Chapter 1. Global Design

Table of Contents

1.1. Context of Use

The JASMINe EoS console, part of JASMINe Monitoring, is deployed in an J2EE application server like JOnAS. Its use is bound to the application server and to the system that run it like Windows, Linux or Mac. The console GUI is done via a web browser supporting the Flash technology.

The goal of Kerneos is to give dynamicity to the module management, i. e. allow the loading of modules without stopping the server. Another goal is to provide a system to ease the development of a new Kerneos module or a new application. The first targetted application is the JASMINe EoS console.

1.2. Console Modularity

To make the JASMINe EoS dynamic, the OSGi framework seems to be a good choice, because it provides a specification for module lifecycle. OSGi has become a standard used by many application servers like JOnAS, Glassfish or JBoss.

1.2.1. OSGi

OSGi reduces complexity by providing a modular architecture for today's large-scale distributed systems as well as small, embedded applications. Building systems from in-house and off-the-shelf modules significantly reduces complexity and thus development and maintenance expenses. The OSGi programming model realizes the promise of component-based systems.

1.2.2. Flex

In order to get a dynamic management of modules inside JASMINe EoS console, the framework used to communicate between the client part and the server part must be able to manage its configuration dynamically.

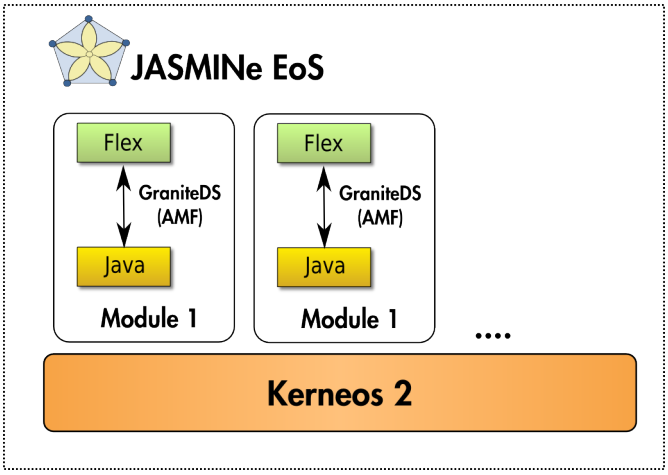
Kerneos uses GraniteDS to allow communication between Flex modules and remote objects.

1.3. General Architecture

JASMINe EoS is composed of modules and its Kerneos console that manages the lifecycle of the modules. As shown on the following picture, each module is a separate archive composed

of a client part (Flex) and a server part (java). Communication between the two parts is achieved by the GraniteDS framework. The data flow format is AMF, used to swap data between Flash and java.

Figure 1.1. JASMINe EoS architecture



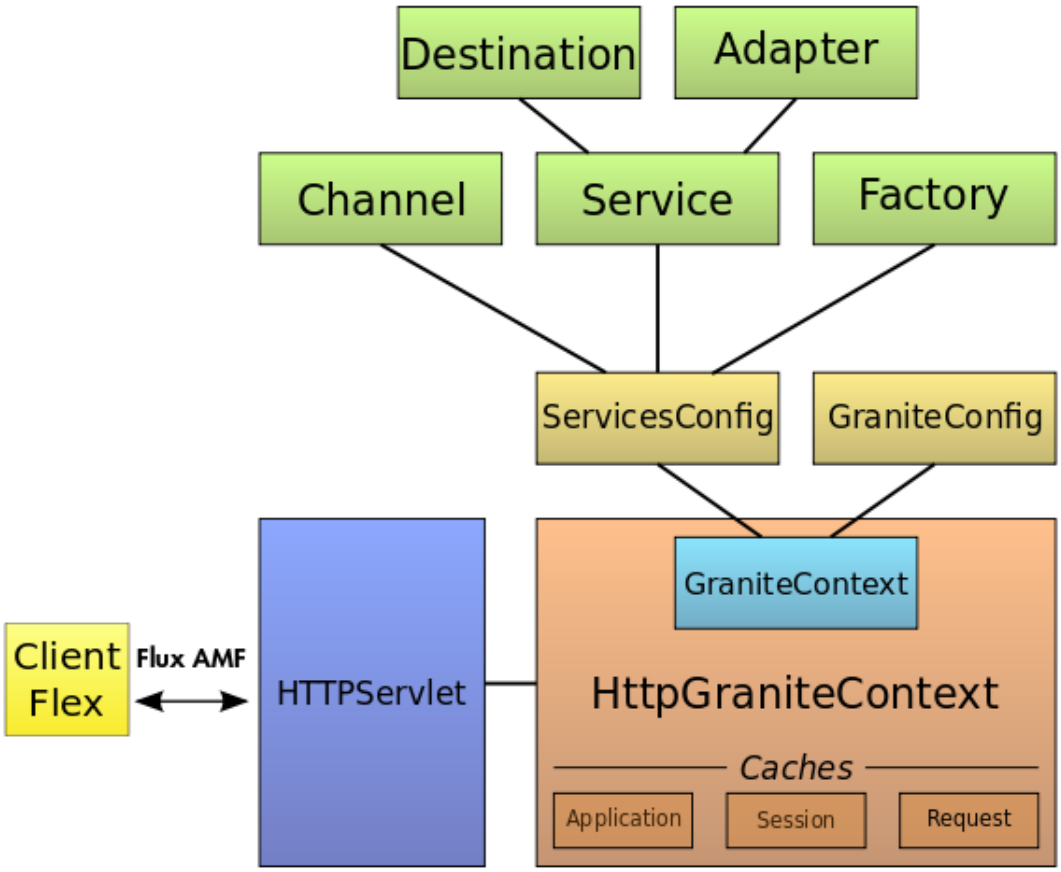
1.4. GraniteDS

Granite Data Service is a framework used to develop and deploy Rich Internet Applications (RIA) based on Flex and JavaEE. GraniteDS is an open source solution under LGPL2 licence. It is an alternative to the Adobe solution: BlazeDS.

1.4.1. GraniteDS architecture

To understand how GraniteDS works, it is important to understand all the objects implied.

Figure 1.2. GraniteDS architecture



Adapter	Allow to define a new Object that can be used with a Destination to fill a particular need.
Destination	This object describes the different properties associated to the Destination like the class name, it belong to a Service and depend on one or more Channels.
Channel	This Object can be use to communicate between the client part and the server part. It holds a description of access type and the URL used to communicate.
Factory	Factory that will create Objects
GraniteConfig	Holds the properties specific to GraniteDS.
GraniteContext	Holds GraniteConfig, ServiceConfig, and information about Context flow.
HttpGraniteContext	Implementation of GraniteContext, with information about the caches.
HttpServlet	Standard interface for managing HTTP requests.
Service	Aggregate Services and Adapters used for a certain type of Message.

ServiceConfig Configuration of all Channel, Factory, and Service objects.

1.4.2. Detail of an Operation

1. The servlet receives the AMF data sent by the Flex client with the Destination.
2. Data is deserialized to a Request
3. We get the Factory associated to the request
   a. Search the Destination
   b. Got the Factory class name
   c. Look in the cache for a such Factory
   d. If not found, create a new instance and put it in the cache.
4. We get the ServiceInvoker matching the Factory
   a. Search Destination object
   b. look for the ServiceInvoker in the cache
   c. If not found, create it and put it in the cache.
5. Invoke the ServiceInvoker
6. Build a reply depending on the result of the call.
7. Serialization of the reply
8. Send back AMF data to the client

1.4.3. GraniteDS OSGi

Figure 1.3. Comparison between GraniteDS and GraniteDS OSGi