DSpace 2.0 Kernel

DSpace 2 Kernel

The DS2 (DSpace+2.0) kernel manages the start up and access to the core services in DS2. It is meant to allow for a simple way to control the core parts of DSpace and allow for flexible ways to startup the kernel. For example, the kernel can be run inside a single webapp along with a frontend piece (like JSPUI) or it can be started as part of the servlet container so that multiple webapps can use a single kernel (this increases speed and efficiency). The kernel is also designed to happily allow multiple kernels to run in a single servlet container using identifier keys.

Kernel registration

The kernel will automatically register itself as an MBean in when it starts up so that it can be managed. It allows startup and shutdown and provides direct access to the ServiceManager and the ConfigurationService. All the other core services can be retrieved from the ServiceManager by their APIs.

Kernel Startup and Access

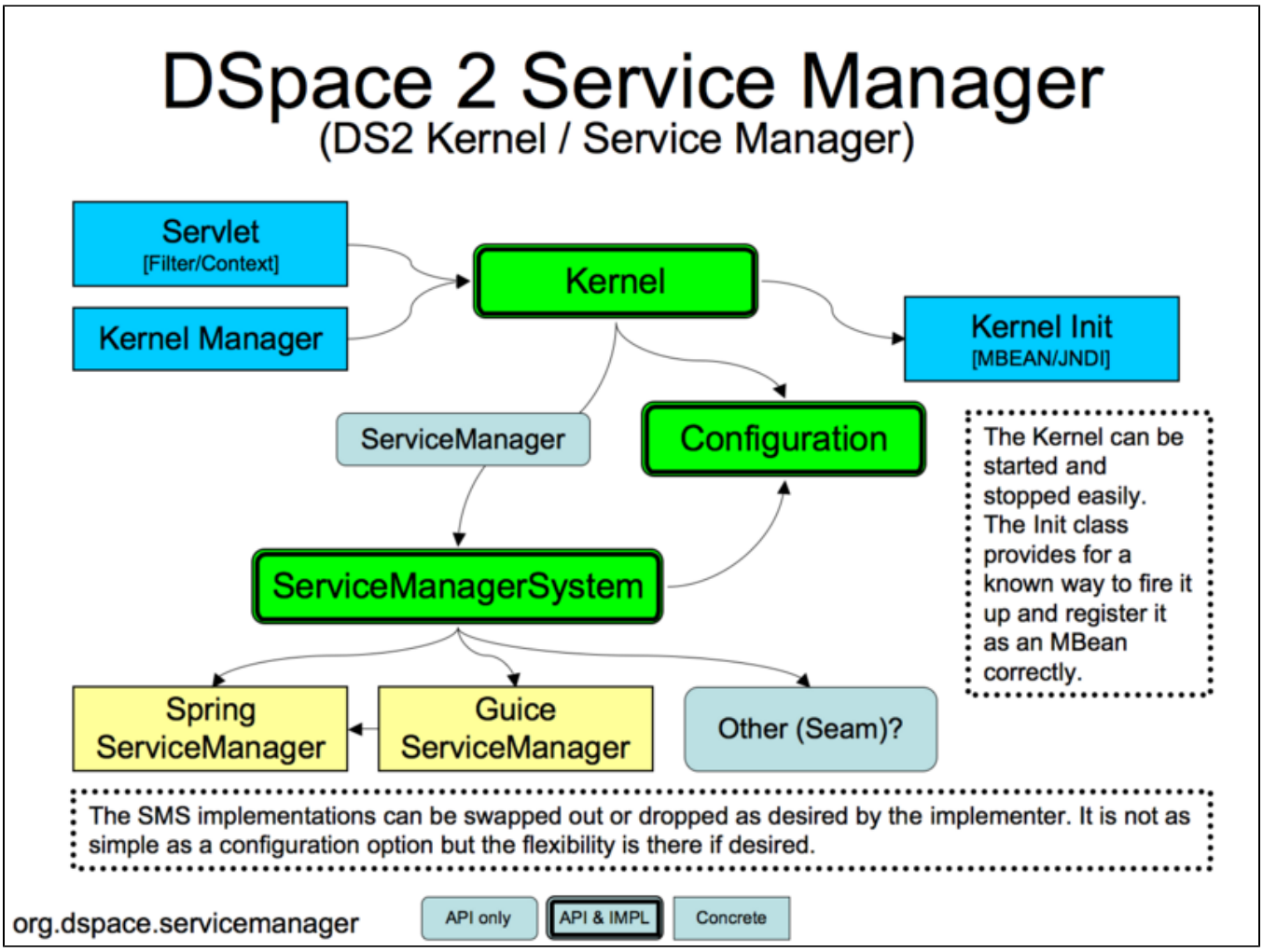
The kernel can be started and accessed through the use of Servlet Filter/ContextListeners which are provided as part of the DSpace 2 utilities. Developers don't need to understand what is going on behind the scenes and can simply write their applications and package them as webapps and take advantage of the services which are offered by DSpace 2. Access to the kernel is provided via the Kernel Manager and the DSpace object which will locate the kernel object and allow it to be used.

Configuration Service

The ConfigurationService controls the external and internal configuration of DSpace 2. It reads in properties files when the kernel starts up and merges them with any dynamic configuration data which is available from the services. The service allows settings to be updated as the system is running and also provides listeners which allow services to know when their configuration settings have changed and take action if desired. It is the central point to access and manage all the configuration settings in DSpace 2.

Service Manager

The ServiceManager abstracts the concepts of service lookups and lifecycle control. It also manages the configuration of services by allowing properties to be pushed into the services as they start up (mostly from the ConfigurationService).
The ServiceManagerSystem abstraction allows the DSpace ServiceManager to use different systems to manage it's services. The current implementations include Spring and Guice. This allows DSpace 2 to have very little service management code but still be flexible and not tied to specific technology. Developers who are comfortable with those technologies can consume the services from a parent Spring ApplicationContext or a parent Guice Module. The abstraction also means that we can replace Spring/Guice or add other dependency injection systems later without requiring developers to change their code. The interface provides simple methods for looking up services by interface type for developers who do not want to have to use or learn a dependency injection system or are using one which is not currently supported.



Testing

The DS2 kernel is compact so it can be completely started up in a unit test (technically integration test) environment (this is who we test the kernel and core services currently). This allows developers to execute code against a fully functional kernel while developing and then deploy their code with high confidence.

Providers and Plugins

For developers (how we are trying to make your lives easier):
The DS2 ServiceManager supports a plugin/provider system which is runtime hot-swappable. The implementor can register any service/provider bean or class with the DS2 kernel ServiceManager. The ServiceManager will manage the lifecycle of beans (if desired) and will instantiate and manage the lifecycle of any classes it is given. This can be done at any time and does not have to be done during Kernel startup. This allows providers to be swapped out at runtime without disrupting the service if desired.
The goal of this system is to allow DS2 to be extended without requiring any changes to the core codebase or a rebuild of the code code.

Activators

Developers can use an activator to allow the system to startup their service or provider. It is a simple interface with 2 methods which are called to startup the provider(s) and later to shut them down. These simply allow a developer to run some arbitrary code in order to create and register services if desired. It is the method provided to add plugins directly to the system via configuration as the activators are just listed in the configuration file and the system starts them up in the order it finds them.

Provider Stacks

Utilities are provided to assist with stacking and ordering providers. The priority is handled via a priority number such that 1 is the highest priority and something like 10 would be lower. 0 indicates that priority is not important for this service and can be used to ensure the provider is placed at or near the end without having to set some arbitrarily high number.