ArchReviewFrameworks

Contents

Survey of frameworks that can be used to support plugability and/or modularity

This document is "In-Progress" please feel free to contribute coments and material.

Past Efforts in this area: WebApplicationFrameworks

OSGi

What is OSGi?

Over the past few years it has specified a Java-based service platform that can be remotely managed. The core part of the specifications is a framework that defines an application life cycle model and a service registry. Based on this framework, a large number of OSGi Services have been defined: Log, Configuration management, Preferences, Http Service (runs servlets), XML parsing, Device Access, Package Admin, Permission Admin, Start Level, User Admin, IO Connector, Wire Admin, Jini, UPnP Exporter, Application Tracking, Signed Bundles, Declarative Services, Power Management, Device Management, Security Policies, Diagnostic/Monitoring and Framework Layering. Wikipedia

OSGi is a standard, there are a number of Open Source implementations, a few known ones listed here.

OSGi Implementations

Apache/BSD Licensed

- Apache Felix Apache Felix is a top level project within The Apache Software Foundation (ASF). It is a community effort to implement the OSGi R4 Service Platform, which includes the OSGi framework and standard services, as well as providing and supporting other interesting OSGi-related technologies. The ultimate goal is to provide a completely compliant implementation of the OSGi framework and standard services and to support a community around this technology. Felix currently implements a large portion of the OSGi release 4 specification, but additional work is necessary for full compliance. Despite this fact, the OSGi framework functionality provided by Felix is very stable. The actual OSGi compliance of Felix can be see here. There is a dutch commercial venture that offers commercial support based on the official release.

- Knopflerfish Knopflerfish is an Open Source product, however its is not fully R4 compliant (whatever thats worth) There is a commercial venture that supports a fully complaint "Pro" commercial version

Eclipse (Common Public License)

- Eclipse Equinox OSGi Probably the most known usage of OSGi, Eclipse OSGi is the backbone of the Eclipse Platform and such, the largest userbase. Equinox R4 compliance status can be seen here Equinox has a Server Side OSGi set of packages that can be run embedded in an Servlet Container or run independently with Servlets embedded into it.

OSGi Platforms

Taking a slightly broader look at the application stack beyond a single technology implementation, there are ready built platforms that provide much of the infrastructure that we are looking at.

Eclipse (Common Public License)

- Nuxeo Nuxeo provides various components, that come together to provide a rich platform for enterprise content management systems (and potentially, other applications of a similar type). The initial release was of the Nuxeo Runtime, that is the foundation of the infrastructure, providing an OSGi-based component model, deployable on JBoss, JBossAS and Eclipse RCP (with other adaptors to come), and an extension model that allows components to define extension points. This has been followed up with Nuxeo Core, that provides content repository support based on JSR-170. In mid-November, Nuxeo Enterprise Platform will add jBPM and JRules integration with document management and transformation services.

Further information on Nuxeo can be found here: Roadmap
Background
Enterprise Platform features

Graham Triggs 12:14, 13 November 2006 (EST)

Benefits of OSGi for Web Applications

The OSGi HTTP Service is a Servlet Container on which servlets can be deploy dynamically and programmatically. To give an example of how this could benefit DSpace, because te servlet can be deployed dynamically into the container, it doesn't actually have to reside in the webapplication war file or be described in the web deployment descriptor (web.xml) thus in a modular DSpace world, a module could deploy servlets into the main DSpace webapplication context without any need to do a build that combines them all together. simply have your module distributed in an OSGi bundle (a.k.a. a Java Jar with a little extra manfest info) and delopy them into the OSGi Framework and tey come up in the webapplication. – MarkDiggory 10:17, 11 November 2006 (EST)

While OSGi's ability to deploy and undeploy these bundles very dynamically is interesting, its more the method of deployment that is of value to a DSpace Application which will provide a means to keep the DSpace modules very separate from one another at compile time, bringing them together to go on the server only at the time of deployment. So we get two things out of OSGi that are of the greatest benefit to DSpace, (1) a clear deployment pattern and (2) a clear separation of concerns for modules. – MarkDiggory 10:17, 11 November 2006 (EST)

The Spring Framework

What is Spring?

Spring is a layered Java/J2EE application framework, based on code published in Expert One-on-One J2EE Design and Development by Rod Johnson (Wrox, 2002).

Spring includes:

- The most complete lightweight container, providing centralized, automated configuration and wiring of your application objects. The container is non-invasive, capable of assembling a complex system from a set of loosely-coupled components (POJOs) in a consistent and transparent fashion. The container brings agility and leverage, and improves application testability and scalability by allowing software components to be first developed and tested in isolation, then scaled up for deployment in any environment (J2SE or J2EE).
- A common abstraction layer for transaction management, allowing for pluggable transaction managers, and making it easy to demarcate transactions without dealing with low-level issues. Generic strategies for JTA and a single JDBC DataSource are included. In contrast to plain JTA or EJB CMT, Spring's transaction support is not tied to J2EE environments.
- A JDBC abstraction layer that offers a meaningful exception hierarchy (no more pulling vendor codes out of SQLException), simplifies error handling, and greatly reduces the amount of code you'll need to write. You'll never need to write another finally block to use JDBC again. The JDBC-oriented exceptions comply to Spring's generic DAO exception hierarchy.
- Integration with Toplink, Hibernate, JDO, and iBATIS SQL Maps: in terms of resource holders, DAO implementation support, and transaction strategies. First-class Hibernate support with lots of IoC convenience features, addressing many typical Hibernate integration issues. All of these comply to Spring's generic transaction and DAO exception hierarchies.
- AOP functionality, fully integrated into Spring configuration management. You can AOP-enable any object managed by Spring, adding aspects such as declarative transaction management. With Spring, you can have declarative transaction management without EJB... even without JTA, if you're using a single database in Tomcat or another web container without JTA support.
- A flexible MVC web application framework, built on core Spring functionality. This framework is highly configurable via strategy interfaces, and accommodates multiple view technologies like JSP, Velocity, Tiles, iText, and POI. Note that a Spring middle tier can easily be combined with a web tier based on any other web MVC framework, like Struts, WebWork, or Tapestry.

You can use all of Spring's functionality in any J2EE server, and most of it also in non-managed environments. A central focus of Spring is to allow for reusable business and data access objects that are not tied to specific J2EE services. Such objects can be reused across J2EE environments (web or EJB), standalone applications, test environments, etc without any hassle.

Spring's layered architecture gives you a lot of flexibility. All its functionality builds on lower levels. So you can e.g. use the JavaBeans configuration management without using the MVC framework or AOP support. But if you use the web MVC framework or AOP support, you'll find they build on the configuration framework, so you can apply your knowledge about it immediately.

The Spring-OSGi project

The Spring-OSGi project makes it easy to build Spring applications that run in an OSGi framework. A Spring application written in this way provides better separation of modules, the ability to dynamically add, remove, and update modules in a running system, the ability to deploy multiple versions of a module simultaneously (and have clients automatically bind to the appropriate one), and a dynamic service model. Spring-OSGi

Quote form Spring developers:

At present we are working hard to ensure sure Spring 2.0 plays nicely in an OSGi environment, something that is very important to us and our users. In addition, a large amount of effort is being invested in our documentation. The return is evident when you open the 2.0 reference manual and see just how comprehensive the coverage is. – http://www.infoq.com/articles/spring-2-update

- External Articles
  - Atlassian Article http://opensource.atlassian.com/projects/spring/secure/attachment/11934/spring_and_osgi-64.html

Combining Spring and OSGi Server Side

Spring-OSGi on the server side has been a recently and rapidly emergent area of developer interest.

Spring OSGi Tutorial

OSGi Serverside tutorial

The Spring OSGi Experience

Spring plugins ala Eclipse

So what about Cocoon and Manakin?

Searching for research in the area of integrating Cocoon, Spring and OSGi, please post it here – MarkDiggory 08:53, 13 November 2006 (EST)

- Subproject in Cocoondev: http://cocoondev.org/main/117/43.html
- Presentation at OSCOM: http://www.oscom.org/events/oscom4/proposals/spring.html