# Maintainability

## Readability

Readability is the ease with which a reader can understand a written text. Here, we analysis the readability from four aspects – source code structure, framework, size of project and the complexity of source code.

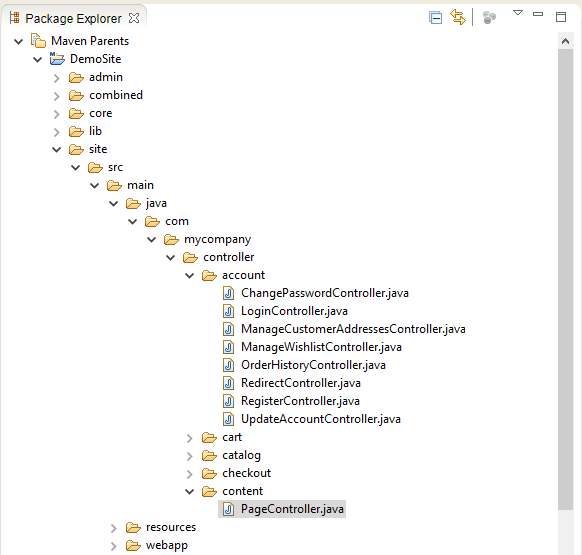


Figure 1. Part of Source Code Structure Example

The source code structure is very clear and easy to understand. The above figure 1 is the snapshot of source code structure for the site part. The site module implements all the features for public customer including account controller, shopping cart, product browser (catalog) and checkout. For each feature, all necessary actions are designed by different controllers such as the account folder contains login controller, change password controller and so on. Because the project uses Spring framework based on MVC pattern, there is page controller to dispatch pages mapping the url. And the webapp folder contains all the pages will be presented in the front-end. Also, there are some documents that introduce all the modules in the website.

Because of using Spring framework, we have to cost more time to understand all the concept about this framework such as MVC pattern, annotation and so on if we don't know that before. Without that, we would be hard to locate a specific class or understand how it works.

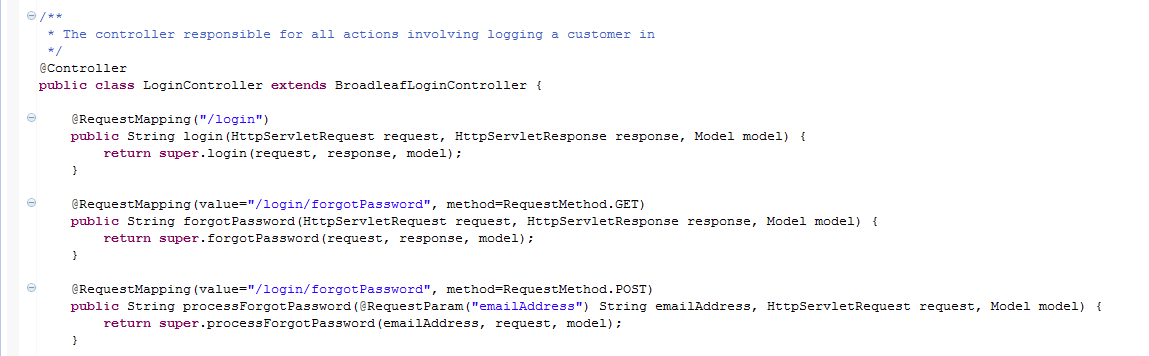


Figure 2. Annotation Example in login

And for the implementation, the project is big and most controllers are inheritance. Some algorithms are too complicated because these are too many if-else statements stacking in one method, as the above shows in the 5.2.1.1’s example. That's hard to read.

So according to the source code structure we can easy to read in module level, which is the benefit to the readability. But the high coupling with framework may hinder the reading if we don't know that before. And the size of the project and the complexity of source code must be a block to the readability due to the high usage of inheritance and the high level of complexity we calculated.

## Analysability

The analysability depends on the ability of tracking error or status. So the logging engineer is important to it. Normally we can get the logs from server log, database log and customized log API like Spring logback. Server log will record the exception java code console when system runs. According to their provided Javadoc which contains all the definition of all classes, we get to know most classes implement exception handler. So the server log will be benefit to analysability.

Except that, Broadleaf also integrates with Quality of Service (QoS) monitoring for vendor services. QoS does not interfere with a call to a vendor service (such as USPS), but does provide a way for that service to report on its status. Based on how to configure QoS, the maintainer can be notified of vendor service status changes in any number of ways. By default, Broadleaf will merely log vendor service status changes to the console. If this behavior is not sufficient, Broadleaf also provides support for email notifications for QoS status changes. So QoS also improve the analysability.