**Simple Usecase of Dropwizard, Websocket and broadcasters**

I have developed a simple use-case of Websocket, broadcasters and RabitMQ. One page as Admin page and other as User page. Whenever User page is connected/disconnected to Admin page, Admin page will display Connected/disconnected message.

**Environment :**

1. Guice for dependency injection.
2. Drop wizard : Dropwizard is kind of ecosystem which contains all the dependencies (such as Jersey, jackson or jetty) bundled into single package or can be added as separate module.
3. Atmosphere : The Atmosphere Framework is designed to make it easier to build Asynchronous Web applications that include a mix of WebSocket, Comet and RESTful behavior.

**Advantages of using DropWizard :**

1. Dropwizard is an open source Java framework for the rapid development of REST APIs.
2. If you are not using dropwizard, you will end up collecting all dependencies yourself and it often results into class loading issues due to version mismatch between various java libraries.
3. Dropwizard is a framework that bundles a few other frameworks and Java libraries needed for fast bootstraping of Java REST services, it includes in a package:
4. Jetty -> as a Servlet container and HTTP server. So you don't have to deploy your app to a standalone server, it is already embedded into your application which you normally start from a main method.
5. Jersey -> A JAX-RS implementation used for creating REST services with Java.(Used to register the resources)
6. Jackson -> de/serialization of JSON for your REST services.
7. Metrics -> for tracking your app performance.

**Advantages of using Guice :**

1. Guice is simple, easy to use, lightweight and focused in DI.   
     
   Spring is not only DI and became kind of heavy. If you are not using hibernate/jpa, quartz, AOP and many other tools that spring makes an easier use of (eg @Transactional), you might not need it.

So Guice increases application start time almost by half.

1. Guice allows the dependency injection configuration to be specified directly in Java code, whereas Spring requires the dependency injection bindings to be given in complex and verbose XML code.