**Tache1 : Etude comparative et technologies choisit :**

1. Etude de de l’utilisation la plus facile

\*Spring MVC ==

spring is for building serious applications with solid foundations, rich user interfaces and a RESTful API, which makes it completely unnecessary for anything significantly simpler than that

\*JSF ==

JSF’s tooling makes it easy to use and there are no external dependencies as long as you stay within the Java EE ecosystem, which JSF leverages well. There are also several built in components that provide tremendous benefit and some third party component libraries that look great.

Note that you should not see JSF as replacement of alone HTML/CSS/JS, you should also take the server side part into account (specifically: JSP/Servlet). JSF removes the need of all the boilerplate of gathering HTTP request parameters, converting/validating them, updating the model values, executing the right Java method to do the business stuff and generating the HTML/CSS/JS boilerplate code. With JSF you basically end up with a XHTML page as view definition and a Javabean class as model definition. This greatly speeds up development.

\* GWT ==

Code is easy to read and write and there’s a Design mode is there for the non-coders.  Client side storage is a little hard to grasp if you haven’t used it before.

\*Grails ==

Grails is, built off Groovy which gives the immediate benefit of being more productive. The Groovy syntax is much terser than Java, it's much easier to do things in one line of Groovy code that would take you several in Java.

Grails specifically provides you with a number of advantages over other web frameworks, Grails is actually built on top of SpringMVC by the way and you can integrate other components using Spring.

* Database Migrations and Versioning - no more application out of sync with database schema syndrome. Most Java web apps suffer from this.
* Artefacts - which make creating new controllers and components easier. No more create a controller, configure it and stuff it into the right place in your web app. doh! Scaffolding also provides you with all some initial components to allow you to start building your pages and customising
* Simpler validation (def simpler than Struts 1.x), e.g. username(size:6..10, blank:false)specifies two validation rules for a username field to be of a specific length and non blank. It's a bit harder in the other Java web app frameworks.
* Built in Webflow (via Spring webflow) which makes creating complex workflows much simpler. Struts 2 can support Webflow, but via a plugin which is a little odd if I rememeber. Spring can too.
* Interceptors - SpringMVC also has these.

1. Comparing Spring MVC to JSF :

Hindering testability: I can’t speak for most frameworks but I can compare Spring MVC with JSF. Let me tell you this: If anybody is telling you JSF is nicely testable he probably doesn’t know automatic testing. With JSF you can test your backend beans using unit tests. You can test the whole UI, by deploying the application to a server and hitting it with Selenium. That’s basically it.

Just in case you are wondering what else one should be able to test: Load a static version of a page in a browser and testing it with selenium, in order test your Client side UI behavior. Test your generated markup without starting a full blown application server. Test the mapping of attributes/parameter to bean methods. Test your generated markup without bootstrapping a complete application. All this is perfectly possible with Spring MVC and probably with many other sane server side frameworks, but not with JSF …

3-what is the fastest?

**Spring is the fastest framework** but it costs around 50% more to scale with it than with GWT. **GWT is the cheapest framework** to scale but it gives a slower response time. Wicket and JSF are slow and expensive.

Spring MVC is a web framework inside the Spring framework. It does provide features as those in JSF 2.0:

* ajax-support
* validation
* dependency-injection
* etc.

Yet, you can use Spring (not Spring MVC) together with JSF 2.0, with spring providing the dependency-injection, aop, transaction management mechanisms, and JSF providing the web layer.

Décision= choisir Spring pour la couche métier et le jsf pour la couche présentation car je suis familiarisé avec

Conclusion =

The primary reason for using both platforms is that while Spring excels at general purpose [java](http://www.javaranch.com/) needs, JSF has more power in the specific area of MVC-based web GUI support. Since they play well together, I get the best of both.

**Tache 2 : Etude de faisabilité**

1. extraction des repositories GIT ou CVS :

Pour cette partie on va simplement télécharger ou bien cloner avec le git ou faire un check out si le repo est en CVS sur le serveur d’application pour bien faciliter le traitement des données.

1. comparaison fichier par fichier :

Donc on va comparer les fichiers pour trouver les ressemblances pour chaque fichier selon une expression régulière contenant le nom propriétaires qui en communs entre tous les fich.

1. Comparaison entre version :

Ligne par ligne, un algorithme sera amené à balayer chaque ligne de chaque version pour trouver les différences et les modifications réalisés.

4-Possibilité de Merge :

Enregistrement des changements avec la détection des lignes et des colonnes afin d’assurer l’insertion de la partie enregistrée dans le fichier de référence ou l’index choisit par l’utilisateur dès le début.

5-Compilation à distance :

A ce niveau là , la compilation de code déjà traité sera compiler à l’aide des commandes Maven qui seront intégrées dans le code java.