Conclusion and future Enhancements

The whole project was a nice experience. We learnt a lot of cross-cutting concerns of applications in general – like validation and how using frameworks drastically reduces effort required, development time, cost involved and ultimately results in better design and code.

Use of MVC design pattern allows “plug and play” of the view layer without affecting business code. As a result we were able to change the whole look and feel(JSP+CSS) of the application without touching the business code.

We learnt how incorporating design patterns in our design/code helps in solving recurring problems while incorporating “best practices”.

We have used current hot, industry specifications like J2EE, jasper reporting, log4j, AJAX, etc. Also following best practices like heavily using xml where ever possible has kept the maintainability and extensibility high.

Use of xml has also kept the whole application very much independent of underlying platform. As an example we succeeded in porting the whole application to run on IBM websphere 2.0 web server and IBM DB2 database, in under an hour. The whole SQL querying has been kept independent of the underlying connector.

Some of the other best practices of the day include use of AJAX for dynamic look and feel of forms. Such AJAX codes have been incorporated where ever they are actually needed.

Things like database configurations, server parameters, passwords etc have been put in appropriate xml documents, making porting to other run-time environments easier.

The whole system is built to be capable of running distributed ground up. The actual webserver can be on one system and the whole database can be on any remote system on the internet. By editing simple xml files we can finish our configurations.

All in all, we got an idea of how much effort/time/cost it takes to design and build an application using appropriate frameworks.

# Future Enhancements

* Integration of SSL functionality to deliver secure login using https protocol.
* User customizable look and feel.
* Integration of mail.
* Introduction of a discussion forum.
* Porting to advanced and highly availability run-times on clouds like **Google App Engine**
* Localization and internationalization. This helps in porting of the application to other languages easily.
* Adding more formats of report generation like excel (xlsx), CSV, etc.
* Implementation of Cache in the server side for faster serving of request. This can be both JSP cache and data cache from database.
* Use of UI frameworks like – dojo/jquery/flex enables UI layer widget level reusability and browser independent UI code.
* Usage of iBatis in our application for externalizing sql code in xml’s is the first step towards using Object-Relational Mapping(ORM) tools – like Hibernate. Usage of such ORM frameworks in the business layer enables richer domain model and better/quicker DB design

References

* [www.wikipedia.org](http://www.wikipedia.org)
* ibatis.apache.org
* en.wikipedia.org/wiki/Java\_Platform,\_Enterprise\_Edition
* java.sun.com/javaee
* tomcat.apache.org/
* [www.eclipse.org/](http://www.eclipse.org/)
* Head first servlets and JSP – o’reilly media
* en.wikipedia.org/wiki/JavaServer\_Pages
* java.sun.com/products/jsp/
* struts.apache.org/
* en.wikipedia.org/wiki/Apache\_Struts
* struts 2 in action - Donald Brown, Chad Michael Davis, Scott Stanlick
* Design Patterns CD(GoF) – Erich Gamma and others
* Core J2EE Patterns Best Practices And Design Strategies – Deepar Alur and others