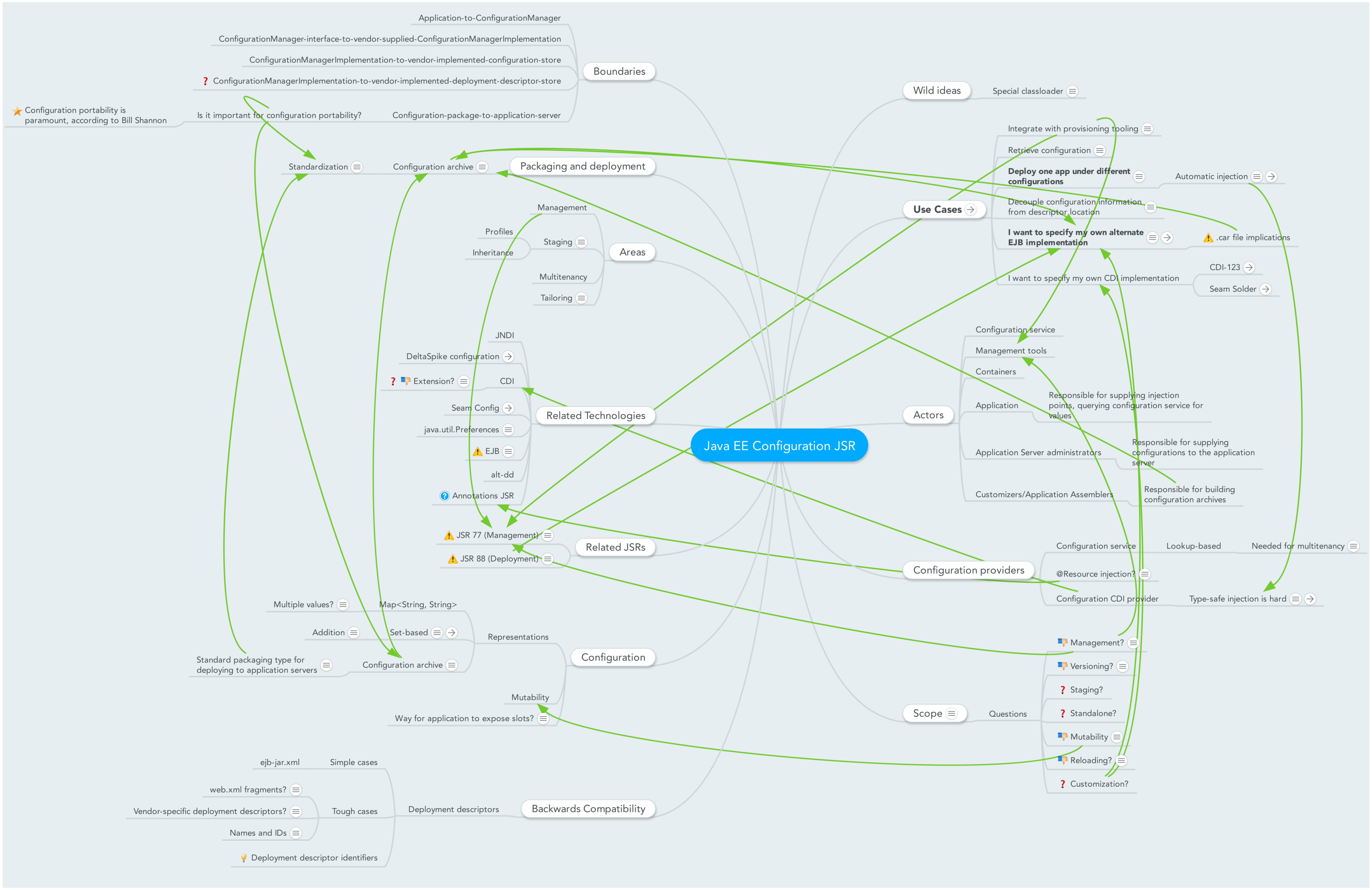
Java EE Configuration JSR
Boundaries
Application-to-ConfigurationManager
ConfigurationManager-interface-to-vendor-supplied-ConfigurationManagerImplementation
ConfigurationManagerImplementation-to-vendor-implemented-configuration-store
? ConfigurationManagerImplementation-to-vendor-implemented-deployment-descriptor-store
Configuration-package-to-application-server
Is it important for configuration portability?
Configuration portability is paramount, according to Bill Shannon
Packaging and deployment
Standardization
Configuration archive
Areas
Management
Profiles
Staging
Inheritance
Multitenancy
Tailoring
Related Technologies
JNDI
DeltaSpike configuration
CDI
? Extension?
Seam Config
java.util.Preferences
EJB
alt-dd
Annotations JSR
Related JSRs
JSR 77 (Management)
JSR 88 (Deployment)
Configuration
Representations
Map<String, String>
Multiple values?
Set-based
Addition
Configuration archive
Standard packaging type for deploying to application servers
Mutability
Way for application to expose slots?
Backwards Compatibility
Deployment descriptors
Simple cases
ejb-jar.xml
Tough cases
web.xml fragments?
Vendor-specific deployment descriptors?
Names and IDs
Deployment descriptor identifiers
Wild ideas
Special classloader
Use Cases
Integrate with provisioning tooling
Retrieve configuration
Deploy one app under different configurations
Automatic injection
Decouple configuration information from descriptor location
I want to specify my own alternate EJB implementation
.car file implications
I want to specify my own CDI implementation
CDI-123
Seam Solder
Actors
Configuration service
Management tools
Containers
Application
Responsible for supplying injection points, querying configuration service for values
Application Server administrators
Responsible for supplying configurations to the application server
Customizers/Application Assemblers
Responsible for building configuration archives
Configuration providers
Configuration service
Lookup-based
Needed for multitenancy
@Resource injection?
Configuration CDI provider
Type-safe injection is hard
Scope
Questions
Management?
Versioning?
? Staging?
? Standalone?
Mutability
Reloading?
? Customization?

@Resource injection?

Might have impact on annotations JSR; e.g. @Resource(configs = "fred, joe, default")

Addition

e.g.

Automatic injection

https://groups.google.com/d/msg/java-config/FrgH9_O1qkU/fW8aLxtplOEJ

As a developper I want to be able to configure something like this :

```
public MyPojo {
private String currency;
private Long currencyRate;

// complex algorithm based on the currency
}
```

How do I configure that depending if my app is dealing with dollars or euros ? The DeltaSpike way of doing is something like :

```
public MyPojo {
@Inject @Config("myCurrency")
private String currency;
@Inject @Config("myCurrencyRate")
private Long currencyRate;

// complex algorithm based on the currency
}
```

The Seam Config way of doing is quite clever. They use a namespace that is relative to the packing. So, without changing the initial code (so, forget about the @Inject above) they have an XML file that is able to override each attribute of any file.

CDI-123

https://issues.jboss.org/browse/CDI-123

Configuration archive

A configuration archive, if it is going to be standardized, needs to permit code and resources to be part of it.

Inciting use case is: I want to supply my own alternate implementation of a business interface as a stateless session bean.

Configuration archive

Must permit code to support alternate implementations of EJBs.

Decouple configuration information from descriptor location

Some standard way to access a configuration resource let frameworks / other JSR decouple from actual descriptor location, and option to store them externally with a large set of (vendor specific) options.

DeltaSpike configuration

http://deltaspike.apache.org/configuration.html

Deploy one app under different configurations

As a release engineer, when I have a package which is to be deployed to multiple environments, I want to be able to inject configurations into the package and then deploy.

EJB

Hazy thoughts about ejb-refs here.

If a configuration archive can have alternate code in it (i.e. I want to implement the Frobnicator interface using a different sort of EJB), then we have EJB JSR implications.

Extension?

Probably shouldn't investigate doing configuration service as a CDI extension because then everything would have to be a CDI bean archive. It is not clear this is the right thing.

I want to specify my own alternate EJB implementation

http://lairdnelson.wordpress.com/2013/06/21/how-to-override-an-ejb-reference/

I receive an .ear file with an EJB implementation of a business interface. I'd like to supply an alternate implementation and configure that EJB module to use mine instead.

(Argues for more than just docs in a .car file.)

Integrate with provisioning tooling

As an Ops, when I have to deploy a java application I want a standard way to define it's configuration, resources and environment to integrate with my provisioning tooling (Nicolas deLoof)

JSR 77 (Management)

Note the requirement that JSR-77 objects return their deployment descriptors as a String.

JSR 88 (Deployment)

Depending on whether use cases such as "I want to implement the Frobnicator using an alternate EJB implementation" are supported by way of configuration archives, there may be effects on this JSR.

Management?

Seems like a different JSR, but there is a lot of cry for this in the community.

Multiple values?

I don't see a need for a configuration (think: properties file, XML) to be exposed as a tree format. Maybe that's naïve.

Mutability

This seems like it should be omitted.

Some configuration, like env-entries, can change just fine, and when it does, listeners should be notified. These kinds of things don't need a redeploy.

Other kinds of things, like new code, should involve a redeploy. This kind of mutability must not be supported.

Names and IDs

Some deployment descriptors let you name an application or a module (e.g. <module-name>). So if we are going to link configuration to modules, we need to link the configuration to them carefully so that a module's name can still be configured. Module-name, in other words, is like a business key—we need to link configuration to Java EE modules by a surrogate key that is nonetheless predictable. META-INF/moduleId, perhaps, or MANIFEST.MF header?

Needed for multitenancy

There apparently needs to be a ConfigurationService that can be programmatically queried in order to support multitenancy, given that a tenant identifier is a runtime attribute.

Reloading?

Reloading should probably be impossible out of the gate as a redeploy would probably be necessary.

Retrieve configuration

I want standardized way (interface? maybe, but in broader sense of the word) get a configuration for already deployed application.

Scope

Always keep in mind: what should be possible, vs. what should be standardized?

For example, it should be possible to load configuration in different orders. But it should be standardized that there must be a prescribed order.

Seam Config

http://docs.jboss.org/seam/3/config/latest/reference/en-US/html_single/

Seam Solder

http://docs.jboss.org/seam/3/solder/latest/reference/en-US/html/introduction.html

Set-based

https://groups.google.com/d/msg/java-config/FrgH9_O1qkU/9hBKNZQ6fjgJ

Need ideas of:

Add and replace
Add as defaults
Full replace
Metadata-complete

Special classloader

For full backwards compatibility, I wonder if it would be possible to load a resource using a special Java EE classloader that was configuration-aware? That would probably provide a ton of backwards compatibility. EclipseLink could load its deployment descriptors, for example, that could be logical assemblages of several deployment descriptors.

Staging

Staging, e.g. "dev" as a shorthand for "all the settings used for development in my organization"

Standard packaging type for deploying to application servers

Seems unnecessary.

Standardization

Don't know if we want to standardize a configuration archive, or simply mandate that application server vendors make a "configuration" (whatever that is) available under a name for subsequent retrieval by the application by using that name.

Tailoring

Tailoring, e.g.

Type-safe injection is hard

https://groups.google.com/d/msg/java-config/FrgH9_O1qkU/fW8aLxtplOEJ

Because of the way CDI works, a @Producer has to produce a value of a common supertype. So there's no way to have a "generic" configuration producer, singular. Instead, several would have to be auto-registered based on discovered injection points.

Use Cases

https://groups.google.com/forum/#!topic/java-config/FrgH9_O1qkU

Vendor-specific deployment descriptors?

There's not really any way to mandate that a vendor upend its vendor-specific deployment descriptors. To process overrides this might be necessary.

Versioning?

This seems to be right out.

Way for application to expose slots?

IWBNI an application could report to the configuration service all of its "slots" that need filling. Think of EJB references.

java.util.Preferences

A few remarks about Preferences "synchronization hell", but otherwise seems to be the right sort of structure?

web.xml fragments?

Can't web.xml fragments be scattered all over the place? What do we do with these?