the web browser client is the most important one, so we‘re going to focus on that.

B/S structure is stand for Browser/Server structure which is the improvement of C/S (Client/Server). Web Browser becomes the main application software on the client side. This model unifies the client, the main business logic implement on the server side, very small part of the business logic achieved in Browser. We can just install a browser like Netscape Navigator or Internet Explorer on the client, on server install Oracle, My SQL or other databases. Browser exchanges data with database through Web Server. In this way it greatly simplifies the load of client‘s computer, reduces the cost of system maintenance and upgrade.

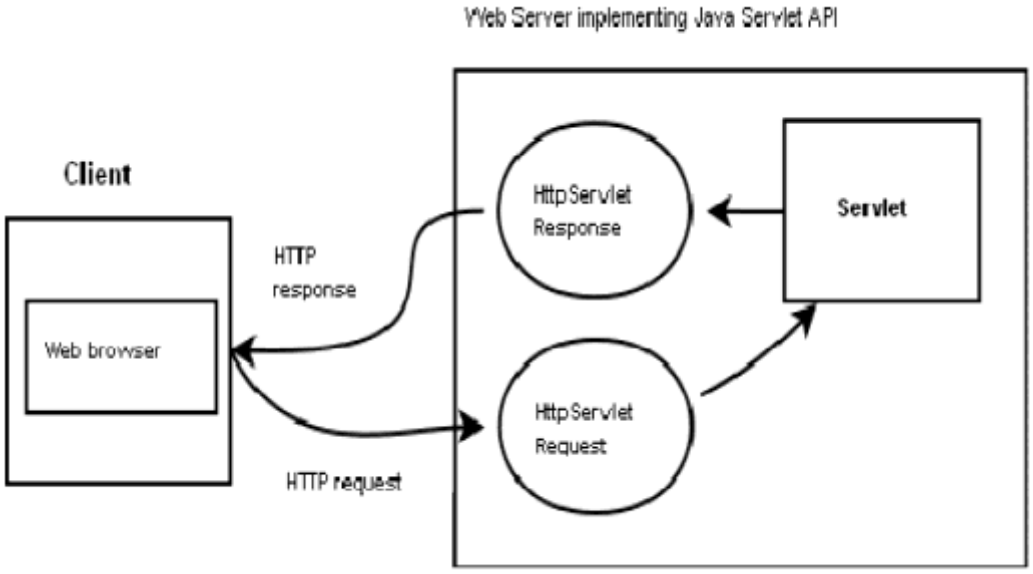

Figure 2-2 how the request-response cycle between a client and a server looks

The interaction between the client and the server is made by the client (the web browser) sending an HTTP(HyperText Transfer Protocol) request to the server. (See Figure 2-2) The request is like a message sent over a network and the server which implements the needed technology converts it to an HttpServletRequest (if it's using the Java Servlet API). This HttpServletRequest is received by a web component which executes on the server. Two very common web components are servlets and jsp pages. A common scenario is that a servlet receives the HttpServletRequest, constructs an HttpServletResponse and lets the server convert it to an HTTP response which is sent back to the client. Another common scenario is that the servlet passes the request on to another web component, but eventually an HttpServletResponse will be generated by some web component and be sent back to the client. [2]

Web components execute in a Java EE runtime environment referred to as a web container. It provides the web components with different kinds of web services. There are three other types of containers that are used in a Java EE application; an EJB container, Application client container and Applet container. In enterprise applications often a database is involved, which all the containers but the Applet container has access to through an API called Java Persistence API. (The applet container has indirect access to it through the web container.)

2.1.2 SERVLET

A Servlet is just a Java class in Java EE that conforms to the Java Servlet API, a protocol by which a Java class may respond to requests. [3] They are not tied to a specific client-server protocol, but are most often used with the HTTP protocol. Therefore, the word "Servlet" is often used in the meaning of "HTTP Servlet". Thus, a software developer may use a servlet to add dynamic content to a Web server using the Java platform. The generated content is commonly HTML, but may be other data such as XML. Servlet can easily construct an HTML page from the Java code, which the server can send to the client as a response to a request. The created page will immediately be shown to the user.

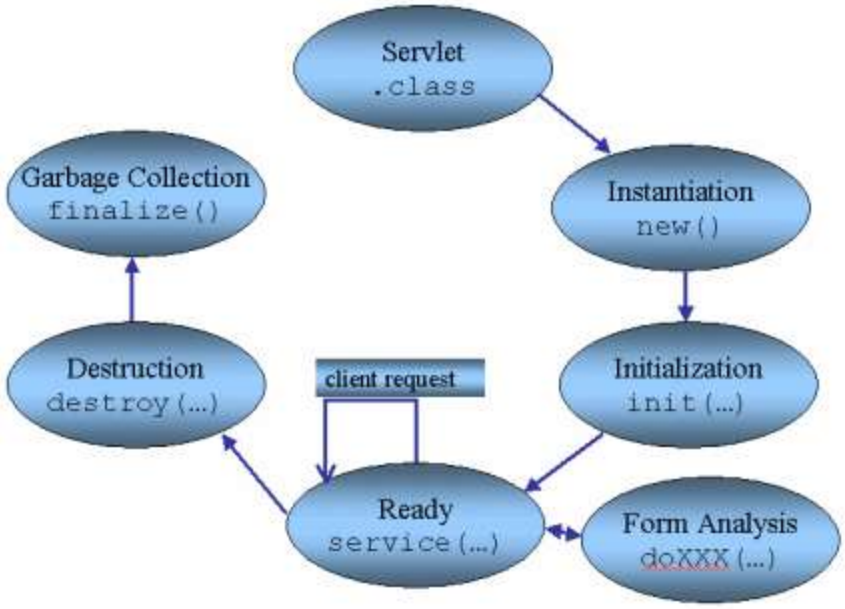

Figure 2-3 the lifecycle of a Servlet

Java Servlet technology provides Web developers with a simple, consistent mechanism for extending the functionality of a Web server and for accessing existing business systems. A servlet can almost be thought of as an applet that runs on the server side--without a face. Java servlet makes many Web applications possible. It has access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

In the most common implementation model each servlet only exists as one instance, running on the server. This instance can handle multiple threads and multiple HTTP sessions. A session is ~~"a~~ series of related browser requests that come from the same client during a certain time period." It can be used for saving values between the calls, like a stateful session bean.

Usually there‘s one session per user so the session can be used by the server to be able to identify different users. A session no longer exists when either it‘s been explicitly invalidated or when it has expired. In Java EE the session is represented by an HttpSession object.

Today Servlet is a popular choice for building interactive Web applications. Servlet containers are usually a component of Web and application servers, such as BEA WebLogic Application Server, IBM WebSphere, Sun Java System Web Server, Sun Java System Application Server, and others.

2.1.3 JSP

Java Server Pages is basically just another way to write a servlet. In fact JSPs are compiled into servlets. The difference is that servlets are made from Java code, with the possibility of writing HTML inside it, but JSPs are often made from HTML code, with the possibility of inserting some Java code into it.

A jsp page could look exactly like an HTML page, but it can also contain XML (Extensible Markup Language), SVG (Scalable Vector Graphics), WML (Wireless Markup Language) and JSP elements. It makes pages easier to combine fixed or static template data with dynamic content. JSP elements are different types of code which creates dynamic content. It can be a tag library (some predefined code which can be used through tags) like Java Server Pages Standard Tag library or a JSP Scripting element (called –scriptlet”), which is just a way to embed regular Java code into the JSP.

Starting with version 1.2 of the JSP specification, Java Server Pages have been developed under the Java Community Process. Today, JSP has been widely used around the world. It was improved more humane, more functions. Although there are many competitors like ASP (Active Server Page) or Pure Servlets, its position has not been replaced.

2.2 Framework

In terms of computer programming, a software framework is an abstraction layer in which common code providing generic functionality can be selectively overridden or specialized by developer code, thus providing specific functionality. Frameworks are a special kind of software or coding libraries, that differ from them in that they are reusable abstractions of

code wrapped in a well-defined application programming interfaces (API). They also contain some key distinguishing features that separate them from normal libraries.

Software frameworks can be seen as typical libraries or even normal user applications. But these distinguishing features separate them from libraries or normal user applications:

- Inversion of control – in framework, unlike in libraries or normal user applications, the overall program's flow of control is dictated by the framework, not by the caller.

- Default behavior – a framework has a default behavior and this behavior must actually be useful, i.e. must be process implementation-oriented or a working solution; framework can‘t be just a series of no-ops or pack of procedures or functions with no straightforward defined application as whole.

- Extensibility – a framework can be extended by the user usually by selective overriding or specialized by user code providing specific functionality.

- Non-modifiable framework code – the framework code, in general, is not allowed to be modified; users can extend the framework code, by writing own classes inheriting from classes introduced by framework, but can‘t modify framework code.

There are different types of software frameworks: conceptual, application, domain, platform, component, service, development, etc.

The designers of software frameworks aim to facilitate software development by allowing designers and programmers to devote their time to implementing project requirements rather than dealing with the more standard low-level details of providing a working system, thereby reducing overall development time. For example, a team using a web application framework to develop a banking web-site can focus on the operations of for example account withdrawals and user management rather than on the mechanics of user request handling, user authentication and authorization processes, browser-compatibility issues, etc. These, so called low-level operations are all handled by framework.

Frameworks represent the cumulated experience of how the software architecture and its implementation for most applications should look like and knowledge of many developers for solving the most common problems. On the other hand, it leaves enough room for customization to solve some particular problems in the application.

Frameworks are of key importance for developing large-scale object-oriented software systems. They promise higher productivity and shorter time-to-market through design and

code reuse. However, many projects report that this promise is hard to fulfill. And furthermore for some project or solutions traditional and procedural approach to development might turn out to be more efficient or less troubled in implementation despite of all advantages brought by framework programming.

2.3 STRUTS

The Apache Struts web framework is a free open-source solution for creating Java web applications. [4] It uses the Model-View-Controller (MVC) design pattern. (See Figure 2-4) The pattern divides the software system into three basic parts which are Model, View and Controller. The controller is responsible for forwarding the request, and deals with interactions between Model and View. The View is used to interface design for displaying the data, like JSP page. The Model encapsulates business logic and data processing method, and can direct operate the data, such as access to the database.

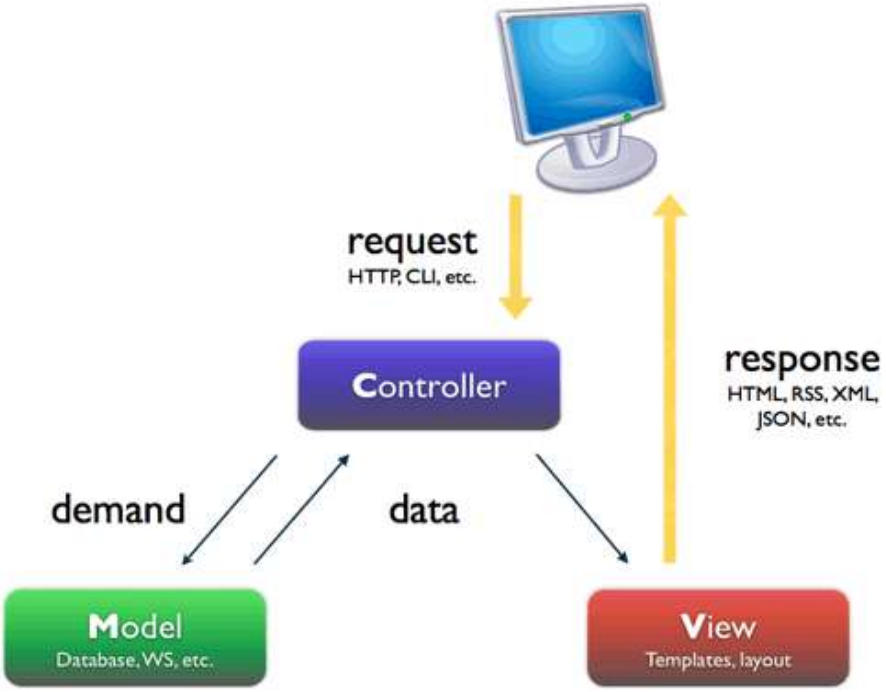

Figure 2-4 MVC pattern in general

There are two versions of Struts (1 and 2), but because of their large structural changes, we only consider the second version which was used in this paper, Struts 2.

2.3.1 STRUTS 2

Struts 2 is the second generation product of Struts. It is the result of a merger between Struts 1 and another framework called WebWork. Struts 2 selects WebWork as the core, using interceptor mechanism to deal with the user‘s request, this design also make the controller of business logic completely divorced from Servlet API, so Struts 2 can be understood as the update product of WebWork.

The high-level design of Struts 2 follows the well-established Model-View-Controller design pattern. The MVC design pattern identifies three distinct concerns: model, view, and controller. In Struts 2, these are implemented by the action, result, and FilterDispatcher, respectively. (See Figure 2-5)

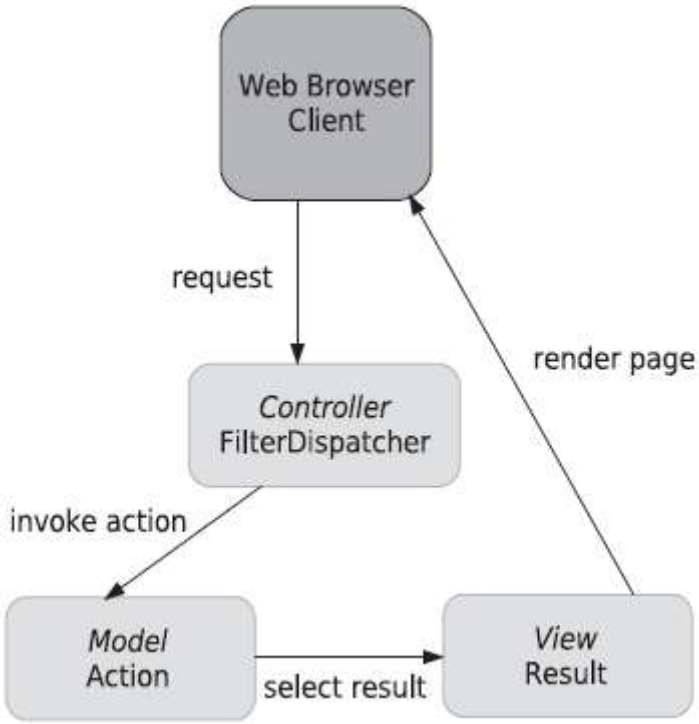

Figure 2-5 three core components of Struts 2 MVC

Controller – FilterDispatcher

We‘ll start with the controller because it‘s the first component to be called when a HTTP request has been sent by the browser client. The roll of the controller is played by the Struts 2 FilterDispatcher. The FilterDispatcher is a servlet filter (a part of the Java Servlet API which can be transparently added to a web application to perform operations against the servlet

request or servlet response. [5]), and its purpose is to map each incoming request against an appropriate action.

Model - Action

Looking at figure 5, it's easy to see that the model is implemented by the Struts 2 action component. It is actually an ordinary Java class which often extends other classes or implements interfaces belonging to the Struts 2 framework. It is both a place to put business logic (directly or indirectly through calls to other objects) and a place to save data. (These two are often called the "state" of the application.)

The developer implements the action classes and decides which features he wants to add depending on the requirements of the business logic. If the developer wants to, the action can be made as simple as any Java class which implements a method named "Execute". This simplicity comes with lots of "behind the scenes things" happening in Struts 2.

A very useful feature of Struts 2 is its error handling. There's actually a whole framework for this contained in Struts 2. When it comes to errors related to certain form fields (like validation errors and type conversion errors) they can automatically be shown in the View page, as long as the developer has mapped different error types to different fields , (either in an XML file or in a method in the action) and is using the Struts 2 tag library. When the validation fails the String "input" is automatically returned from the action. If this String is mapped against a certain page Struts 2 automatically calls that page too.

View - Result

The view is the presentation component of the MVC pattern. Looking back at figure 5, we see that the result returns the page to the web browser. This page is the user interface that presents a representation of the application's state to the user. These are commonly JSP pages, Velocity templates, or some other presentation-layer technology. While there are many choices for the view, the role of the view is clear-cut: it translates the state of the application into a visual presentation with which the user can interact. For the View pages, Struts 2 provides a big variety of tags. Three common types of tags are data tags, control-flow tags and User Interface (UI) tags. [6] The data tags can be used for creating instances of objects and putting them on the Value Stack among other things. Among the control-flow tags you can find things like an "if" tag which is useful for conditional expressions. The UI tags generate

HTML markup code. It can for instance generate a ―select" tag including the underlying ―option" tags from a Struts 2 UI tag with only one line.

From figure 5, you can see that the action is responsible for choosing which result will render the response. The action can choose from any number of results. Common choices are between results that represent the semantic outcomes of the action‘s processing, such as ―success" and ―error."

2.3.2 Components

Apart from the MVC components Struts 2 has a few other important components worth mentioning.

2.3.2.1 Interceptors

A very important feature of Struts 2 is the interceptors. An interceptor is a reusable component which can be used for separating things like validation and logging (things that don‘t really belong to the business logic) from the action code.

The interceptors are invoked before and after the action for every action that uses them. There‘s a standard stack of interceptors that‘s included in Struts 2 and it‘s commonly used by the actions. To use this standard stack the developer just has to extend one class (ActionSupport) in each action class.

Some common issues that are handled by interceptors are: validation, data transfer (that‘s how the form values are moved into the action), logging, injecting objects from the Servlet API (like HttpServletRequest) into the action through setters, uploading files and exception handling.

2.3.2.2 The ActionContext and the Value Stack

The ActionContext is a ThreadLocal component which contains all the data about the context of the current action. It has access to the attributes saved in the HttpServletRequest,

HttpSession and the data saved in Application scope. But the most important part of the ActionContext is the Value Stack.

The Value Stack is a storage space for all the application data that belongs to the current request. The current action is on this stack, and while it is, its properties are accessible like they were properties of the Value Stack itself. The params interceptor (See –Interceptors") puts the action on the stack and is also responsible for putting request parameter values onto the matching action properties before the action is invoked. This is a very important feature of Struts 2.

2.3.2.3 OGNL

Another important thing to know about Struts 2 is OGNL (Object-Graph Navigation Language). This is an expression language used in the View pages for connecting form data to data on the action (through the Value Stack). For example it's used for setting the name of a form parameter to a value that matches a certain property of the action being called. But since all the form data are of the type String some type conversions need to be made as well. This is also done automatically by OGNL.

It's also possible to use OGNL for setting or getting attributes from the HttpSession or HttpServletRequest, from the JSP.

2.3.2.4 Themes

Struts 2 uses themes by default. A theme is a collection of a special kind of files called templates. These templates are written in the FreeMarker language and deal with the UI tags of Struts 2 (mentioned under –The Result"). Each template generates one UI tag as HTML (if HTML is the chosen View technology).

The themes are important since they control the layout of the rendered HTML pages, but for developing a web application with simple views (where everything is rendered in two columns) one can just use the default xhtml theme. To do this all that's needed is to include this tag in the JSP page:

<s:head/>

Then, for instance the error messages of the application will automatically be red and placed in an appropriate place.

2.3.2.5 Localization

Struts 2 also supports localization (the possibility for the user to switch the language of the application). All the developer has to do to prepare the different languages is to split all the text that‘s shown in the JSPs into rows or words where every row or word has a key (like an id) and the actual text. Then all these rows are put inside a .properties file. This is an example of a row in such a file (the key is to the left and the value to the right):

initialize.user.inexistant = Cet utilisateur n"existe pas.

The point is to have several properties files – one per language – where the keys are constant and just the values (the text to be shown) differ between the files. As long as Struts 2 can find the location of these files (there are several ways to do this), the developer just has to use the ―key‖ attribute on the Struts 2 tags to show its value. This makes the text dynamic (considering the chosen language) instead of static

2.3.3 How Struts 2 works

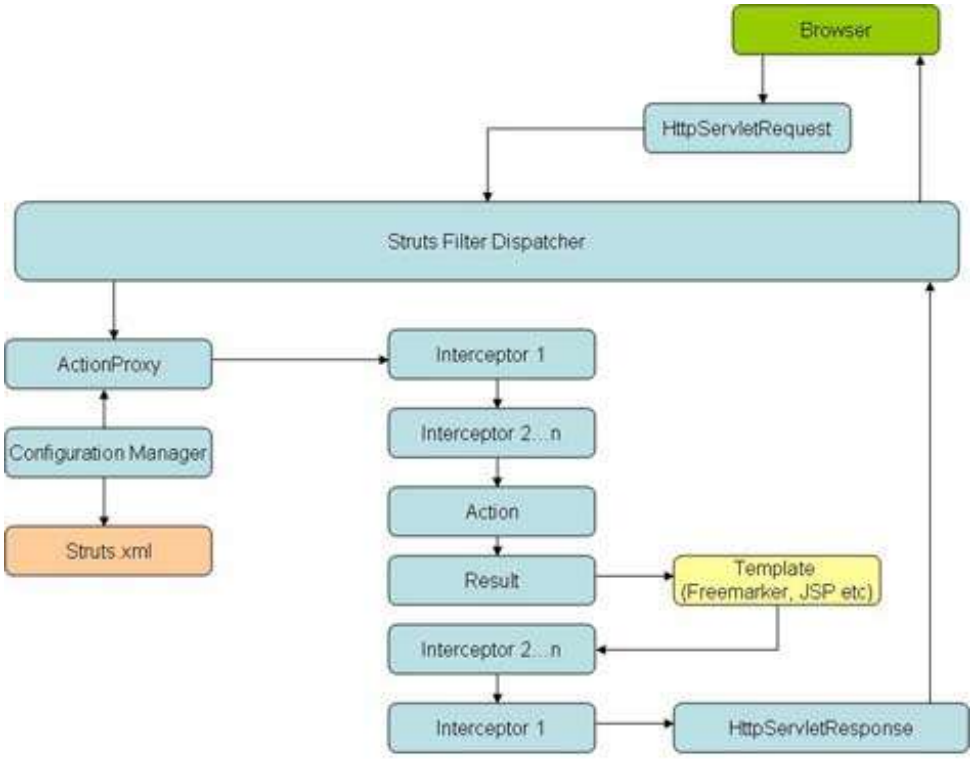

Figure 2-6 Struts 2 request processing

We can see the general processing of Struts 2 in Figure 2-6. First, the Browser sends a request to the Filter Dispatcher which can map this request to an appropriate action. The interceptors that implement common concerns across actions are then called (in the before() method) in advance of invoking the Action itself. Interceptors built into Struts 2 can perform core processing, like populating request parameters into Action classes, performing validation, uploading files, and so on. You can also define custom interceptors as you want. The Action class typically invokes the business layer and populates the model objects, which are instances variables of the Action class. Next, the request is dispatched to the view layer (which is built on a technology like JavaServer Pages, FreeMarker, or Velocity), which renders the GUI. The interceptors are executed again (in reverse order, calling the after() method). Finally, the response returns through the Filter Dispatcher chain.

Of course, Struts 2 framework has more than just its MVC components, and also has a few other important components which are interceptors, OGNL (Object-Graph Navigation Language), and the Value Stack. These components interact together to implement a cleaner MVC design.

2.4 HIBERNATE

A very common issue in Java development today is how to map the object oriented (OO) Java code against a relational database. This is called object/relational mapping (ORM) and Hibernate is a framework made for providing good solutions to this problem. It‘s a so called persistence framework.

Hibernate also uses the Java EE 5 APIs JDBC (Java DataBase Connectivity), JPA (Java Persistence API), JTA (Java Transaction API) and JNDI (Java Naming and Directory Interface). [7]

Hibernate consists basically of four parts:

- Persistent objects
- ORM (Object/Relational Mapping) mapping files
- Interfaces called by Java applications to perform CRUD (Create – Read – Update - Delete) operations on the persistent classes
- Interfaces used for Hibernate Configuration

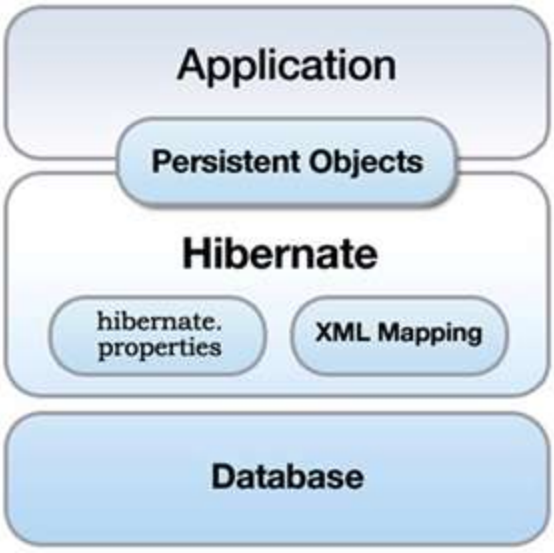

Figure 2-7 Hibernate Architecture

2.4.1 Persistent Objects

A persistent object in Java is just a POJO (Plain Old Java Object) class, which means that it only has an empty property and constructor access methods (“getters” and “setters”) for the properties of the class. It handles the data existing physically regardless of application execution. [8] Generally, when developing an application using DBMS data, the business layer of the application handles the application data via SQL for the specific DBMS. But Hibernate-based applications can integrate application data and DBMS with persistent object as the main.

Persistent classes shouldn‘t contain calls to objects in the Hibernate API. Instead this can be done by a DAO (Data Access Object) or directly (without any help classes) in a small application. The reason for this is that the persistent classes should be able to be used in “normal” Java applications as well as for persistence.

2.4.2 ORM Mapping Files

ORM Mapping files in Hibernate is the xml file like *.hbm.xml. All the mapping information lies in these files, tt responsible for mapping the persistent classes against the database tables. Hibernate creates SQL to be executed based on Hibernate Mapping XML. In this case the code of the persistent classes becomes “clean” and just like a POJO.

There are usually one Hibernate mapping file per persistent Java class. The mapping file maps the id property of the Java class against a Primary Key in the database, all the other properties against table fields and maps relations (of all types) to other persistent classes. Most of these mappings are easy to understand (like the “id” and “property” tags). The hardest part is mapping more complicated relationships (like unidirectional one-to-many which was used in this application) and composite ids.

2.4.3 Interfaces called by Java applications to perform CRUD operations on the persistent classes

We know the operation for database can be divided into create the data, read the data, update the data and delete the data. Hibernate provide some interfaces which can easy perform these operations. Four important interfaces that Java developers need to use are SessionFactory, Session, Transaction and Query.

The SessionFactory is used to create Sessions, there is usually only one SessionFactory per application.

The Session is the key object because it has methods for handling all the CRUD operations. It is an object performing a connection between Hibernate and DB connection, which maintains connection until the session is closed after opening a single DB connection on session development. As all the objects (persistent objects) loaded by Hibernate is related with session, the object changes are automatically reflected by session or handled with lazy loading. Session can also create and return a Transaction or Query object. It is not thread safe so make sure that only one session per thread exists.

The transaction can be used for separating different units of persistence operations, and it must be handled ―manually".

The hibernate query is used to easy handling of queries against the database. Through use of a query object the user is relieved from having to write SQL code and can instead just take the name of the class and the id value of an object.

2.4.4 Interfaces used for Hibernate Configuration

An instance of the Configuration object must be created first when you build a SessionFactory. This object is responsible for the initialization and configuration of Hibernate, but it needs to know the values of certain paths and properties to the mapping files or annotated classes (if that approach is used). There are two different ways to set these paths and values.

The first way is to use an XML configuration file. This configuration file must called hibernate.cfg.xml and is on the classpath, so Hibernate can find it automatically. In this file you also can specify the mapping files that are used along with all the properties concerning the database connection.

The second way is to use a file which must be called hibernate.properties and must be on the classpath, but this can only be used for the database connection properties. The mapping files must still be specified in the hibernate.cfg.xml file (it's possible to have both).

2.5 EJB 3

EJB 3 stands for Enterprise JavaBeans version 3. It is the server-side component architecture for Java Platform Enterprise Edition (Java EE). The initial goal for EJB was to provide a simpler alternative to CORBA through components and framework benefits. Since the earlier versions e.g. 2.1 were considered a bit too complex, some major changes were made in version 3.0 that made it a lot easier to use this technology.

An EJB is just a POJO (Plain Old Java Object) with the addition of something called annotations. [9] An annotation starts with ‗@' and is followed by a specific word. The combination of the ‗@' and the word means that the EJB gets access to a specific service that belongs to the environment the bean resides in.

There are three different types of EJBs: Session beans, Message-driven beans and Entities.

2.5.1 Session beans and Message-driven beans

Both of Session bean and Message-driven beans are managed by an EJB container which can provides services like security, portability and automatic transaction handling. The container is also responsible for handling the creation and destruction of beans so the EJB client never has to do that. When the container needs to create a bean instance, it will inject the bean with the needed resources and places it in a bean pool. Several bean instances which kept in the pool are ready for executing methods, at last, they will be destroyed by the container when a bean instance is no longer needed it.

Session beans is used to implement business logic, it include two types which are Stateless and Stateful. Stateless beans does not save any state between different calls by the client, which means that the same user can use two different bean instances when using different kinds of services on the same web page. It can be performed like ―@Stateless" in Java code. Stateful beans save the state of the bean between different invocations by a client, and the container makes sure that the same client uses the same bean instance as long as the session lasts. A stateful bean could be useful for handling for instance a shopping cart for a web merchant. It can be performed like ―@Stateful" in Java code.

The purpose of session beans is to directly respond to method calls, but message-driven beans (MDBs) indirectly handle receiving of messages. The message-driven bean uses an API called Java Messaging Service (JMS) which provides reliable, thread-safe and asynchronous messaging by connecting to Message Oriented Middleware (MOM) software. MOM acts like a ―middleman" between the producer and the consumer (receiver) of the message and makes sure that the message will be delivered even if the consumer is not available when the message is being sent. [9]

2.5.2 Entities

Like Session beans and MDBs, an entity is just a POJO with the addition of some annotations. In general, an entity is basically the Object Oriented equivalent of a relational database table. It is used to implement the Object/Relational Mapping and responsible for mapping the database table records to the entity object in memory, in fact, create a new Entity Bean object is equivalent to create a table record, delete a Entity Bean will also delete the corresponding record from the database, when modify a Entity Bean, the container will automatically synchronize the state of Entity Bean with the database. One row in the database table corresponds to one entity instance. But since it's a POJO you can also have business logic in your entities if preferred.

2.6 SPRING

The Spring Framework is an open source application framework for the Java platform. The core features of the Spring Framework can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the Spring Framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBean (EJB) model. [10]

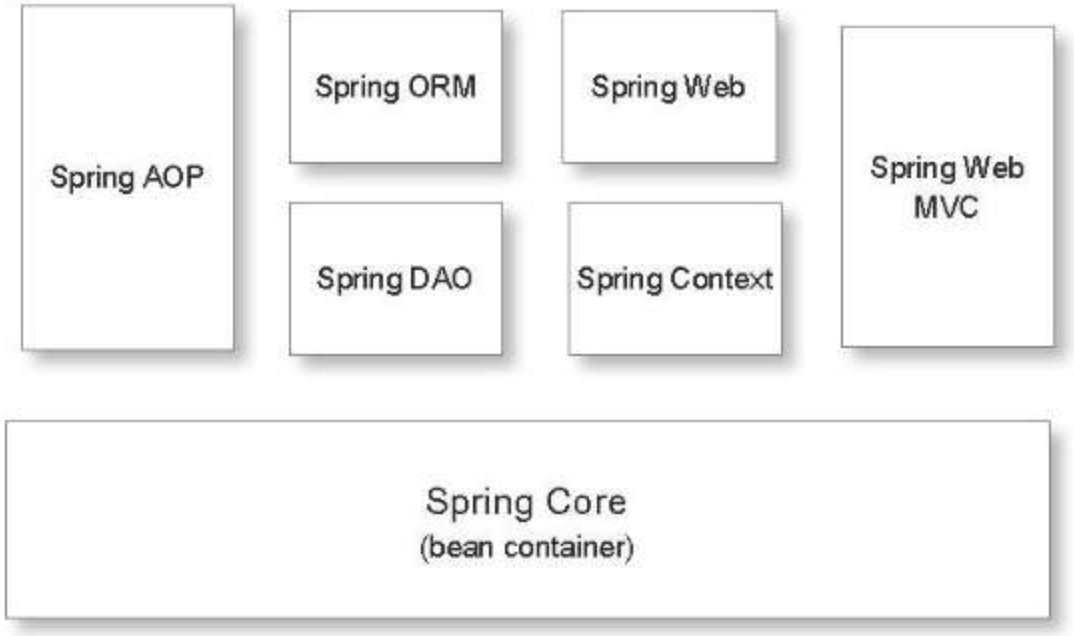

Figure 2-8 Spring Framework

The Spring Framework comprises several modules that provide a range of services (See Figure 2-8):

- **Spring Core**: provide basic functions of Spring framework. The main component is BeanFactory which is an implementation of the factory pattern that applies Inversion of Control (IOC) principle to promote loose coupling and separate your application's configuration and dependency specifications from the actual application code.

- **Spring Context**: is a configuration file which provides context information to the Spring framework. Spring context include enterprise service, such as JNDI, EJB, Email, Internationalization, validation, and scheduling functions.

- **Spring AOP**: stands for Spring Aspect-oriented programming. This module directly integrates Aspect-oriented programming into Spring framework so it can easily let any object support AOP. It also provides transaction management services, through Spring AOP, the declarative transaction management can be integrated into the application without relying on EJB components.

- **Spring DAO (Data Access Objects)**: as a general database access functions, this module also support JDBC interface.

- **Spring ORM**: Spring framework support several ORM frameworks, such as JDO, Hibernate and iBatis SQL Map.

- **Spring Web**: Web context module is based on application context module, it support integration with Jakarta Struts. Web module also simplifies the task for handling multi-part requests and binding request parameters to domain objects.

- **Spring MVC**: Spring MVC contains many view technologies, like JSP, Velcity, Tiles, iText and POI.

Central to the Spring Framework is its Inversion of Control container, which provides a consistent means of configuring and managing Java objects using reflection. The container is responsible for managing object lifecycles: creating objects, calling initialization methods, and configuring objects by wiring them together.

Objects created by the container are also called Managed Objects or Beans. Typically, the container is configured by loading XML files containing Bean definitions which provide the information required to create the beans.

Objects can be obtained by means of Dependency lookup or Dependency injection. Dependency lookup is a pattern where a caller asks the container object for an object with a specific name or of a specific type. Dependency injection is a pattern where the container passes objects by name to other objects, via either constructors, properties, or factory methods.

Dependency Injection has several important benefits. For example:

1. Because components don't need to look up collaborators at runtime, they're much simpler to write and maintain. In Spring's version of IoC, components express their dependency on other components via exposing JavaBean setter methods or through constructor arguments. The EJB equivalent would be a JNDI lookup, which requires the developer to write code that makes environmental assumptions.

2. For the same reasons, application code is much easier to test. For example, JavaBean properties are simple, core Java and easy to test: simply write a self-contained JUnit test method that creates the object and sets the relevant properties.
3. A good IoC implementation preserves strong typing. If you need to use a generic factory to look up collaborators, you have to cast the results to the desired type. This isn't a major problem, but it is inelegant. With IoC you express strongly typed dependencies in your code and the framework is responsible for type casts. This means that type mismatches will be raised as errors when the framework configures the application; you don't have to worry about class cast exceptions in your code.
4. Dependencies are explicit. For example, if an application class tries to load a properties file or connect to a database on instantiation, the environmental assumptions may not be obvious without reading the code (complicating testing and reducing deployment flexibility). With a Dependency Injection approach, dependencies are explicit, and evident in constructor or JavaBean properties.
5. Most business objects don't depend on IoC container APIs. This makes it easy to use legacy code, and easy to use objects either inside or outside the IoC container. We say that an IoC container isn't **invasive**: using it won't invade your code with dependency on its APIs. Almost any POJO can become a component in a Spring bean factory. Existing JavaBeans or objects with multi-argument constructors work particularly well, but Spring also provides unique support for instantiating objects from static factory methods or even methods on other objects managed by the IoC container. [11]

This last point deserves emphasis. Dependency Injection is unlike traditional container architectures, such as EJB, in this minimization of dependency of application code on container. This means that your business objects can potentially be run in different Dependency Injection frameworks - or outside any framework - without code changes.

In many cases one need not use the container when using other parts of the Spring Framework, although using it will likely make an application easier to configure and customize. The Spring container provides a consistent mechanism to configure applications and integrates with almost all Java environments, from small-scale applications to large enterprise applications.

The Spring Framework has its own AOP (Aspect-oriented programming) framework which modularizes cross-cutting concerns in aspects. The motivation for creating a separate AOP framework comes from the belief that it would be possible to provide basic AOP features without too much complexity in either design, implementation, or configuration. The Spring AOP framework also takes full advantage of the Spring Container.

The Spring AOP framework is interception based, and is configured at run time. This removes the need for a compilation step or load-time weaving. On the other hand, interception only allows for public method-execution on existing objects at a join point. [12]

3. Two cases' design and solution

3.1 Specification of the system

3.1.1 Introduction

In order to help GCE Technologies manage their control, we need to develop a system which should have many functions like penalty management, incident management, report generation, etc. The system had been poorly developed before, using php, mysql and jquery. The PHP development was done without any structure in the code, many things run hard and even has a few bugs. The system is less scalable and difficult to maintain. To commercialize this system, we decide to rewrite it in JAVA/J2EE, Struts, Hibernate, and MySQL. The goal is to get a clean and healthy code, structured and more efficient than the current software. We will have scalable software that can develop customized for the next client.

There were five people in Groupe HN to develop this system. Our manager jimmy met customers every week, and then assigned the tasks for us. It almost spent 5 month to finish this system.

The system has three roles which are administrator, manager and employees. The administrator is responsible for maintaining the system, and solves the unexpected problems for other roles when they use the system. The manager can issue some orders or release some informations to employees, and the employee is able to ask questions, view informations etc.

3.1.2 Architecture

The architecture that was used in this project is the well known four-tier layered architecture. Layered architectures have become the dominant design pattern when it comes to server-side development. This means that all involved components are divided into different tiers or layers, where each tier has a special responsibility. These are the four tiers of the four-tier layered architecture (Depicted in Figure 3-1):

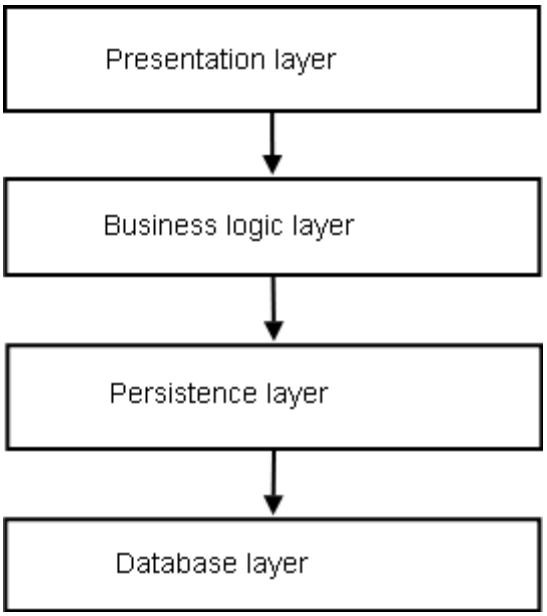

Figure 3-1 the four layers architecture

The layer at the lowest level is the database layer, which can be for instance a MySQL or Oracle database.

The persistence layer is an architectural layer between the business logic layer and the database whose job is to provide an abstract interface to information storage mechanism. An API to such an interface should be abstract, and independent of storage technologies and vendors. It would typically have features such as the following:

- Store / Retrieve of whole object networks by key.
- Logical database cursor abstraction for accessing some / all instances of a given type.
- Transaction support - open, commit and abort.
- Session management.
- Some basic querying support, for example, how many instances of a given type exist etc.

The business logic layer is the most important part of the application. This is where the logical parts that make up the application are. They consist of distinct actions that do a bit more than just receiving a value, and this is where EJBs belong. For interaction with a database, this layer uses the persistence layer.

The Presentation layer deals with all the user interaction. That means handling and rendering the components that are visible to the user (like HTML and JSP pages), taking care of input data from the user and passing it on to the next layer, which is the Business logic layer.

This choice of four layers architecture can have many beneficial effects on the application if it is applied in the proper way. First, since the architecture is so simple, it is easy to explain to team members and so demonstrate where each object's role fits into. Second, this architecture makes the project easy to divide work along layer boundaries. Finally, a four-layer architecture makes it possible to code the bulk of your system (in the business logic layer and the persistence layer) to be independent of the choice of persistence mechanism and windowing system.

3.1.3 Project Overview

This is a medium enterprise project. The whole system can be divided into several modules, such as incident management, budget management, user management or penalty management.

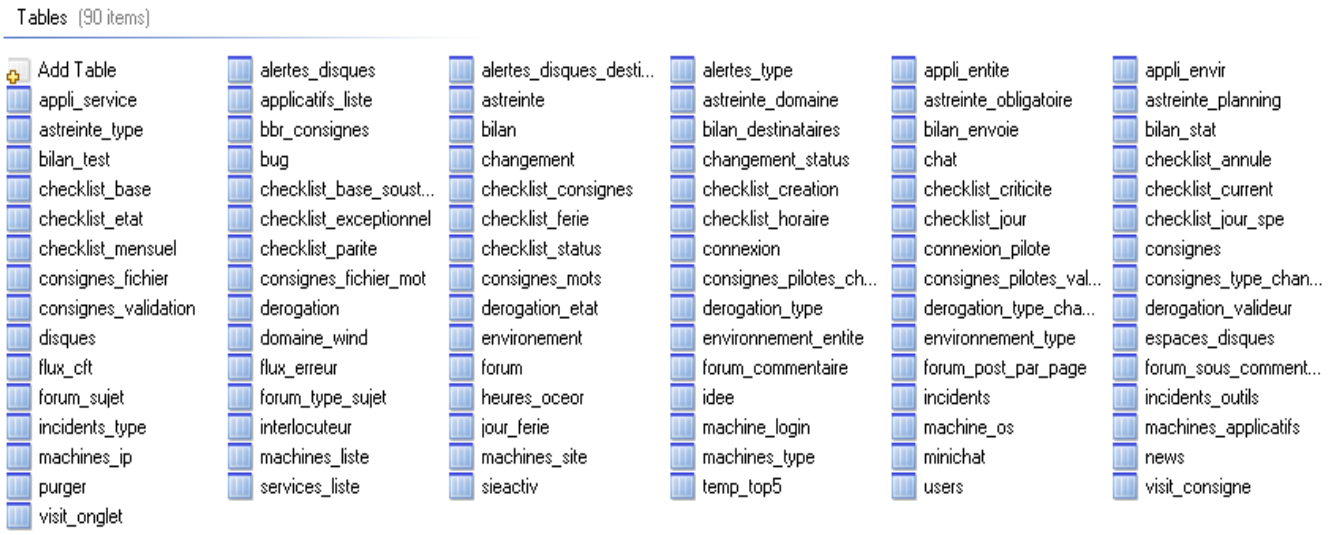

Figure 3-2 database tables

The project has about 100 tables (See Figure 3-2), most of the functionality for each module are based on the data‘s operation which are create, update, read and delete operation. For example, in instruction management module, when we click the “consigne” icon on the left menu, the system will list all the instructions. (See Figure 3-3)

Figure 3-3 list all instructions

Of course, all these data are selected from table "consignes" in database, and we also can create a new instruction when click "ajouter" icon, (Figure 3-4) update an instruction, (Figure 3-5) and delete an instruction. (Figure 3-6)

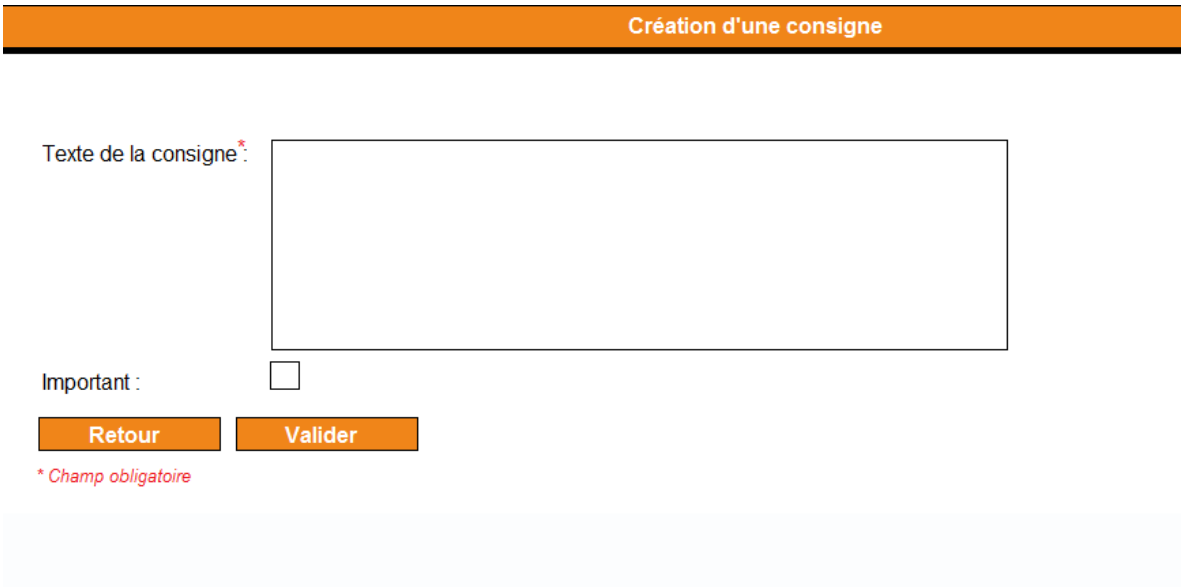

Figure 3-4 create a new instruction

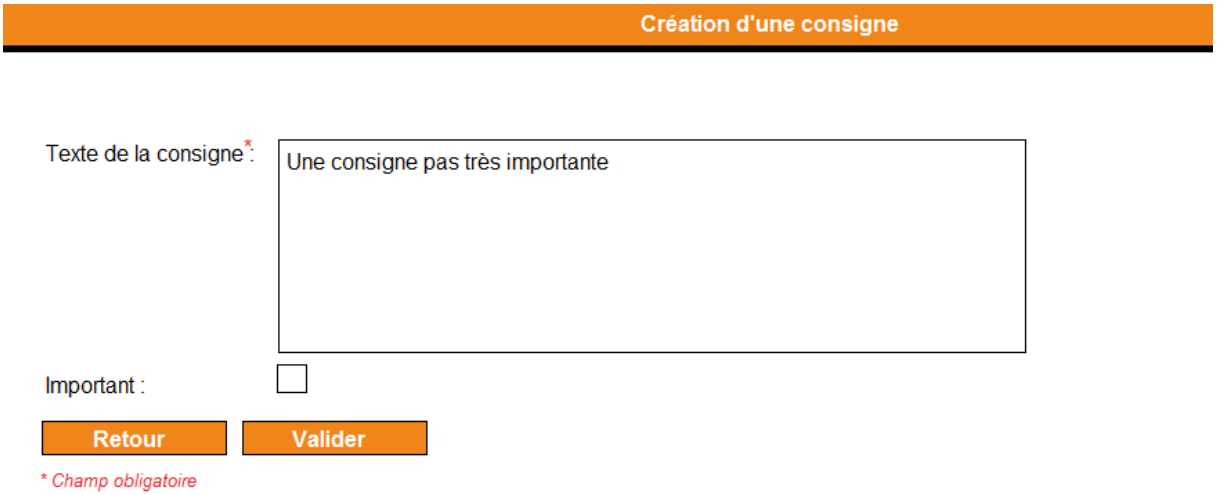

Figure 3-5 update an instruction

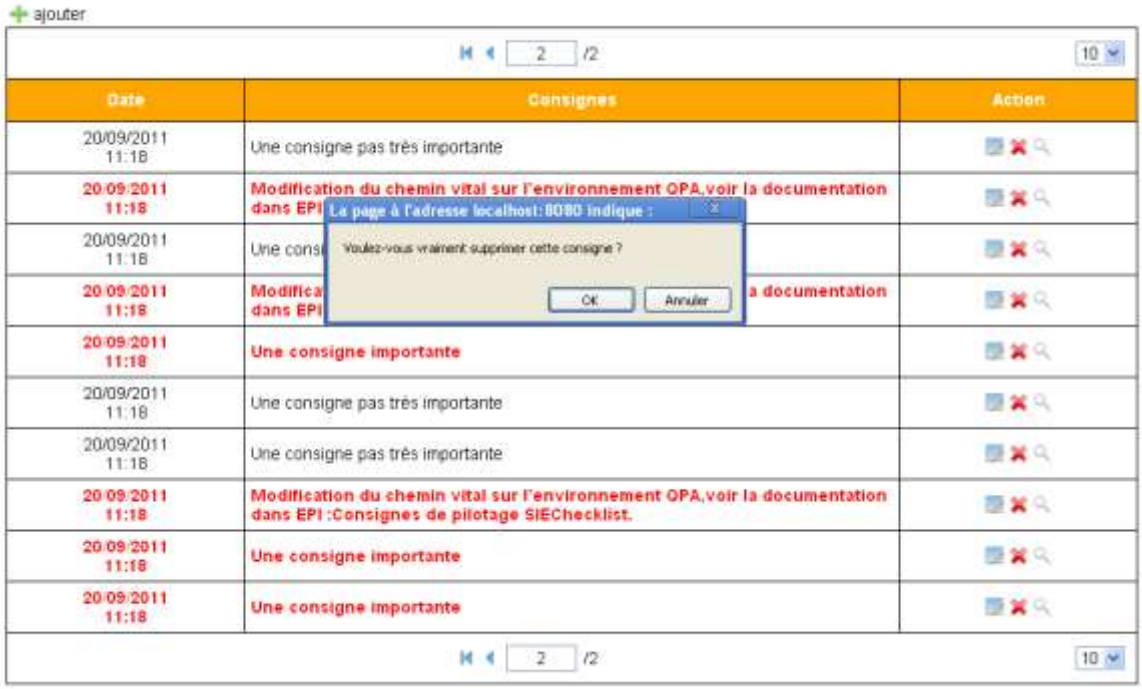

Figure 3-6 delete an instruction

For each module, the method for operating the database will be different. Some methods could be very simple, such as query all the data from the table, and some others could be very complicated, for example, add a paging or sort to the data. This should according to the requirement of customers.

Besides, there are also some functions like send a email to the administrator, download files from server side or export the data to the excel. (See Figure 3-7 and 3-8)

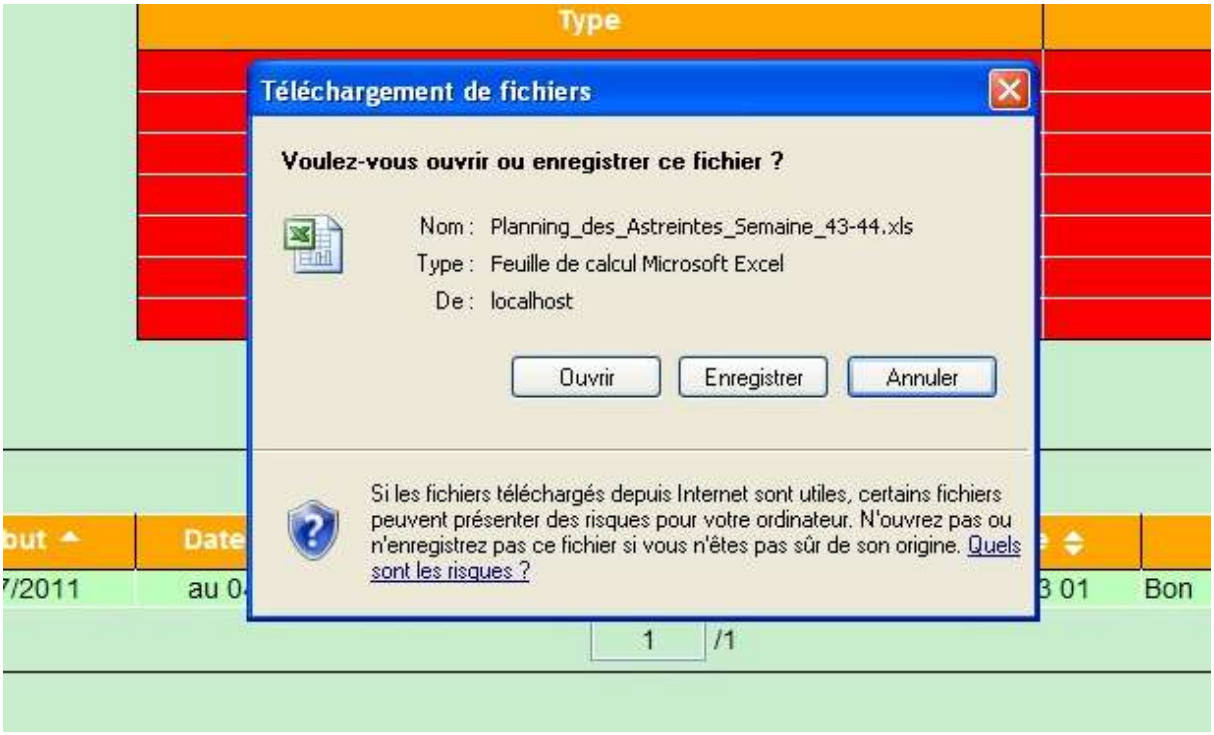

Figure 3-7 export excel

Semaine 32 : du Lundi 8 août 2011 au Lundi 15 août 2011			
Astreintes	**Période**	**Collaborateur**	**Téléphone**
grave			
Kevin	du 12/08/2011 10h au 25/08/2011 10h	Shirley tracy	706012222
Kevin	du 03/08/2011 10h au 19/08/2011 10h	Jimmy	706012222
Allison	du 04/08/2011 20h au 18/08/2011 20h	Fannie	1520002123
médiocre			
Kevin	du 06/08/2011 12h au 09/08/2011 12h	Jimmy	706012222
Kevin	du 06/08/2011 12h au 08/08/2011 12h	Jimmy	706012222
Allison	du 05/08/2011 19h au 31/08/2011 18h	Fannie	1520002123
anodin			
Kevin	du 19/07/2011 00h au 25/08/2011 00h	Cora	706012222
Kevin	du 06/08/2011 11h au 25/08/2011 22h	Shirley tracy	706012222
Diana	du 01/08/2011 20h au 26/08/2011 20h	Ellen	4540253652
Allison	du 01/08/2011 22h au 01/09/2011 22h	Pitter	1520002123
Allison	du 03/08/2011 16h au 31/08/2011 18h	Fannie	1520002123

Semaine 33 : du Lundi 15 août 2011 au Lundi 22 août 2011			
Astreintes	**Période**	**Collaborateur**	**Téléphone**
grave			
Kevin	du 17/08/2011 20h au 20/08/2011 20h	Cora	706012222
Kevin	du 12/08/2011 10h au 25/08/2011 10h	Shirley tracy	706012222
Kevin	du 03/08/2011 10h au 19/08/2011 10h	Jimmy	706012222
Allison	du 04/08/2011 20h au 18/08/2011 20h	Fannie	1520002123

Figure 3-8 the contents of the excel

3.2 Design and Solution

As we have mentioned in the beginning that the main purpose of this thesis is to make an investigation of the benefits and disadvantages of using the Struts and Hibernate frameworks compared to using none of frameworks. An appropriate way to solve this problem seemed to be to do a case study, where the first case is implementing the specified application using Hibernate and Struts, and the second case is changing the implementation of the application that using none of the frameworks but only EJB 3.0 technologies (where the persistence part should be done by the Java Persistence API), Servlets and JSPs.

The first case we used is the project we had referred before, however, in order to make comparison more intuitive, we just select one module of the project as the example and implementation it in two ways.

3.2.1 Functionality Specification

I select User Login part to make the comparison, when a user enters the correct information, he can login the system successfully, and this part have some functions which are:

- Validate the user name and password.
- There should be a web page where the user can initialize the password when he forgets it.
- The user should be able to send an email to the administrator when he forgets his password.
- The password should be encrypted by MD5.
- A message should be shown to the user when he enters the wrong password.
- The latest instructions need to be performed when the user login the system successfully.
- All the information of the user should be put in Session when he login the system successfully.

3.2.2 The GUI

One thing that is the same in both cases is the Graphical User Interface (how the web pages look). Therefore, a short presentation of it is made, with pictures and text.

3.2.2.1 The First Page

The first page to be shown in this application is the Login page (“accueil.jsp”).

Figure 3-9 login page

Figure 3-9 shows the appearance of the page:
At the top there‘s a system‘s name “OUTIL DE SUIVI PILOTAGE MUTUALISE”, and followed by two text fields which are user name and password. Below that are button to validate the information. Finally, if user forgets his password, there‘s a link “Vous avez perdu votre mot de passe?” to initialize the password.

3.2.2.2 The Sencond Page

This page, "reinitialize.jsp" shows functions to initialize the password when the user forgets his password. (See Figure 3-10)

Figure 3-10 initialize password page

At the top of the page is also the system's name and followed by a request which means if you enter the login, you will receive your new password in your email. Below is asked to enter user login, which can be done using the button "REINITIALISER", and if you don't want to do this, click "RETOUR" button to return to the first page.

3.2.2.3 Result Page

The last page to be shown (see Figure 3-11) is just, if the user enters the correct information, he will login the system successfully and view the latest instructions.

Figure 3-11 success page

At the top of this page is a logo of the company and the system‘s name. The date, user name and logout button are presented in the top right. Below that are the system menu and the latest instructions. The instruction in red color indicates that it is more important than others.

If the user enters the wrong password, an error page will be shown, (see Figure 3-12) this page is almost the same to the first page, but there is an error message to be performed when a user enter the wrong login or password

Figure 3-12 error page

3.2.3 First case: implementation using Hibernate and Struts

This implementation was made using Hibernate and Struts framework which we have finished during my internship.

Environment of the first case:

Technologies: Struts2, Hibernate3, JSP, CSS, Javascript.

Developing tools: Eclipse, MySql workbench.

Server: JBOSS.

Database: MySql.

Concerning the four layers architecture, here‘s a list of where all the components of this application belong:

- **Presentation layer**: The –View" and –Controller" parts of the Model-View-Controller pattern of Struts 2 (The FilterDispatcher, JSPs using the Struts 2 tag library and access to the Value Stack)
- **Business logic layer**: The –Model" part of the MVC pattern of Struts 2 (The Action + POJOs).
- **Persistence layer**: Hibernate (Hibernate Mapping files, Hibernate Session etc.), DAOs (Data Access Objects) and Persistent classes.
- **Database layer**: MySQL database that we have referred before.

The first thing that was done was to set up Struts 2 and Hibernate (adding the correct libraries etc.) so they could be used by the application.

In order to make the application support Struts 2, we should configure an xml file called web.xml (See Annex 1)

The database tables we used are –User" and –Consignes", and the Hibernate mapping files and their corresponding POJOs should be created.

Another thing that was added to facilitate the persistence operations was a static Java class that creates a SessionFactory when the application starts executing. This class also keeps a Session and a Transaction object available through methods (like getSession and beginTransaction) (See Annex 2)

After that the Struts 2 components were created (the actions, mapping of actions against requests and JSP pages using the Struts 2 tag library).

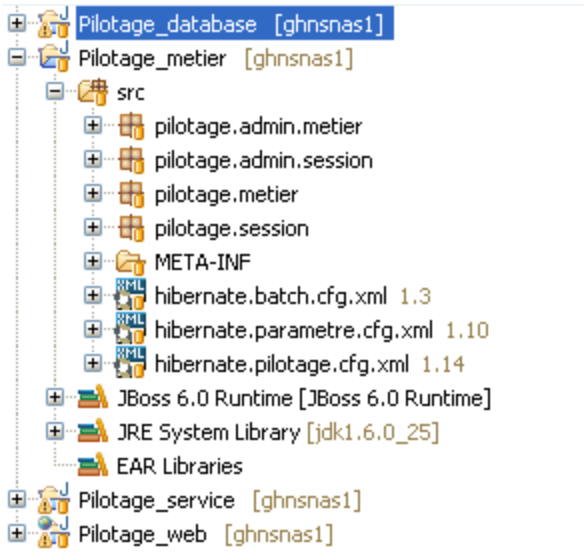

Figure 3-13 project structure

Figure 3-13 presents the structure of the project which the first case also used. ―Pilotage_database" and ―Pilotage_metier" are the persistence layer. ―Pilotage_database" include all the DAOs, and ―Pilotage_metier" contains all the Persistence classes and Hibernate Mapping files corresponding all the database tables, the SessionFactory is in the package ―pilotage.session".

―Pilotage_web" act as the Presentation layer and the Business logic layer include all the JSPs and Actions. (See Figure 3-14)

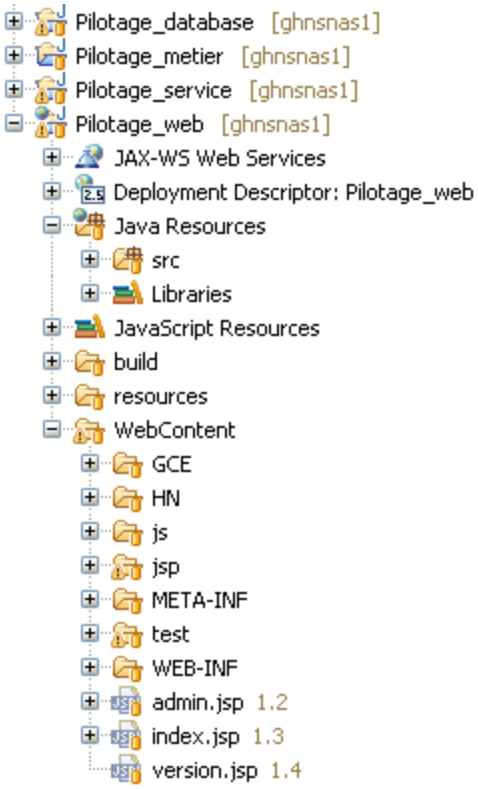

Figure 3-14 the Presentation layer and the Business logic layer of the project

The "Pilotage_service" contains some other services such as export the excel, send the email and so on.

Here's a more detailed description of everything that happens in this case (and how it happens) in the order that the components are called:

Login system:

When the user has submitted the Login and Password, Struts 2 creates an instance of the LoginAction class and saves the submitted values to its properties. Then its "validate" method is called (automatically by Struts 2) and the values are validated against certain criteria. For example, in this case, the "Validate" method will encrypt the password through invoking a method in "Piloage_service", and then invoke the DAO which in the persistence layer to make the comparison between the login and password and the data from the User table. If there are any errors the value "input" is returned by Struts 2, which means that the error page is shown with some error message.

If the validation went well the "login" method of the action is invoked. This method saves the User information (select from the database) to the HttpSession, selects all the latest instructions from the "Consignes" table and gets a list of "Consignes" object which gets populated by Struts 2 and since it also has a property for the index of the chosen row the methods become very simple, finally return to the success page (accueil.jsp).

To be able to use the data transfer features and tags of Struts 2 in an appropriate way, in the "accueil" page, the rows with date, instructions and all the icons are generated by iterating through a list of "Consignes" object. When Struts 2 goes through the list it tries to match all the properties of the current "Consignes" object against the names of the tags (for data and instruction).

Another thing that was needed when the user has pushed a button was a way to know which row this button belonged to (since all the rows belong to the same HTML form in this application). This problem was solved by a hidden value which gets the index of the row the user wanted to change using JavaScript.

Initialize the password:
When the user forgets the password, he can click the link "Vous avez perdu votre mot de passe ?" redirect to the initialize password page. One feature of Struts 2 is that you can call different methods of the action class depending on which button was pushed. So for the "REINITIALISER" button the "initialize" method which is also in LoingAction class is called, this method can send an email to get a new password. If it's failed to send the email, the initialize password page is also shown but with an error message. This is automatically handled by Struts 2.

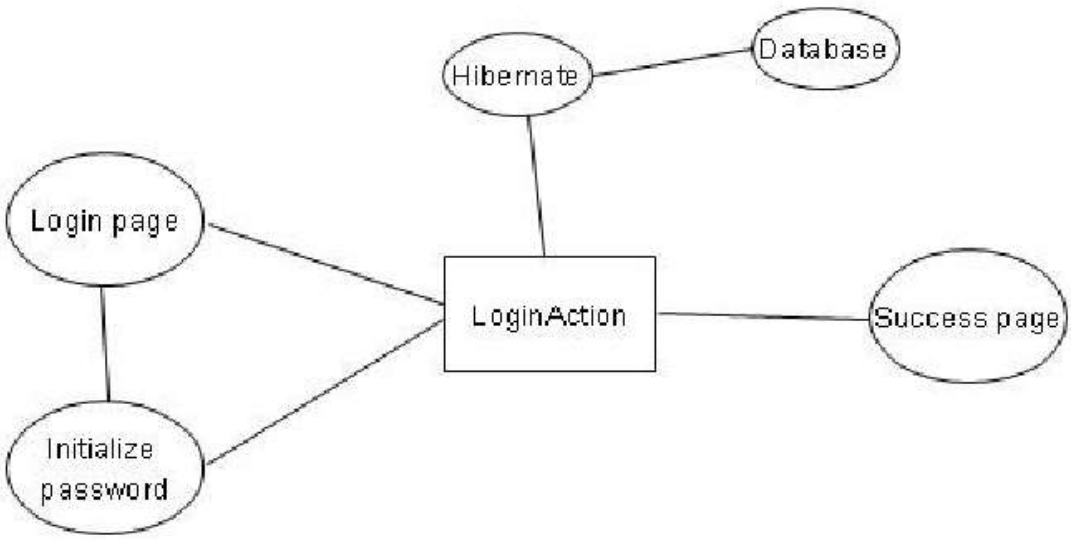

Figure 3-15 the process of the first case

Figure 3-15 shows the process of this case, all pages directed control by an xml file (struts.xml). The action handles the business logic.

3.2.4 Second case: implementation without using Hibernate and Struts

Since the difference between this case and the first one merely is about none use of frameworks, a lot of code could be reused from the first application, as well as the database.

Environment of the second case:
Technologies: Servlet, EJB3, JSP, CSS, Javascript.
Developing tools: Eclipse, MySql workbench.
Server: JBOSS.
Database: MySql.

Concerning the four layers mentioned earlier, here‘s a list of where the components of this application belong:

- **Presentation layer**: JSPs and Servlets.

- **Business logic layer**: Session beans and POJOs. Most of this code can be reused from the first application.
- **Persistence layer**: The EJB 3 entities, JPA and the session beans.
- **Database layer**: The same MySQL database that was used in the first application.

The first thing was done was to create the entities for database (persistence layer) according to the specification made by the company. After that a test servlet was made which created some User entities and persisted them to the database. The purpose of this was to test whether the object-relational mapping worked the way it should, and the test went well.

It can be mentioned for the JSPs that they were made in a simple way (because the front end was less important in this case), using mostly HTML forms, simple Javascript and CSS for page layout, and submit buttons for passing values from the user to the server (and not using any tag libraries).

Login system:
For the login page, it looks the same in this application as in the first one, when the user submits the information, the servlet LoginServlet is called, which receives the values as a HttpServletRequest.

A problem that appeared was that names which contained either of the French letters ‗ê', ‗é' or ‗à' could not be read correctly. This happened because of a flaw in the handling of multipart requests which makes it impossible to read characters (from the request) in the UTF-8 (8-bit UCS/Unicode Transformation Format [13]) format. Instead they are received in the ISO-8859-1 format (the default encoding of file type documents delivered by HTTP), so a conversion method had to be used, from that format to Unicode.

When the login and password have been received (in the right format), they are passed on to a session bean which can validate them. The servlet also invoke the method to encrypt the password. If it's the wrong password or login, the servlet will redirect to the first page, and put the error message into the HttpSession. On the contrary, if the information is correct, they are save to the current HttpSession object, which means that they will ―survive" as long as the user session lasts. Then another session bean finds all the latest instructions in the

„Consignes" table and returns them to the servlet as a java.util.Set (a collection class which automatically removes duplicates from the list). More specifically, it's just the date and the name of instructions that are saved to the set. At last, the set of instructions should be saved to the current HttpSession object and the servlet will redirect to the success page.

In the success page (accueil.jsp), the above mentioned items are collected again from the Session and for each instruction in the instruction set, the date and the instructions are performed in the page.

Initialize the password:
When the user forgets his password and presses the link for initialize the password, a different servlet is called (InitializeServlet). This servlet is handling all the method that is needed to send a email to the administrator.

The first thing that is done in this servlet is collecting the login name from the HttpServletRequest. The next thing to be done is checking whether this is a existing user in the User table. If it's true, send a email to the administritor to initialize the password.

In this case, the error handling is done by creating a HashMap for each request that has been received by a servlet. For each input value like login and password, the value is validated (if it's needed) against certain criteria and if the validation fails a specific error message is put in the HashMap. Right before the redirection to a JSP is made the error HashMap is sent to it as an attribute to the HttpServletRequest object. Each JSP knows which error messages to search for and shows them if they're found in the HashMap.

Figure 3-16 shows the process of the second case.

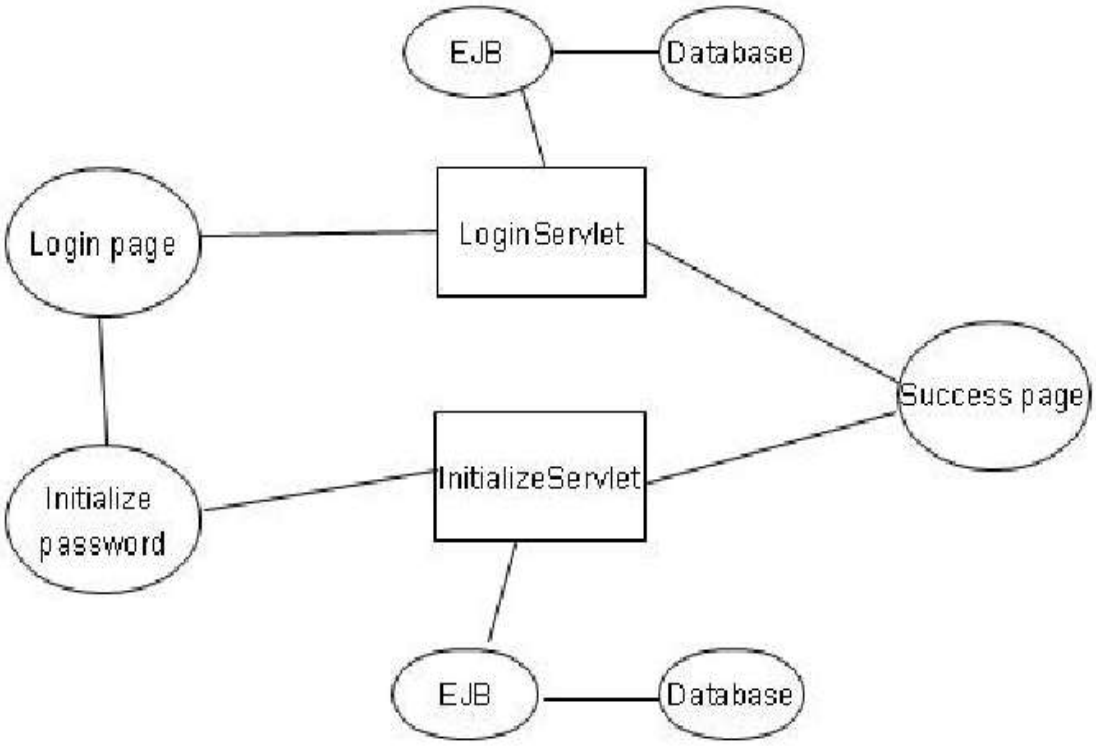

Figure 3-16 the process of the second case

3.3 Results

As specified under –Design and Solution" the purpose of this thesis was to investigate the advantages and disadvantages of using the Struts and Hibernate frameworks, by creating an application in two cases. – The first case without using the frameworks and the other case using them.

One important thing that has become clear throughout the course of this thesis is that to be able to perform a thorough investigation as mentioned above more time would have been needed than the time that was disposed. That is mostly because the Struts 2 framework is very complicated and takes a lot of time to learn, but also because there's a lot to learn about the EJB3 and Servlet technology (which was used in the sencond case).

Therefore the focus of this thesis has been on finding the most significant advantages and disadvantages of using the frameworks from the perspective of the developer. The results are as follows:

3.3.1 Advantages of using Struts 2

- Automatic type conversion.
- Automatic data transfer between actions and HTML forms.
- Easy mapping of requests to certain method in actions.
- Automatic handle the layout of JSP pages by using themes.
- Easy handling of errors.
- Support for localization and internationalization of the text.
- Using many of Struts 2 tags for generating HTML tags.
- ―Clean" Java code.
- Many possible choices for implementation.
- Testability.

Here follows a more detailed description of this list.

3.3.1.1 Automatic type conversion

One extremely useful feature of Struts 2 is the automatic type conversion. Struts 2 uses OGNL (Object Graph Navigation Language) for type conversion. Some type conversion needs to be made during the data transfer between actions and HTML forms. This is done automatically by Struts 2 and it can save a lot of time for developers. The framework includes converters for basic and common object types and primitives.

The more complex Java data types like List and Map can even use the automatic type conversion. For example, as we mentioned in the case 1, if we get a list of latest instructions in the actions, we can simple get it in the HTML forms by using the same object name.

3.3.1.2 Automatic data transfer between actions and HTML forms

Another important feature of Struts 2 is the automatic data transfer between actions and HTML forms. We can transfer more specifically data between Java properties of the action class and form fields in both directions.

The developer doesn‘t need to write code for retrieving the data from the action in the JSP, or for retrieving the data from request in the action. This also saves a lot time, because the data is already there.

3.3.1.3 Easy mapping of requests to certain method in actions

In the first case, when we validate the password or initialize the password, we can use the same action ―LoginAction”, just call the different method, ―login” method for validating the password, ―initialize” method for initializing the password. So in Struts 2, it is possible not just to map a certain HTML form to a certain action but also to map a certain button to a certain method in the action, this can be done just by setting an attribute in the Struts 2 form tag.

But in the second case where servlets are used the first method that‘s called is always the method ―doPost”, there‘s no way to call a different method as the first case with that technology.

3.3.1.4 Automatic handle the layout of JSP pages by using themes

When we have a simple layout, the default themes of Struts 2 are very useful. Struts 2 can automatic handle the layout of JSP pages by using themes. It just uses some Struts 2 tags on the JSP page to include the standard theme and then all the components of the page are put in a two column table and aligned nicely.

Figure 3-17 login field

For example, in login field, Struts 2 just need use two simple tags:
<s:textfield name="username" label="Login :" size="40" maxLength="30"></s:textfield>
<s:password name="password" label="Mot de passe :" size="40"> </s:password>
If the align by using Struts 2 tag is not meet your requirement, add an attribute theme="simple" then you can use the HTML tag as you want.

Also, the standard theme can automatically generates red error messages, which is what most developers want and what most users expect. See Figure 3-18:

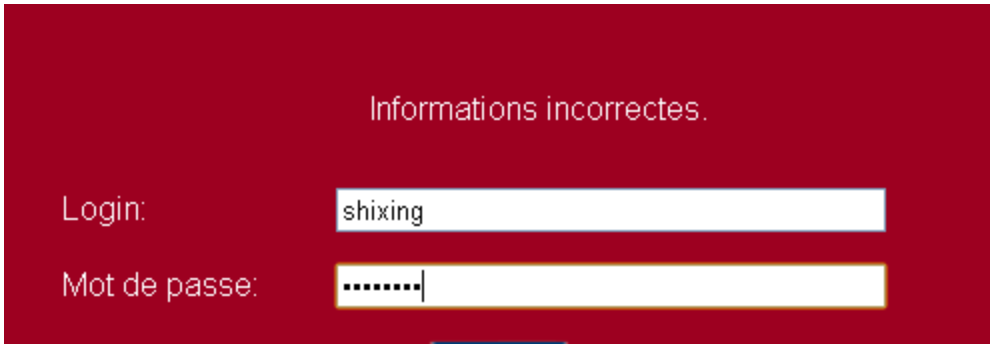

Figure 3-18 error message

In the second case (with EJBs, servlets and JSPs) this had to be done by using style sheets.

3.3.1.5 Easy handling of errors

Struts 2 has done a lot of effort for handling of errors. The user can choose between having a validate method in an action, putting the validation code in the execute method of the action or writing XML validation files.

In our case, we put the validation code in a method called –validate" which can validate the login and password. This is especially useful for validation form field values. All the developer has to do when a validation error is found is to call a method where the field name is connected to the error message to be shown. Then the input page (the page that those form fields belong to) is automatically shown along with the error messages (in the case, the error message is –informations incorrectes"), put next to the form fields that they belong to.

Validate method:

```
public void validate() {
            //Si on n'a plus les parametres en session, on les remet

    if(!Boolean.TRUE.equals((Boolean)session.get(PilotageConstants.PARAM_IN_SESSI
ON))){
                EnterAction.putParametersInSession(session);
            }
            //if the username or password are incorrect, give an error message
            try{
            if(!LoginDatabaseService.findUser(username, password,
           (String)session.get(PilotageConstants.CLE_ENCODAGE))){
            this.addActionError(getText("login.erreur.information.incorrect"));
                }
            }
            catch (Exception e) {
            this.addActionError(getText("error.message.generique") + " : " +
e.getMessage());
            }
    }
```

If the user enters wrong information, we just need to invoke a method –addActionError" which has been defined in Struts 2, the error method will be shown in Login page.

It saves some time for the developer not having to worry about how to show the right error messages.

3.3.1.6 Support for localization and internationalization of the text

There are two important elements when discussing multi-language support in application development – internationalization (I18N) and localization (L10N).

Internationalization (I18N) refers to the process of designing a software application so that it can be adapted to various languages and regions without engineering changes. For web applications, this is of particular importance because the potential users may be from world-wide. Localization (L10N) refers to in-application support of different date, time, numbers, currency etc. formatting, with respecting on local (regional) grammar and formatting issues – for example different numbering in Arabian, Roman and Cyrillic language groups.

Each client needs multi-language support for his application, so we have to develop an application in that scenario if user accesses that site at their location so he can access an application on his mother tongue language.

Struts 2 can achieve this problem very easy manner. It is very supportive for localizing your application. If you have created different language files and put them where Struts 2 can find them the text in the JSP pages is automatically shown in the current chosen language.

3.3.1.7 Using many of Struts 2 tags for generating HTML tags

As mentioned in the „Struts 2" chapter it's possible to use Struts 2 tags for generating HTML tags, which is a very useful and time saving feature. It's also a very important part of Struts 2.

The Struts 2 tags can be divided into two types:

Struts Generic Tags

Struts Generic Tags are used to control the execution flow when pages are rendered. It can further classified into Control Tags (such as if, else and iterate) and Data Tags (such as bean, push and i18n).

Struts UI Tags

Struts UI Tags are mainly designed to use the data from your Value Stack or from Data Tags. These tags are used to display the data on the HTML page. The UI tags are driven by templates and themes.

The JSTL (Java Standard Tag Library) often used with Java EE applications also has a lot of useful tags but not for generating HTML. Although there are other tag libraries can perform this task, you have to use additional frameworks. The Struts 2 tag library combined with the OGNL (Object-Graph Navigation Language), the struts 2 Value Stack and automatic data transfer seems to facilitate things for the developer more than the JSTL does and it‘s definitely worth using.

3.3.1.8 “Clean” Java code

One of the fundamental design goals of Struts 2 framework is to bring MVC (Model-View-Controller) design pattern into the Web application development. MVC pattern enables separation of concerns and allows for the clean and loosely coupled code which is easy to maintain. This means the method that does all the business work of an action can have this simple interface:

public String execute()

The action class just has to extend one class, called ActionSupport which implement many default interfaces and methods, and if you want some session information in your action, just implement SessionAware in this action. The session interceptor detects that interface and sets the session on your action.

If you don‘t like having to pass around HttpServletRequest and HttpSession objects as method parameters just to be able to use them (like in the second case where servlets are used), you can use a Struts 2 feature where these objects are injected as properties of your action, through certain get methods (but then you have to implement certain interfaces).

It can also be called ~~“c~~leaner” code to use the Struts 2 tags than using scriptlets in the JSPs (as in the second application) since the code will take up less space and spend less time while doing the same thing.

3.3.1.9 Many possible choices for implementation

One general advantage of Struts 2 is that it gives the developer many possible choices for implementation. This concerns both the configuration of Struts 2 itself and the implementation of the action classes.

Depending on the developer‘s wishes, lots of XML files and interceptors (a reusable component which can be used for separating things like validation and logging from the action code) and less Java code can be used, or less XML files and interceptors and more Java code.
In the first case, we can put email information (such as email receiver, content, address) or error message into a property file, and Struts 2 provide some interface to invoke them in a simple way (like getText() and so on). This freedom of choice is not as big in the case of the second application.

3.3.1.10 Testability

A project should pay much attention to its test. The usability of the project depends on testing situations. Struts 2 support many methods to be tested.

Struts 2 Actions can be tested by instantiating the Action, setting properties, and invoking methods. Dependency Injection support also makes testing simpler.

3.3.2 Disadvantages of using Struts 2

- Time consuming to learn.
- Rigid approach.
- Dependent on Servlet.
- Hard to change the FreeMarker templates.
- Hard to debug the parts that are not Java code.

Here‘s a more detailed description:

3.3.2.1 Time consuming to learn

The biggest disadvantage of Struts 2 is that it‘s so complex that it takes a lot of time to fully understand it and learn the different parts. We know Struts 2 is a framework. It mixes many different technologies like interceptors, Java classes, XML files and FreeMarker templates (inside themes), so it is hard to get a grasp as to where to start.

The creators‘ idea of making a web framework that cleanly separates the different parts of the Model-View-Controller pattern and keeps ~~”e~~lean” Java code resulted in a combination of interceptors, Java classes, XML files and FreeMarker templates and more. The huge drawback of this is that to an ~~”or~~dinary Java developer” it might be confusing trying to learn the different technologies and putting them together. And for anyone who wants to have a deeper understanding of how and why things are happening, Struts 2 will most likely be a source of irritation since it‘s using so many default things like default interceptors and default themes. It‘s possible to learn about these things but (as mentioned before) it‘s very time consuming.

3.3.2.2 Rigid approach

Struts 2 has a fixed process, but the other way round, it is too rigid.

For example, when the action finish its business logic, it will go to the presentation layer (most of them go directly to the JSP page), and you will configure the forward. Every time related to forward the page, it has to configure once. For the login page, if the user enters the correct information, the action will forward to the success page, the configuration code is as follows:

```
<action name="loginAction" class="pilotage.login.LoginAction">
<result name="success">jsp/user_pages/login/accueil.jsp</result>
<result name="input">jsp/user_pages/login/loginUtilisateur.jsp</result>
</action>
```

These codes are written in an XML file called struts.xml. We can imagine that A big project has a lot of JSP page, and this XML file involve all the configure code to forward the page. It

will very inconvenient for maintain or update. After each configuration change, requiring re-deployment of the entire project, and restart the server (like tomcat), if the business of a complex system changes frequently, it would be inconceivable for such a complicated operation.

3.3.2.3 Dependent on Servlet

Another disadvantage of Struts 2 is that it‘s Excessive dependence on the Servlet. It needs to rely on ServletRequest and ServletResponse when Struts handle actions, all of which cannot escape the Servlet container.

3.3.2.4 Hard to change the FreeMarker templates

One feature of the first case is that it uses the theme of Struts 2. This default theme puts all the web components in a table structure with two columns (Login TextField), but for the other pages which needs a structure with many columns (for showing the instructions list and buttons to delete or update). Since there was no default theme for this layout it seemed like the only solution to the problem was to learn the FreeMarker language that‘s used in the theme template files and learn how a theme was built. Though, this is like learning a whole new framework and there was no time for this. So we have to use default JSTL (JSP Standard Tag Library) or add an attribute theme="simple" (the first case we have mentioned).

This whole issue feels like a major drawback of Struts 2.

3.3.2.5 Hard to debug the parts that are not Java code

Another problem for a Struts 2 developer is that it‘s hard to debug the parts of the application that are not Java code. This can be for instance Struts 2 tags, interceptors and XML files (struts.xml).

3.3.3 Advantages of using Hibernate

- Database independent.
- Time saving.

- Flexibility.
- Possible to specify a name for the foreign key constraint and an index column in the Database.

Here‘s a more detailed description:

3.3.3.1 Database independent

One important feature is that Hibernate is database independent and you can use any database of your choice, so it can easy migrate your code between different databases. Good for updating, maintainning your application. As we have to work on POJO class for interacting with the database, so it‘s basically reduce dependency on JDBC.

3.3.3.2 Time saving

In Hibernate, we can generate the SQL on the fly and then automatically executes the necessary SQL statements. This saves a lot of development and debugging time of the developer. Writing JDBC statement, setting the parameters, executing query and processing the result by hand is lot of work. Hibernate will save all tedious efforts.

Hibernate also maps your domain object with the relational database. Now you can concentrate on your business logic rather than managing the data in database.

For the second case which uses JDBC we have to manually handle exception and every time we need to open or close the connection, create or close the statement, resultset whatever we have used. In Hibernate, we don‘t need to care about these and they have automatically executed by Hibernate.

3.3.3.3 Flexibility

Hibernate is a very flexible framework. Same problem can be solved by several solutions. For the Collection you can use Set, List or Bag, for the Query you can use iterator or list, for the composite primary key you can configure in the hbm (hibernate configuration file) or customize CustomerType, and for a table, you can choose to map a single object or map into the parent and child objects. All of these you can choose according to your actual situation.

3.3.3.4 Possible to specify a name for the foreign key constraint and an index column in the Database

Hibernate is standard ORM solutions and it also supports JPA (Java Persistence API). But a useful feature in Hibernate over JPA is it‘s possible to specify a name for the foreign key constraint and an index column in the Database.

One Hibernate specific annotation (that means that it‘s not a part of the Java Persistence API) is the ForeignKey annotation. This is useful when one is automatically generating database tables but still wants some control of how the tables are defined. More specifically, with this annotation it‘s possible to set the name of a Foreign Key constraint in the database.

Another Hibernate specific annotation is the index column which affects database tables. If you want to create an index column for a database table there‘s no way to do that using JPA, but with Hibernate‘s IndexColumn annotation it‘s possible. An index column is good for speeding up database operations.

3.3.4 Disadvantages of using Hibernate

- The automatic generation of database tables might be difficult to use.
- Not suitable for smaller project.
- Hibernate is not a standard.

Here‘s a more detailed description:

3.3.4.1 The automatic generation of database tables might be difficult to use

When we use hibernate in eclipse which can automatic generate the database tables to the hbm.xml files and simply add the hibernate configuration file to the project, but these files sometimes may be hard to use and to debug if it doesn‘t work properly.

Hibernate also limits the object model which we used, for example, a persistent class cannot be mapped to multiple tables.

3.3.4.2 Not suitable for smaller project

Hibernate is good for big project, but in smaller project, the developer don‘t want to use it because of its lower development efficiency and operational efficiency.

It is clearly that just using JPA (Java Persistence API) and JDBC can obtain the highest efficiency, because we don‘t need to care about any configuration files, import other Jar drivers, and learn other technologies. Compare to EJB which we used in second application, hibernate has lower development efficiency in small project.

3.3.4.3 Hibernate is not a standard

One advantage that JPA has over Hibernate is that it‘s an open standard which has been implemented by different vendors. This means that if the JPA implementation (persistence provider) of one vendor doesn‘t satisfy your needs you can switch to another implementation.

But if you would find a bug in Hibernate or if it just wouldn‘t suit your needs you would have to implement the solution yourself or wait for somebody else to do it since there‘s only one Hibernate implementation available.

4. Conclusions and Future work

4.1 Conclusions

Struts 2 is a very complicated web application framework which combines a lot of different technologies like Java classes, interceptors and XML files. Though, apart from this fact it has a lot of advantages when being compared to using only servlets, JSPs and EJB3 technology. If the developer has enough time to put into learning a new framework and is interested in learning new techniques, Struts 2 is worth using. Because its advantages far outweigh its faults.

From the Hibernate version 3.2 it became possible to use annotations with Hibernate and since Hibernate implements the EJB 3 Java Persistence API these annotations look almost the same (Hibernate has some additional annotations). So the biggest difference between case 1 and case 2 when it comes to Hibernate is the interfaces used for the actual persistence operations and the fact that JPA (used in case 2) is a standard and Hibernate is not. If the developer wants to use a standardized framework JPA is better than Hibernate, but if the developer needs to specify more specific details about database tables the Hibernate annotations are better than the JPA ones. So choosing the best persistence solution is mostly a matter of situation and taste since both are very good persistence solutions.

In conclusion, there are many advantages of using software frameworks in implementation of whole range of applications or projects, from simple one-case solutions to large-scale, enterprise-level system architectures. But, on the other hand, frameworks won‘t solve all problems and are neither suitable nor a good choice for ever project, solution or implementation.

Application frameworks offer a variety of advantages:

1. Using code which has already been built, tested, and used by other programmers increases reliability and reduces programming time. In a corporation this code reuse effectively saves money.
2. Thanks to framework modularization, software development teams can be split into groups developing different modules. This separation of tasks lets each team focus on

more specific goals and use their individual strengths. For example, the programmers who are experts at user interface design might work on the client application while the security experts test and strengthen the framework upon which the application is built.

3. Struts 2 and Hibernate framework can provide security features which are required for most of applications. This provides every application written with the framework to benefit from the added security without the extra time and cost of developing it. Struts 2 follows MVC design pattern, so when you use it you‘ve to follow their coding convention which makes your code clean and extensible for future purpose. When you use a time tested and trusted framework, you can be assured of security. Every popular framework implements high standards of security. Besides, they have a loyal online community who tenaciously work to make the security even better. If you find any vulnerability, you can report it on the framework‘s website so that they can solve it.
4. By handling –lower level" tasks frameworks can assist with code modularity. Business logic, for example, can remain in the application while the tasks of database connectivity and handling user logins can be handled separately in the framework.
5. Frameworks often help enforce platform-specific best practices and rules. A desktop GUI framework, for example, may automatically build toolbars and buttons common to the local operating system. A web application framework may assist with encrypting user passwords or payment processing.
6. Using frameworks, the developers can devote more time in developing the software requirement, not in preparing the environment and tools of application development. It helps the developer to write the codes faster and thus saves the overall expenditure of a project. You‘ll never worry about the project deadline because it can help you to develop the project rapidly.

Although Frameworks have some shortcomings, for example, they are not for the small projects, because in a very simple project custom coding will work faster than setting up a framework. And for the novice user, it is tougher to use the framework quickly as it is big and complex abstract and user has to spend more time in assessing the concept, function and its uses in developing the program, which enhances the development but after learning how to use it efficiently, it becomes easier and quicker to develop any program, module of application. Frameworks often require a significant education to use efficiently and correctly (i.e. Struts 2 have a high learning curve). Therefore specific frameworks very often become more valuable to individual programmers when they are used repeatedly.

4.2 Future work

This project uses a combination of Struts and Hibernate framework to implement, in the structure of the project (see figure 27), we can see that data access is achieved by "Pilotage_database" (DAO) which has a lot of methods for accessing to the database. Whenever the business logic layer (action in Struts) wants to invoke these methods, we need to redefine them (instantiate a new object) before calling the method in them.

In fact, we can implement a unified management through some technologies, and this will greatly reduce the system coupling. Spring can act this role. All instantiated tasks can be managed by Spring. In this way, it can be constituted SSH (Spring + Struts + Hibernate) which is a very popular structure we usually used.

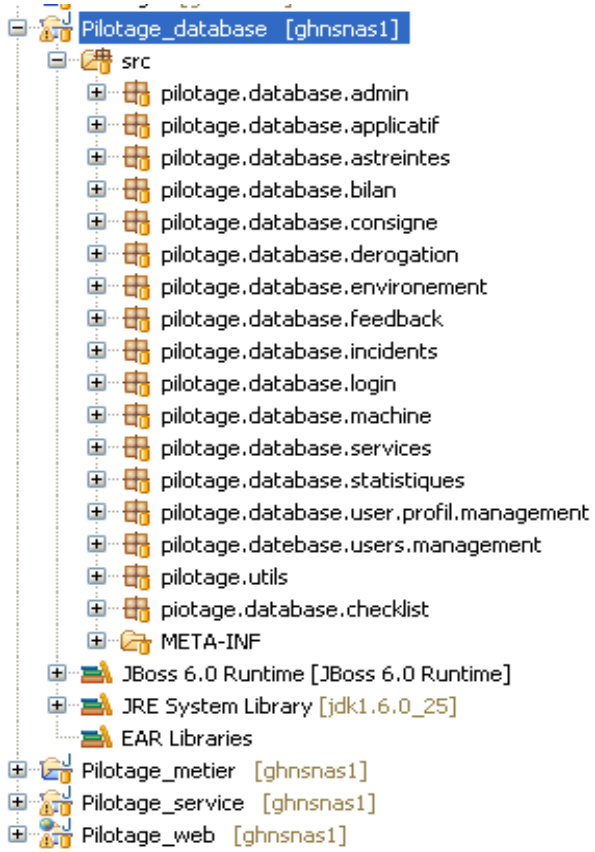

Figure 4-1 project structure

4.2.1 Add Spring Ability

Actually, Spring is a framework, so we should add some libraries before using it. Then, the original project requires a simple modification which mainly reflected in some configuration files.

In Hibernate, only the model and mapping files are required, because Spring will handle the Hibernate configuration. We still use the example which mentioned earlier. For the persistence layer, we need to create a POJO class for "users" table and then create the corresponding mapping file (Users.hbm.xml).

In Struts, implements the Bo (Business object) and DAO (Data access object) design pattern. All the Bo and DAO will be DI (Dependency Injection) by Spring in the Spring bean configuration file. In the DAO, make it extends Spring's HibernateDaoSupport to integrate Spring and Hibernate integration. And Struts 2 action is no longer need to extend the ActionSupport, Spring will handle it.

Almost all the configuration is done in Spring, it is specialized in integration work.
Firstly, we need to create an xml to declare the Spring's beans: Action, Business object and Data access object.

UsersBean.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

        <bean id="loginAction" class="pilotage.login.LoginAction">
                <property name="usersBo" ref="usersBo" />
        </bean>

        <bean id="usersBo" class="pilotage.database.login.UsersBoImpl" >
                <property name="usersBoDAO" ref="usersBoDAO" />
```

```
        </bean>

        <bean id="usersDAO" class="pilotage.database.login.UserDAOImpl" >
                <property name="sessionFactory" ref="sessionFactory" />
        </bean>
</beans>
```

Sencondly, create a sessionFactory bean to integrate Spring and Hibernate, it used to declare the session factory and replace the Hibernate configuration file.

HibernateSessionFactory.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

<!-- Hibernate session factory -->
<bean id="sessionFactory"
   class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">

   <property name="dataSource">
     <ref bean="dataSource"/>
   </property>

   <property name="hibernateProperties">
      <props>
        <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
        <prop key="hibernate.show_sql">true</prop>
      </props>
   </property>

   <property name="mappingResources">
                <list>
```

```
        <value>pilotage/metier/Users.hbm.xml</value>
                </list>
    </property>
</bean>
</beans>
```

Then, create an xml file (SpringBeans.xml) which is a core Spring's bean configuration file, act as the central bean management.

SpringBeans.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

        <!-- Database Configuration -->
        <import resource="config/spring/DataSource.xml"/>
        <import resource="config/spring/HibernateSessionFactory.xml"/>

        <!-- Beans Declaration -->
        <import resource="config/spring/UsersBean.xml"/>
</beans>
```

At last, to integrate Struts 2 and Spring, just register the ContextLoaderListener listener class, define a ~~"c~~ontextConfigLocation" parameter to ask Spring container to parse the ~~"S~~pringBeans.xml" instead of the default ~~"a~~pplicationContext.xml~~".~~

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_9" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
```

```
    <display-name>Pilotage</display-name>

    <filter>
        <filter-name>struts2</filter-name>
        <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>struts2</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/classes/SpringBeans.xml</param-value>
    </context-param>

    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <jsp-config>
        <taglib>
            <taglib-uri>/WEB-INF/tld/c.tld</taglib-uri>
            <taglib-location>/WEB-INF/tld/c.tld</taglib-location>
        </taglib>
    </jsp-config>
</web-app>
```

Spring, Struts, Hibernate for the application of different levels, the Spring was the role of the binder, Struts users business processing, if you use the Tag it is also responsible for the pages show, Hibernate is used for data persistence.

4.2.2 Advantages of using Spring

- Reduce coupling
- Use interface programming
- Replace EJB
- Both comprehensive and modular
- Ease of testing

4.2.2.1 Reduce coupling

One of the most important advantages for the Spring in this project is that it can remarkably reduce coupling of the system. We can see all the relationship between different layers is controlled by Spring configuration file. All the dependencies can be injected by the Spring framework using beans. This can prevent direct contact between classes, and implement loose coupling.

4.2.2.2 Use interface programming

Another useful feature of the Spring is to program to interfaces.

The project should be maintained frequently, sometime it has some bugs. When many programmers are working on a project, work can get done much easier and safer when well-understood methods and functionality are promised via interfaces. If one programmer wants to instance a class, knowing that it inherits from an specific interface will allow him to seamlessly do equality testing with a known method, without having to ever delve into the class and try to understand if equality is established and how.

In traditional project development, for frequently customer requirement variation, we have to keep rewriting the existing business codes if we don‘t use interface programming. It can generate the new bug during rewriting the code, and it also can affect the classes which invoke this business, cause they all need to modify, affect the stability of the system. Interfaces are communication contract between the external world and your system. Well defined interfaces can provide compile time validation and save you and consumers of your system a lot of troubles.

Project which use interface-based programming have clear business logic, well understood code, easy to extend, and have strong maintainability, even if replace a group of people, new people can easily get started, it is extremely important for the company.

4.2.2.3 Replace EJB

Spring also makes it much easier to use, as well as implement EJBs. Many EJB applications use the Service Locator and Business Delegate patterns. These are better than JNDI lookups throughout client code, but their usual implementations have significant disadvantages. For example:

1. Typically code using EJBs depends on Service Locator or Business Delegate singletons, making it hard to test.
2. In the case of the Service Locator pattern used without a Business Delegate, application code still ends up having to invoke the create() method on an EJB home, and deal with the resulting exceptions. Thus it remains tied to the EJB API and the complexity of the EJB programming model.
3. Implementing the Business Delegate pattern typically results in significant code duplication, where we have to write numerous methods that simply call the same method on the EJB.

For these and other reasons, traditional EJB access can reduce productivity and result in significant complexity.

Spring steps beyond this by introducing codeless business delegates. With Spring you'll never need to write another Service Locator, another JNDI lookup, or duplicate methods in a hand-coded Business Delegate unless you're adding real value.

Spring is designed so that applications built with it depend on as few of its APIs as possible. Most business objects in Spring applications have no dependency on Spring.

4.2.2.4 Both comprehensive and modular

Spring has a layered architecture, meaning that you can choose to use just about any part of it in isolation, yet its architecture is internally consistent. So you get maximum value from your learning curve. You might choose to use Spring only to simplify use of JDBC, for example, or you might choose to use Spring to manage all your business objects. And it's easy to introduce Spring incrementally into existing projects.

4.2.2.5 Ease of testing

Spring's inversion of control approach makes it easy to swap the implementations and locations of resources such as Hibernate session factories, datasources, transaction managers, and mapper object implementations (if needed). This makes it much easier to isolate and test each piece of persistence-related code in isolation.

Applications don't contain any code directly using J2EE services such as JNDI, which is typically hard to test. And JavaBean properties are simple, core Java and easy to test: simply write a self-contained JUnit test method that creates the object and sets the relevant properties.

Spring has a package called org.springframework.test which provides valuable superclasses for integration tests using a Spring container, but not dependent on an application server or other deployed environment. The enabling functionality in the org.springframework.test package includes:

1. The ability to populate JUnit test cases via Dependency Injection. This makes it possible to reuse Spring XML configuration when testing, and eliminates the need for custom setup code for tests.
2. The ability to cache container configuration between test cases, which greatly increases performance where slow-to-initialize resources such as JDBC connection pools or Hibernate SessionFactories are concerned.
3. Infrastructure to create a transaction around each test method and roll it back at the conclusion of the test by default. This makes it possible for tests to perform any kind of data access without worrying about the effect on the environments of other tests [14].

4.2.3 Disadvantages of using Spring

Spring is a powerful framework that solves many common problems, but it also has some disadvantages. For example, add a new framework may be make the project more complicated, and reduce the maintainability. If you cannot make a good configuration, the system also can slow. For the beginner, you should spend some time to learn all the technologies.

In summary, it‘s very easy to suggest some future work starting from this thesis since some limitations had to be made compared to the original plan because of lack of time. We can integrate Spring framework into the entire project according to the user requirement. It provides a consistent way of managing business objects and encourages good practices such as programming to interfaces, rather than classes. It also provides a unique transaction management abstraction, which enables a consistent programming model over a variety of underlying transaction technologies, such as JTA or JDBC. But if you don‘t have much time to learn it, or worried about the system complexity, you can just use Struts and Hibernate without Spring. All of this should be determined based on the actual conditions of the company.

References

[1] Rima Patel Sriganesh, Gerald Brose, Micah Silverman, 2006, Mastering Enterprise JavaBeans 3.0, Wiley Publishing, Inc, ISBN-10 0-471-78541-5, chapter 1

[2] Sun - JavaEE5Tutorial. http://java.sun.com/javaee/5/docs/tutorial/doc/geysj.html

[3] Wiki - Java Servlet. http://en.wikipedia.org/wiki/Java_Servlet

[4] Apache Struts. http://struts.apache.org/

[5] Servlet Filter. http://javaboutique.internet.com/tutorials/Servlet_Filters/

[6] Donald Brown, Chad Michael Davis and Scott Stanlick, 2008, Struts 2 in Action, Manning Publications Co, ISBN 1-933988-07-X, chapter 6

[7] Christian Bauer, Gavin King, 2005, Hibernate in Action, Manning Publications Co, ISBN 1932394-15-X, chapter 4

[8] Hibernate definition. http://dev.anyframejava.org/docs.en/anyframe/plugin/hibernate/4.6.1/reference/html/pt02.html

[9] Debu Panda, Reza Rahman and Derek Lane, 2007, EJB 3 in Action, Manning Publications Co, ISBN 1-933988-34-7, chapter 4

[10] Wiki – Spring framework. http://en.wikipedia.org/wiki/Spring_Framework

[11] Spring Framework. http://www.theserverside.com/news/1364527/Introduction-to-the-Spring-Framework

[12] Wiki – Spring Framework. http://en.wikipedia.org/wiki/Spring_Framework

[13] Wiki – Uniode. http://en.wikipedia.org/wiki/UTF-8

[14] Spring testing. http://static.springsource.org/spring/docs/2.5.x/reference/testing.html

Annexes

1. Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_9" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
	<display-name>Pilotage</display-name>
	<filter>
		<filter-name>struts2</filter-name>
		<filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
	</filter>
	<filter-mapping>
		<filter-name>struts2</filter-name>
		<url-pattern>/*</url-pattern>
	</filter-mapping>
	<welcome-file-list>
		<welcome-file>index.jsp</welcome-file>
	</welcome-file-list>
	<jsp-config>
		<taglib>
			<taglib-uri>/WEB-INF/tld/c.tld</taglib-uri>
			<taglib-location>/WEB-INF/tld/c.tld</taglib-location>
		</taglib>
	</jsp-config>
</web-app>
```

2. PilotageSession.java

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class PilotageSession {
	private static final String HIBERNATE_CFG_Pilotage =
"/hibernate.pilotage.cfg.xml";
	private static SessionFactory sessionFactoryPilotage;
	static {
		sessionFactoryPilotage = getNewSessionFactoryPilotage();
	}
	private static SessionFactory getNewSessionFactoryPilotage() {
		return new
Configuration().configure(PilotageSession.class.getResource(HIBERNATE_CFG_Pilotage)).
buildSessionFactory();
	}
	private static SessionFactory getSessionFactoryPilotage() {
		return (sessionFactoryPilotage != null) ? sessionFactoryPilotage :
getNewSessionFactoryPilotage();
	}

	public static Session openNewSession() {
		Session session = getSessionFactoryPilotage().openSession();
		session.beginTransaction();
		return session;
	}
	public static Session getCurrentSession() {
		Session session = getSessionFactoryPilotage().getCurrentSession();
		session.beginTransaction();
		try{
			session.createSQLQuery("SELECT 1").list();
		}
```

```
                catch (Exception e) {
                        closeSession(session);
                        session = getSessionFactoryPilotage().getCurrentSession();
                        session.beginTransaction();
                }
                return session;
        }

        public static void closeSession(Session session) {
                if ((session != null) && session.isOpen()) {
                        session.close();
                }
        }

        public static void rollbackTransaction(Session session) {
                if ((session != null) && session.isOpen()) {
                        session.getTransaction().rollback();
                }
        }
}
```