Sakai

Below is a description of Sakai

Sakai OAE exposes APIs at a number of levels. There are the standard Java APIs expressed as interfaces, describing service isolation, deployed as OSGi bundles within an OSGi1 container. We use Apache Felix. These APIs operate at conceptual levels such as Content, Activitites, Social Connections, User and Groups.... to name a few.

At the next layer out there are service contracts expressed as Message formats as a result of internal events. Those generally contain pointers back to the Resource that caused the event, with metadata to indicate state before and after the event as well as the operation that triggered the event. We use the OSGi Event service for initial event propagation since its efficient at high volume synchronous and asynchronous events. One of the listeners to the OSGi Event Service is a JMS producer that bridges the event onto JMS (using ActiveMQ in Multi Master mode) and allows the event to propagate out of the JVM process over the network. The OSGi-JMS bridge is configured to selectively propagate events to control network bandwidth consumption. It generally used where we believe there may be some off App Tier processing or there is an interest in capturing the event for warehouse or data processing purposes. An example of an OSGi event consumer (other than the OSGi-JMS bridge) is the indexing service that maintains a Solr index in near real time. Examples of JMS consumers are Content preview services, email messaging delivery.

Further out the app server processes expose RESTfull interfaces. These are almost entirely of the flavour expressed by Roy Fielding2, and mostly conform to the protocol expressed by Apache Sling3. Like Atom, we deal with URLs (URIs) that point to Resources, and then perform actions on those resources based on the HTTP method and the parts of the URL that are not the URI to the Resource itself. Our RESTfull interfaces are rarely bound to fixed locations (eg not /db/record/12312342) and have meaning (eg /user/ieb/profile.json or /pyhsics/course101.allcontent.json). Unlike Atom we use json almost exclusively.

The aim of OAE was to be a full SOA architecture using fast targeted app servers to service a UI largely written client side (JQuery). Those app servers had to be fast (1ms http round trips on most requests) to ensure that the UX was responsive. Behind that the app servers would off load everything they could outside the request cycle to backend processing on servers dedicated to a single purpose (content preview generation of PDFs, a cluster of Solr servers). Where there was already a viable service implementation that could be used the App servers were intended to proxy request (with protocol translation if required) through to those service implementations. The aim always to minimise the number of CPU cycles and bandwidth required to respond to a UI request. We didn't make much use of traditional SOA transports, like SOAP, preferring to use lower cost message passing (restful JSON).

1 http://en.wikipedia.org/wiki/OSGi
2 http://roy.gbiv.com/untangled/2008/rest-apis-must-be-hypertext-driven
3 http://sling.apache.org/site/url-decomposition.html