There are two main web servers that are widely used on the current software development. One is TomCat, and the other is Jetty. At the beginning of this semester, I used TomCat to publish a web application. It works fine, just a little complicated to publish a web application.

For this project, Clayon and Ellis recommended me to use jetty to set up a web server. I never heard it before, but after using it for one semester, I think Jetty is better than Tomcat.

Both of them are servlet engine and support Jave Enterprise. So, both of them can be wrote by Eclipse. Tomcat appeared before Jetty. At that time, there are so many web applications developed by individual. So Tomcat focus on provide busy connection. So it always consumes large amount of memory. But for Jetty, it is developed for light application. The memory consumption is much less than Tomcat, and it is very easy to start and maintain.

Here is the introduction for jetty. It is a web server servlet and to contain different web apps. It is developed by eclipse foundation. I think everybody is familiar with eclipse. This is an very powerful open source IDE which integrated many language compiler and syntax. So, Jetty can well cooperated with eclipse. For the friendly use and flexible structure, it has been used in many fields, like yahoo, Google App Engine, and so on.

The structure of Jetty is like MVC. We learned MVC in COMP 310, it is the same thing here - Controller, Model and View. In Jetty, controller is the connecter. The connecter is in charge of to receive the request of client. And pass the requester to model. Also, we can set the security thing rules or other utilities in the connecter to filter some requests. The view is about front end, it can be implemented by java script or Android web tool kit. Model is about handler. Basically, Jetty means the controller and handler.

This is the structure of handler; it looks very complicated, but actually it is very easy to use. All the things have been integrated perfectly. To use handler, just need to understand observable design patterns.