WhichContainer

"The Problem is Choice."

Containers, containers, containers. Boy there are a lot! It can be very daunting to figure out what container you need or should use. This page tries to give a fair answer. In my not so humble opinion, you should still do your own evaluation, rather than trust me to have the right answer.

It's assumed you're sold on java, know what a container is, and are otherwise a reasonably experienced programmer.

Decision-making for dummies

This questionairre doesn't take into account a ***huge*** amount of choices. It will most likely give you a suboptimal choice most of the time. However, mostly all of the projects (exception: geronimo) referred to are stable projects of rather high quality.

1 *Do you want or need to use a J2EE-like and/or compatible container?*

- **Yes**: use Apache's in-development Geronimo, or JBoss. They're not certified J2EE yet though. If you want something certified by sun, visit their website. Skip to 5.
- **No/don't know**: don't use J2EE. EJBs suck. Go to 2.
  2 *Do you want to write XML web services (ie, SOAP)?*
- **Yes**: use Apache's Axis and associated projects. Answer "yes" at question 3.
- **No**: good for you (XML is overrated). Continue right on.
  3 *Do you want to write server side web applications?*
- **Yes**: use a servlet engine, either Jetty or Apache's Tomcat. Skip to 5.
- **No/don't know**: don't use a servlet engine. Go to 4.
  4 *Do you want to run the container as a standalone (daemon, commandline tool or GUI) program?*
- **Yes**: You should evaluate loom. If you pick this, you can skip 5. If not, you'll have to write your own framework(which is not hard). Make sure to take a look at libraries like commons-cli and the java service wrapper to make that job as easy as possible.
- **No**: Go to 5.
  5 *Would you like to use a "framework" that makes some design decisions for you?*
- **Yes**: you need to pick one or two. That's the next question.
- **No**: you don't want a container at all. Go away. Do make sure to evaluate every link on this page and others, since you'll want to steal ideas from the container world.
  6 *Are you writing a web application? (ie one that speaks HTTP)*
- **Yes**: did you pick a servlet engine? If not, you might want to investigate using an embeddable webserver like simple. Go on to 7.
- **No**: good for you. You can skip a whole lot of container choices! Go to 9.
  7 *Are you going to be doing a lot of XML processing?*
- **Yes**: evaluate Apache's cocoon. You're done.
- **No**: good for you. More choices to make...
  8 *Would you like to use proven technology which everyone uses even if it means you don't get some fancy features?*
- **Yes**: evaluate Apache's Struts and Keel, the latter being based on Apache Excalibur's Fortress container. You're done.
- **No**: more choices!
  9 *Would you like a framework that offers things like AOP, a big suite of components, and similar things out of the box?*
- **Yes**: evaluate Spring Framework, especially if you're doing J2EE development. You're done.
- **No**: evaluate using Apache's Excalibur, PicoContainer or WebWork. You'll have significantly more work to do, but it may be worth it. You're done.

Feature-based decision-making

You might want to take a look at this container database.

Sensible decision-making

Look at that database referred to above and the process outlined above. A few things to look out for:

- **standards**. J2EE, SOAP, HTTP, ISO, Unified process. Figure out which standards matter to you and which projects support what.
- **project maturity**. Take a look at the number of releases made, the number of bugfixes, the documentation, who else is using it.
- **project activity**. Take a look at the bugtracker and the development mailing list. If there's no messages on the developer mailing list, its a bad sign.
- **developer pool size**. Be weary of "one-man codebases". Even with projects with many developers listed as active, some might actually not be. Investigate cvs commit logs and mailing list posting.
- **licensing and ownership**. Is (L)GPL (not) a problem for you, or do you (not) want BSD-style licensing? What about patent clauses? Is the project copyrighted by a company, a non-profit or not-for-profit, or one or more individuals?
- **code quality**. Find or obtain things like reports of unit tests, integration tests, stress tests.
- **user support**. Read the user mailing list. Do people provide useful answers to support questions? Evaluate the bug tracker. Do bugs get fixed? How quickly? Is there a track record of fixed bugs?
- **lock-in**. How easy is it to *stop* using a particular solution and replace it with something else? The easier the better.

and so on and so forth. Lastly...

- **featureset**. What do you *need* (no, you don't need a *fancy* persistence framework, AOP, lots of magic)? Do you need transactions, and are they available and easy enough? Do you need to connect to a particular database, and is that easy to do? etc etc.

Comments

The above is currently just the ideas of a single person. Feel more than free to add and correct. If you're not sure you're being objective enough, add your comments below instead of just changing the main text.