ALT-2006. Update to Research Paper: The WAFFLE bus: a model for a service-oriented learning architecture.

1. Introduction

It is 6 months since the original WAFFLE bus paper was written and significant progress has been made, especially in relation to the Service-Oriented Virtual Learning Environment (SOVLE) component. This document describes the latest SOVLE design.

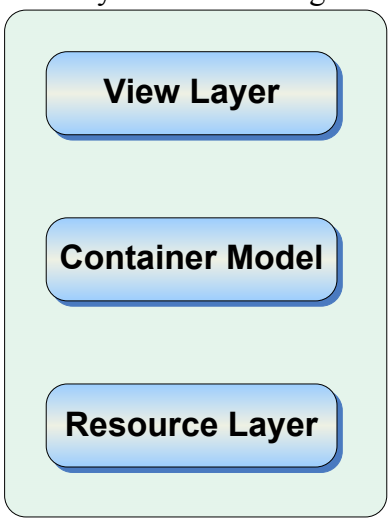
2. SOVLE Features

The JISC ELF Toolkit project, Service-Oriented Consumer Kit for ELF Tools (SOCKET) has helped illuminate the possibilities that arise from a VLE built on service-oriented architectural principles. Using mature technology and standards, we have produced a design for a SOVLE that is characterised by the following features.

- SOVLE can act as a complete VLE solution, or as a services gateway for existing VLEs.
- A completely configurable container/resource model: buildings, floors and rooms, for example; or modules and courses; or any combination.
- Each user can have a personal VLE (like a personal Bodington Building).
- Resources can be components or Web services (consumers or providers).
- A homogeneous management layer exists for both components and services.
- Every resource is a plugin.
- Resources can be hot deployed & undeployed, and dynamically re-configured.
- The capability is present to manage a distributed Service-Oriented Learning Architecture (SOLA) based on an Enterprise Service Bus (ESB) methodology and secured by the Guanxi security system.
- Multichannel access is fundamental to design: PCs, portals, mobile devices.
- Decimated development cost compared to typical monolithic VLE – only the framework is being built: all resources are imported.

3. Implementation

The SOVLE conforms to what software architects call a Model-View-Controller (MVC) design pattern. This maximises the separation of software concerns into the three layers shown in Figure 1.



Brief summaries of these layers are given below.

3.1 View Layer

Suitable candidates for the SOVLE view technology are Cocoon, JavaServer Pages (JSP), and Freemarker, all of which were investigated in the course of the SOCKET project. (Note that each individual plugin resource is free to make an independent choice on the view technology – there are no restrictions.)

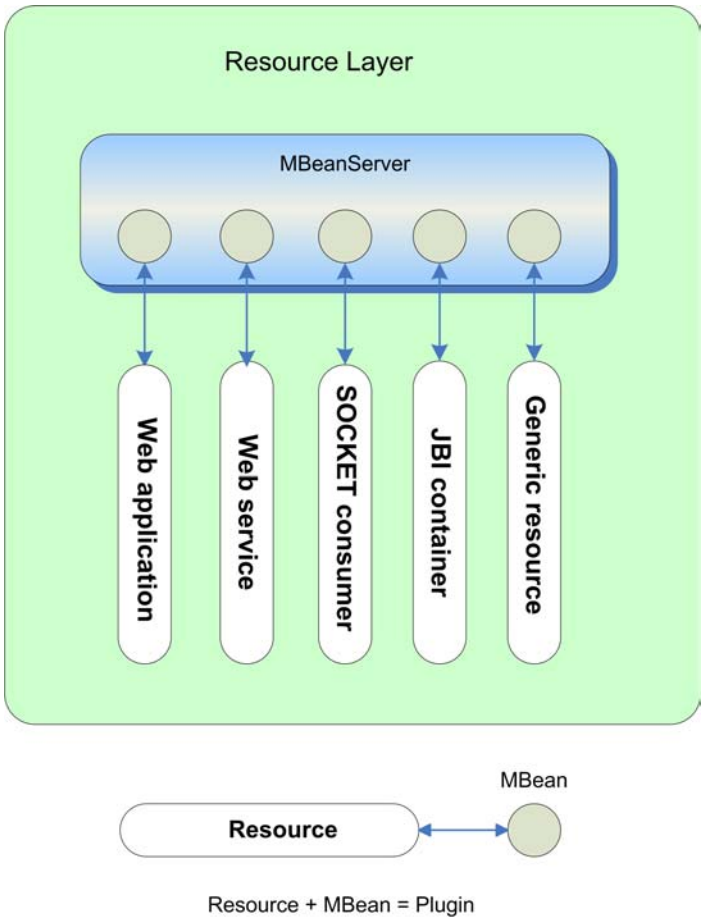
Figure 1. Three-layer structure.

3.2 Container Model Layer

Each SOVLE view consists of the contents of one (or more) containers. Each view is described by an xml file conforming to a group- or role-dependent container configuration schema. This can be passed to the view technology directly (Cocoon or Framework, for example) or as a JavaBean (JSP).

3.3 Resource Layer

Examples of possible types of SOVLE resources are Web applications, Web service providers, SOCKET service consumers, Java Business Integration (JBI) containers, and generic resources (Figure 2). Web applications can be exposed externally as services or Web Services for Remote Portlets (WSRP) using Web service proxy objects.



JBI is the standard Java technology for the integration of Web services. Eventually, SOCKET services will also be implemented inside a JBI container. The JBI objects are the basic units ("service containers") of a distributed Enterprise Service Bus (ESB) system. Message mediation such as XSLT transformation and Web service workflows is accommodated as part of the JBI architecture.

Figure 2. Plugin structure of resource layer.

Each resource is associated with at least one Java Management Extensions (JMX) Management Bean (MBean). The MBean exposes the functionality of the underlying resource, an Application Programming Interface (API) for applications, or Service Programming Interface (SPI) in the case of services. Fine-grained access to a resource is achieved by mapping group information onto application roles.

All MBeans are registered with an MBeanServer object which becomes the runtime service registry for the SOVLE through which a management application can control and configure the system.