

# Project Management Server using Java Server Faces

Anil Sharma – Founder and Chief Architect, Vertex Logic, Inc  
[www.vertexlogic.com](http://www.vertexlogic.com)

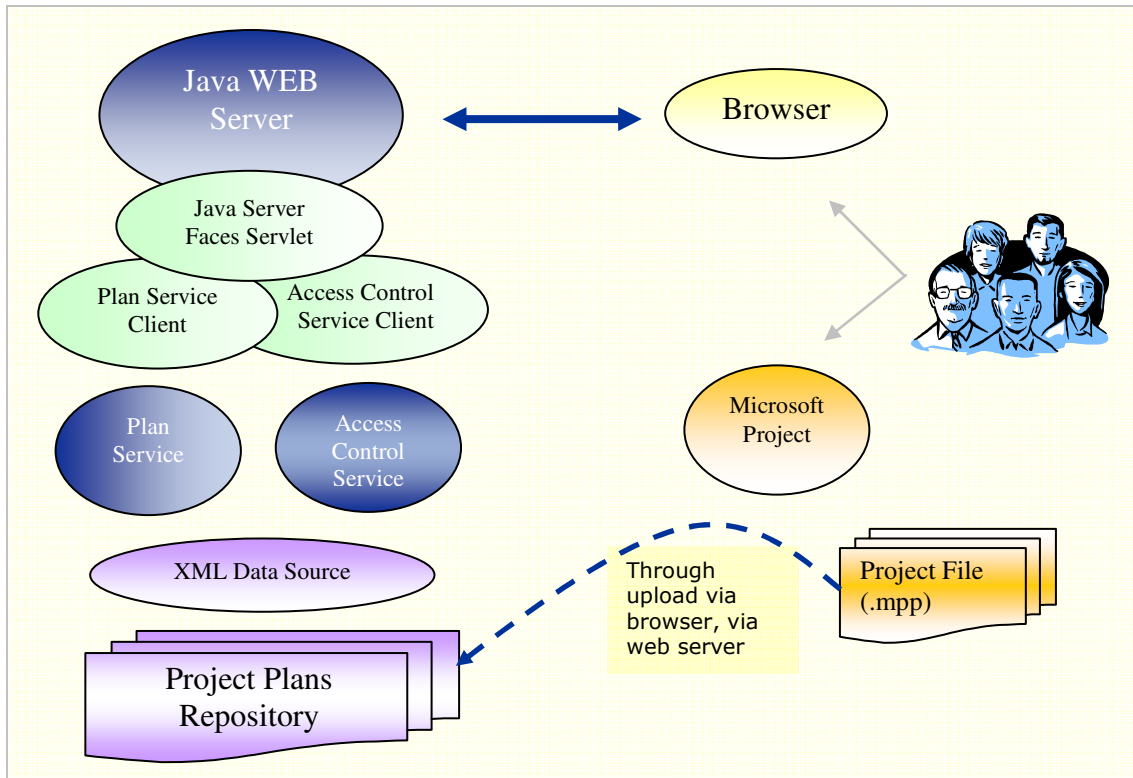
This article discusses how we used Java Server Faces (JSF) to build a scalable and secure (role-based access control) system. Following the principles of Service Oriented Architecture (SAO), we have divided the application into a set of services and used service-clients as backing beans of JSF. The experiences discussed in this article are based on the development of a commercial software system called eManagement PM.

eManagement PM is a project management server for organizing and managing project plans created using desktop applications such as Microsoft Project. It builds a central repository of the project plans and enables team-collaboration around it. Users can view status reports, update plans, add notes or raise issues.

## ***Application Components and Architecture***

The following is the list of the components used to build the system. The diagram below shows the architecture.

- Java WEB Server (Tomcat)
- Java Server Faces
- Java RMI services
  - Plan Service – for the management of plans
  - Access Control Service – for role-based access control and user management
- JSF Backing Beans
  - Plan Service Client
  - Access Control Service Client
- XML Data Source
- A set of business objects
  - Plan
  - UserTask
  - Problem etc.
- JSP Pages built using JSF tags



Components of eManagement PM

## ***Project Management Applications – Functions***

Since our intent is not to discuss the application, therefore we will limit our discussion of the product feature-functions to a typical scenario.

- Project Manager creates a project plan using Microsoft Project and imports it into eManagement server using a WEB interface (uploads the file to the server).
- Project plan is stored as XML files in the eManagement repository.
- Team members access project plans using WEB browser to view status reports, update it, raise issues etc.

eManagement pm									
User: Anil									
List of Plans Team Members My Profile Help Logout									
Change Plan Start Date Edit Plan Export Plan Derivative Plan Plan Projection									
Available Plans	Details of Project Management Plan								
Project K2	All Tasks		Active Tasks	My Tasks	Overdue Tasks	Completed Tasks	Top 10	Issues	
Marketing_campaign	<b>Id</b>	<b>Description</b>	<b>Resource</b>	<b>Status</b>	<b>%</b>	<b>Start Date</b>	<b>End Date</b>	<b>Predecessors</b>	<b>Action</b>
Project_Management_Plan	1	Project Management Plan		inactive	0 %	07/01/2003	09/03/2003		Notes(1)
Power_Generator	1.1	Initiating		inactive	0 %	07/01/2003	07/02/2003		Notes(1)
New_Product	1.1.1	Project Initiation		inactive	0 %	07/01/2003	07/02/2003		Notes(1)
	1.1.1.1	Identify goals and objectives	Project Director	ready	0 %	07/01/2003	07/01/2003		Notes(1)
	1.1.1.2	Document project costs and benefits	Application Architect, Project Manager, Technical Architect	inactive	0 %	07/02/2003	07/02/2003	1.1.1.1	Notes(1)
	1.1.1.3	Develop project charter	Project Manager, Acceptor	inactive	0 %	07/02/2003	07/02/2003	1.1.1.2	Notes(1)
	1.2	Planning		inactive	0 %	07/03/2003	07/31/2003	1.1	Notes(1)
	1.2.1	Define Scope		inactive	0 %	07/03/2003	07/11/2003		Notes(1)
	1.2.1.1	Develop strategies and plans	Project Director	inactive	0 %	07/03/2003	07/03/2003		Notes(1)
	1.2.1.2	Conduct planning workshop	Project Director	inactive	0 %	07/04/2003	07/04/2003	1.2.1.1	Notes(1)
	1.2.1.3	Research previous experience	Project Manager	inactive	0 %	07/03/2003	07/03/2003		Notes(1)
	1.2.1.4	Define scope	Project Manager	inactive	0 %	07/07/2003	07/07/2003	1.2.1.2, 1.2.1.3	Notes(1)
	1.2.1.5	Develop high-level work breakdown structure	Project Manager	inactive	0 %	07/08/2003	07/08/2003	1.2.1.4	Notes(1)
	1.2.1.6	Build detailed work breakdown structure	Project Manager	inactive	0 %	07/09/2003	07/09/2003	1.2.1.5	Notes(1)
	1.2.1.7	Document project costs and benefits	Application Architect, Project Manager, Technical Architect	inactive	0 %	07/11/2003	07/11/2003	1.2.1.6	Notes(1)
	1.2.1.8	Specify deliverables and acceptance criteria	Project Manager	inactive	0 %	07/10/2003	07/10/2003	1.2.1.6	Notes(1)
	1.2.1.9	Document assumptions	Project Manager	inactive	0 %	07/10/2003	07/11/2003	1.2.1.4, 1.2.1.5, 1.2.1.6, 1.2.1.7, 1.2.1.8	Notes(1)
	1.2.2	Document Project Support Plans		inactive	0 %	07/23/2003	07/24/2003	1.2.1	Notes(1)
	1.2.2.1	Document risk management plan	Project Manager, Acceptor	inactive	0 %	07/23/2003	07/23/2003		Notes(1)
	1.2.2.2	Document configuration management plan	Project Manager, Acceptor	inactive	0 %	07/23/2003	07/23/2003		Notes(1)
	1.2.2.3	Document data management plan	Project Manager, Acceptor	inactive	0 %	07/24/2003	07/24/2003		Notes(1)

## Why Java Server Faces?

JSF enables easy manipulation of the state of a UI page by separating rendering from the application logic. The rendering becomes a mere depiction of the state and JSF does a beautiful job of it. This also means reducing an application problem to manipulation of state variables and data structures. We have a rich meta-model in the backend. JSF reduced our problem to the manipulation meta-variables or application data according to the application logic.

For example, we have a requirement –

*“When a task is marked as completed, user should not be able to edit its end date and/or percentage-completed value.”*

JSF makes it easy to satisfy this requirement –

*“The application logic sets the editable meta-property of ‘end date’ and ‘percentage completed’ attributes to false, when the state of the task is changed completed.” We achieved the required result by tying ‘editable’ meta-property to “rendered” attribute of the tag.*

Conversely, JSF has contributed to the richness of our meta-model. Since we did not use scripting with JSF tags, we had to rely on the pre-computed state variables for the rendering of UI. It means we were automatically led to a more expressive model. Since, it is always easy to change the business rules of an expressive system; this has turned out to be a blessing in disguise.

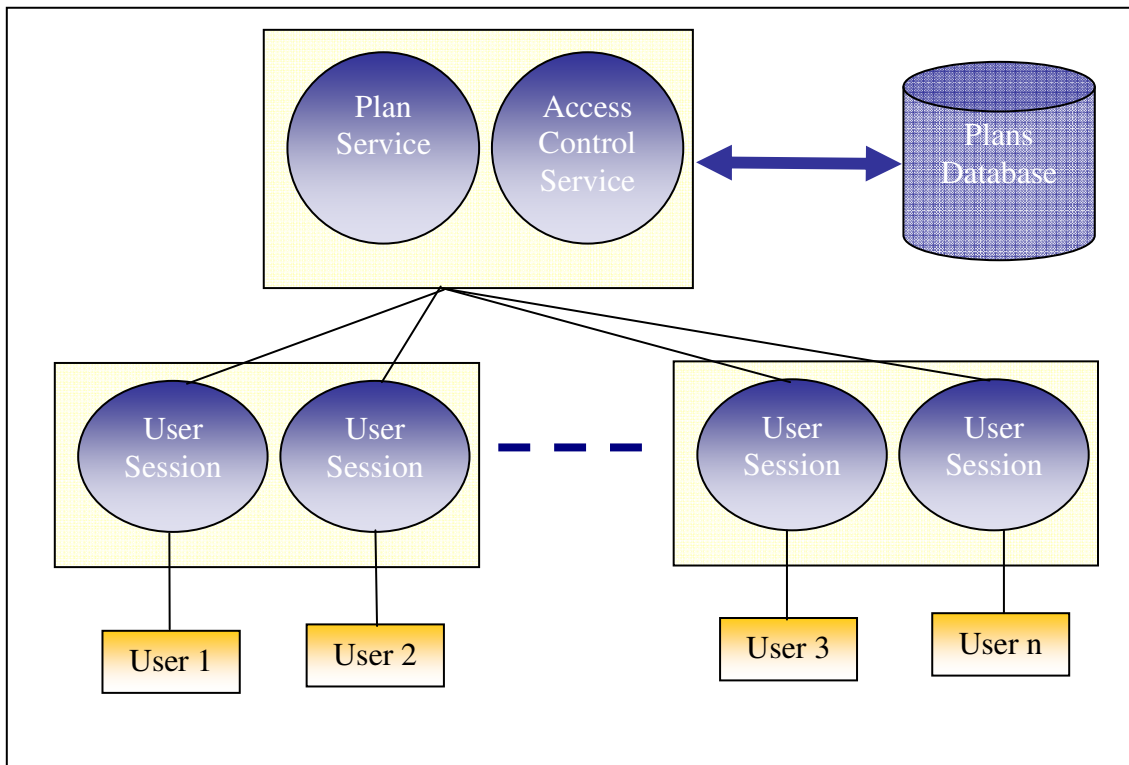
## Role of Backing Beans

We have divided the backend into a set of services and used service-clients as backing beans. These beans also act as the container of other fine-grain beans such as business objects (a value object). The backing beans also play the role of stateful session objects. Since a plan is a complex object and its client view can significantly differ from the server image during a user session, the stateful backing beans managed the client view of the plan until it was saved by the user into the service repository. The backing bean sends only changes to the service to reduce the traffic between the two.

## Scalable Architecture

The system is scaled up by distributing client sessions on a cluster of hardware. This is possible due to following reasons:

- Some of the heavy lifting is delegated to the backing beans.
- The traffic between the central service and the backing bean is limited by sending only modifications.



## **Role-based Access Control**

The access control rules are enforced at two levels. The access control rules affecting user display are enforced by the backing bean. JSF made it easy and is achieved by tying the 'editable' and 'visible' meta-properties to 'rendered' attribute of the tag. The values of these properties are computed based on the privilege of the user and object ownership.

### ***Wish-list***

The following list is due to limitations of JSF or our own ignorance.

- JSF should generate events upon entry and exit from a lifecycle phase. This will be useful for the application logic. For example, we have to make some decisions based on the value of a hidden-component in the response phase. The decision is made in the 'get' method of the corresponding backing bean attribute. Since the 'get' method is also called during the 'restore' phase. We are using some tricky programming to ascertain that the intended logic is executed in the response phase.
- JSF should allow the passing of parameters to action methods. Using hidden properties to pass parameters makes the debugging difficult.
- Can JSF cache the view and not restore each time? We feel it will lead to better performance. We haven't dug into the details of its implications; therefore it is an open question.
- JSF should allow expressions as value. Especially we could not figure out how to specify negation of a boolean – example, render a component when 'editable' is false.

```
< ... rendered="#!{planClient.plan.access.editable}" ...>
```

### ***Downloading eManagement PM***

An evaluation copy eManagement PM discussed in this article is available for download from [www.vertexlogic.com](http://www.vertexlogic.com).