Nk that

Future of Java EE By Andreas Schaefer (andy at madplanet dot com)

Does Java EE has a future or not. Currently the ????specification is growing and in the EJB3 spec. EJB2 was not retired requiring the Application Servers to support both. On the other hand Spring is taking advantage of the perceived complexity of Java EE and uses it to market their software. I think that Java EE is too ridged in its design and rather then offering services to components it forces components in its container molds. Still I do not want to get ride of the application server because the server make the applications???? manageable, clusterable and deployable. How about an application server where containers and services could be deployed as well, where components are not limited to the serve????r side and parts of components could run on the client side and where components could interact with the container rather than just being hosted by it. For example a ????MDB with a permanent failure cannot tell the container to stop sending messages and the EJB Timer Services only starts working after one instance of an EJB is created.

In my opinion the Java EE specification has or will soon reach???? is limits and many developers will loose interest in working with it. EJB3 with the annotations and the POJO EJB idea is a good indicator that something must be done but just to say that we need to enable every dummy to write Enterprise applications is not the right way. Java EE needs a new direction that enables good developers to write complexer applications with greater ease. ???? [edit]Adding JUnit and Coverage Tools to the Java Platform By Bob Evans (bobevans at agitar dot com)

Many language platforms, like Ruby and Microsoft .NET ship with a unit testing framework as part of the platform. Why not include JUnit in the Java Platform, or at least include it in the JDK? Code quality is a constant sore spot for commercial applications, so it seems like making ????the tools that contribute to higher quality more widely available will encourage better code. While we're at it, lets put in a code coverage tool as well, so we can see how well we're testing. We already have some profiling and management tools built in, so this seems like the missing whole in the picture.

I'd like to discuss this idea, and concerns around improving code quality with developer testing in general.