LAMS 2 Architecture

Introduction

In LAMS 1.0.x, there is no separation between tools and core services. Although, LAMS 1.0.x is a modular application (with four modules: Authoring, Monitor, Administration and Learner), if a new tool is implemented, different pieces of code must be added to all four modules. In addition, once the new code is added, the whole LAMS system must be recompiled and redeployed (see diagram below).

While LAMS 1.0 already has a large number of activity tools to create and run sequences (about 17 activity tools at the end of 2005), adding a new activity tool into LAMS was a complex task which involved a good knowledge of all main four modules. Even if the source code is available for people to review and use, it was very difficult to create a new tool from scratch.

Additionally, we were always asked *"I have this sensational assessment engine that my university has bought for a 10 million dollars, but it doesn't do the sequencing that LAMS does. Can you guys integrate it as a tool in LAMS?"*. Unfortunately, the answer was: *"Well, it's not very simple"*. And truly it isn't, as it requires changes to all modules and an interface to the assessment tool, take care of authentication/authorization issues, etc.

LAMS 2 implements a modular architecture where tools and learning activities can be added on-the-fly to a LAMS 2.0 server. In order to provide such modularity, LAMS 2 implements a **Tools Contract**. The Tools contract is a set of expected behaviours and APIs that each tool has to implement to communicate with LAMS Core. The LAMS Core has modules for Authoring, Monitor, Administration and Learner.

The new architecture proposes a clear and defined separation between tools and core service responsibilities/functionalities.

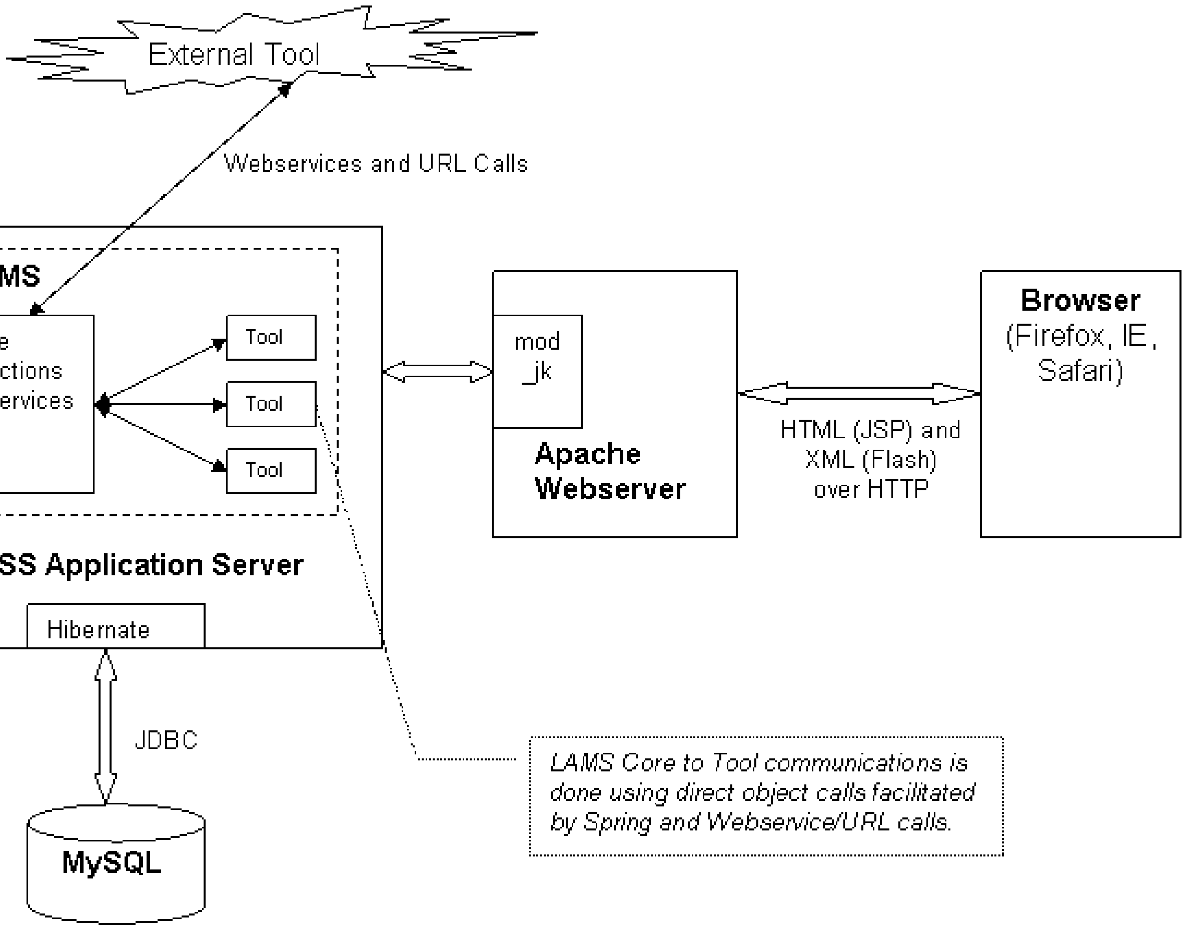
As you can see in the diagram above, LAMS Core is mainly the same main modules we previously have but now LAMS 2 has **abstracted** the interface to *"talk"* the activity tools.

Activity tools are now (almost) completely independent web applications that interact with LAMS core modules and services through the Tools Contract. This contract establishes the expected behavious and APIs that each tool will have to implement to be instanciated as a LAMS activity tool.

Using this Tool Contract, not only LAMS tool can be *"LAMS Tools"* but also -in principle- external tools that implements it can become LAMS Tools.

Component Technologies

The diagram below illustrates the various technologies used in LAMS and how the different components communication. Apache is not required for LAMS however it is common in large deployments.


External Tool
Webservices and URL Calls
LAMS
Core Functions & Services
Tool
Tool
Tool
JBOSS Application Server
Hibernate
JDBC
MySQL
mod_jk
Apache Webserver
Browser (Firefox, IE, Safari)
HTML (JSP) and XML (Flash) over HTTP
LAMS Core to Tool communications is done using direct object calls facilitated by Spring and Webservice/URL calls.