# Roadmap for 2.1

## Roadmap for 2.1

## Modularity

For a long time we've been on the brink of having an actually modular server, not just a conceptually modular server. As reflected in a lot of recent discussion on the dev list, I think we are so close we should really work hard to make this a pleasant reality. Some of the steps I see:

### enhance the plugin framework

enhance the plugin framework so bits of more config files can be added, in particular artifact_aliases.properties and config_substitutions.properties. IIUC Paul has some schema changes and xml handling rewrites in the wings as well

### plugable admin console framework

finish up the pluggable admin console work and actually split the admin console into appropriate bits for the plugins

### rearrange the build so it is organized by plugin (component/feature)

### actually assemble the servers we ship using plugins, perhaps using gshell scripts

### have more server-building tools.

For instance, a "button" to push that spits out a server with just what is needed for a set of apps and nothing else.

## Clustering

### Security

### jaspi

### triplesec

### administration

### beyond javaee security for jetspeed and roller

There are some good things about javaee security such as the idea of using Permissions and evaluating them in a policy but there are also a lot of limitations and problems such as no support for restricting access to user generated content that didn't exist when the security was originally configured. At least roller and jetspeed have run into this problem. I think triplesec may provide a fairly generic solution and it might be interesting to see if other projects are interested in this.

## Other apps

- roller
- jetspeed

- proximity etc

It would be great to get "all popular apps" available as geronimo plugins.

## Management and troubleshooting

### ARM

### "trace on error" facility.

Have a list of info that each component can add to as the request moves through the system. If there's an error, we can show the entire path taken with all data. Otherwise we discared it.
server farm management (gshell?)

> **jdillon**
>
> This is definitely on my "in my head" roadmap for the shell, though we need those clustering bits first to give it something to work with. One thing we might want to add before then is some kind of reboot facility to the server, which is possible using the GShell launcher process. So a user can install some muck, tweak some configuration, then tell the server to reboot and basically pick up a pristine configuration and working environment. The same basic mechanism could be used to create a rollback configuration, or named configurations, etc.

## Transaction manager

implement a "do work in a new tx" interface, hook it up to openjpa. IIUC IBM has proposed an interface that lets server pieces submit work to be done in a new transaction, thus eliminating the need to deal with suspend etc yourself. There's been some discussion on the openjpa lists, and we should definitely do this. There may be more commonj work to do also, but I've more or less lost track of that project.
make sure recovery actually works

## Core

### Logging Reform

We've talked about this here and there, most folks agree we need to reform our logging usage... but so far its yet to happen. It is a fairly simple task IMO, but its huge, since it pretty much touches almost everything. But its something that can be easily staged and then divided up amongst the surfs (and lords) for code weeding.

IMO, we need to...

1. Decide on slf4j or commons-cli
2. Define a general logging usage policy
3. Implement said policy on existing logging usage
4. Improve, add, augment, whatever, existing logging to be more useful (for users and developers)

For #1 I'm obviously for slf4j... for a few reasons which I've outlined in previous emails. I list this as the first step, as if we do go for it, then it has some slight impact on the policy and implementation work (like its varargs support instead of requiring log.isDebugEnabled() guarding).

IMO, this is low handing fruit, which we can easily pick and which our users (and developers) can feast upon the juicy flesh of... yummy.

### Better Spring application management

### OSGi-aware kernel

Investigate OSGI and figure out how it is different from what we are doing, what it can do better, and what is missing from it. Figure out an integration strategy whether it be "run OSGI as an application" or "replace the kernel with OSGI"

Quote from Felix's Karl Pauls:

"It makes sense to go down the "replace the kernel with OSGi" route. Most projects start with the other approach but in the end realize that it is a lot more work without many of the
benefits 🙂"

and further along this path

"Geronimo would be perfect as a starting point because it integrates a lot of other projects (hence, might "convert" them along the way)."

and the last but not least

"I'd be willing to help. Alas, my time is limited as well but I can provide help and insides where necessary."

and as a starting point

"At Apache Felix, we have developed a maven2 plugin that might be helpful in migrating to OSGi. Maybe that can be a starting point:

http://felix.apache.org/site/maven-bundle-plugin-bnd.html

Felix itself is very small (~300k) and can be easily embedded:

http://felix.apache.org/site/launching-and-embedding-apache-felix.html

See [Discuss] What next? for more information on the topic"

> ⚠ Don't break stuff if we start using OSGI more 🙂

### Annotation-based gbean configuration injection mechanism

### Figure out what to do with our "config.ser" in modules/configurations. At least get it into real xml, maybe using something like jaxb with schema/gbean.

> ⚠ **jdillon**
>
> OOOOH HEEEELLLL YA!!!!
>
> I'd really like to see the configuration for a module, in plain xml, based on some schema that is descriptive and extensible enough to handle pretty much any kinda of configuration we want to toss at a set of beans (or anything users want to, and they will come up with some crazy shiz for sure).
>
> And... I want to see those XML files deployed into the repository ASIS... IE... not in any *ucking jar/ear/war/whatever file.
>
> And, on a related not... those damn CAR files that are expanded... well, they shouldn't be. The repository should contain artifact **files** only... so if we need to unpack them on the fly, then do it. Or if users want to use an unpacked dir for their deployment, then use the deploy/* directory or something, but leave the repository/* as files.