FULCRUM
Logic

Rapid Implementer of Technologies

SAAS

Best practices for SAAS implementation using an Open Source Portal (JBoss)

Introduction

JBoss Portal is a very popular open source portal offering from Red Hat. It is JSR-168 compliant and provides support for deployment in a clustered environment with high-availability. The JBoss Portal framework provides support for features like customizable UI tier using themes and skins, creating workflows using a process server and also supports SOA.

In this white paper we attempt to explore various design approaches to implement a SAAS application using JBoss as the portal vendor.

Some of the key considerations in implementing a SAAS solution involve:

1)   Addressing MultiTenancy

2)   Providing a unique user experience

3)   Keeping user/tenant data separated and secure

4)   Having a SOA based architecture and using Web Services based messaging for the middle-tier or business layer.

For the purpose of this white paper we will consider the scenario of a sample timesheet application that will be hosted on JBoss.

FULCRUM
Logic *Rapid Implementer of Technologies*

MultiTenancy

Multi-tenancy refers to the ability to host a software solution that serves multiple client organizations or tenants. Its scope includes:
1) Application customization
2) Data isolation
3) Security isolation

Portlet design considerations for multitenancy

Application customization:

A portal implementation offers a lot of features which can be leveraged to provide a customized feel for every tenant. Each portlet can be made configurable for each subscriber. To enable a high level of reuse, the degree of configurability in the portlets must support subscriber-specific settings in a range from name-and-value pair configurations, such as subscriber IDs or subscriber's service endpoints, to subscriber-specific look-and-feel. JBOSS can provide a clone-and-configure approach to the implementation and deployment of the portal in multi-tenant applications i.e. the ability to clone a portal and configure it. JBoss like many other portal vendors supports the concept of a virtual portal which is a logical copy of an existing portal utilizing the same hardware and software resources.

FULCRUM Logic

Rapid Implementer of Technologies

Follow the steps to create a virtual portal in JBoss shown in the figure below

Figure 1:

Figure 2:

Figure 3:

FULCRUM Logic *Rapid Implementer of Technologies*
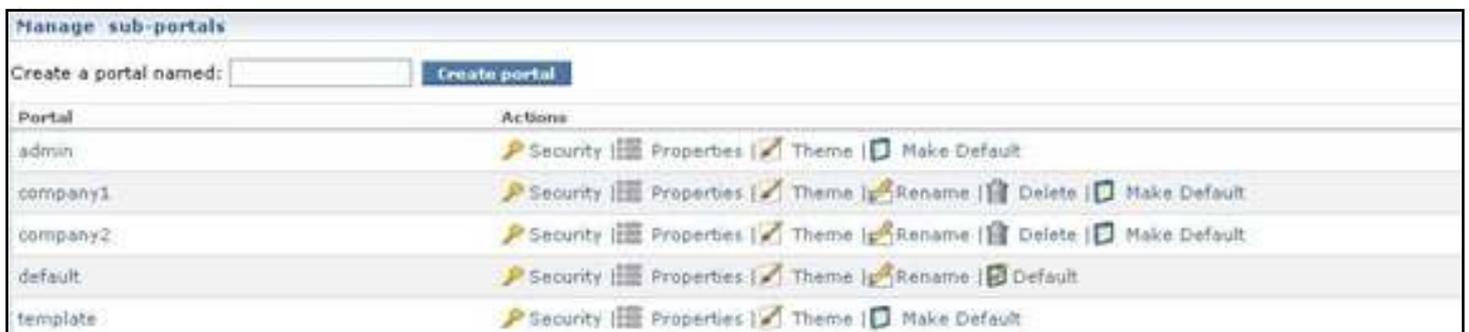
Security Isolation:

Each virtual portal supports isolation of user populations for each tenant bank through a multi-tenant LDAP tree structure in a single instance of a directory Server. A sample scenario as outlined by folks at IBM involve the following steps which can also be recreated in JBoss

1) Create a multitenant user directory structure in LDAP by:

 - Create a realm for each tenant i.e. a separate tree hierarchy starting at dn [dc=tenant1, dc=com]

 - For each security realm, a new partition and security context entry must be created in the server.xml configuration file in Apache Directory Server (Apache Directory Server Install Directory / instances / default /conf /server.xml)

2) Dynamic LDAP routing with Spring security

 The basis of dynamic LDAP routing hinges on the possibility of dynamically selecting the LDAP security context at run time based upon a lookup key. In a multitenant environment, this translates to authenticating and authorizing against an LDAP source that is derived on the fly based on the tenant's ID.

Providing a unique user experience:

Unique User Experience is providing a different User Interface to different users as per the application scope. Unique User Experience consists of look and feel , structure and scope for every user role.

Using the model proposed by Jesse Richard Dalton we attempt to show how a Unique User Experience (UUE) can be provided to tenants using Jboss.
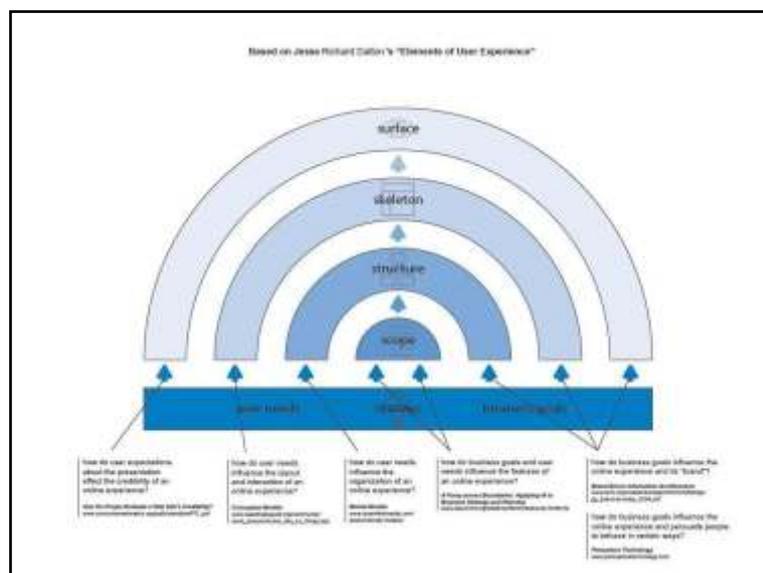

Figure 4: Elements of User Experience

JBoss provides a lot of features which can be utilized to create a distinct look and feel that is also consistent with the organization branding. Also it provides flexibility to the tenant to customize the look and feel based on their specific business requirements.

At a high level providing a UUE can be divided into the following 4 S's:

1. Surface

2. Skeleton

3. Structure

4. Scope

Surface is the client-facing element directly visible to the client. This layer is supported by themes and skins of the portal application. Themes and skins control the overall look and feel of all the pages in a Portal. Changes to them are made via updating stylesheet classes provided by JBoss. Themes and skins can be set at a virtual portal level. They can also be set at a user level based on the user role and access. This can be done dynamically in the application or can be set by an Admin using the portal's administrative console.

Skeleton is responsible for ensuring that the page architecture and layout are as per the user needs and can be customized. A UUE in this context can be achieved through providing different layout options of the page. By default there are three types of layouts: - right, center and left.

Structure is the next element of UUE. Structure differentiates a site or application as per the organization, line of business and user needs. A virtual portal provides a quick way to achieve this objective.

Virtual Portals are logical portals that share the same hardware and software installation. Virtual portals can also specify a default locale through portal's administrative console to implement internationalization functionality for organizations that require different languages to cater to their distinct global locations.

Scope includes access control and roles. Portal provides access functionality at the page level accessibility for user role and user group. For example hiding certain sections or links on a page from anonymous users or displaying content relevant only to a specific user group. Above that Portal also provides functionality at the application.

Per the JSR 168 spec, portlet has 4 modes view, edit, config and help. View and help modes can be available for all registered users or anonymous users. Edit mode can be available to registered users and configuration mode is available only to administrator. These modes provide another level of customization at the portlet level.

The screen shots below illustrate this for our sample scenario of a timesheet application.
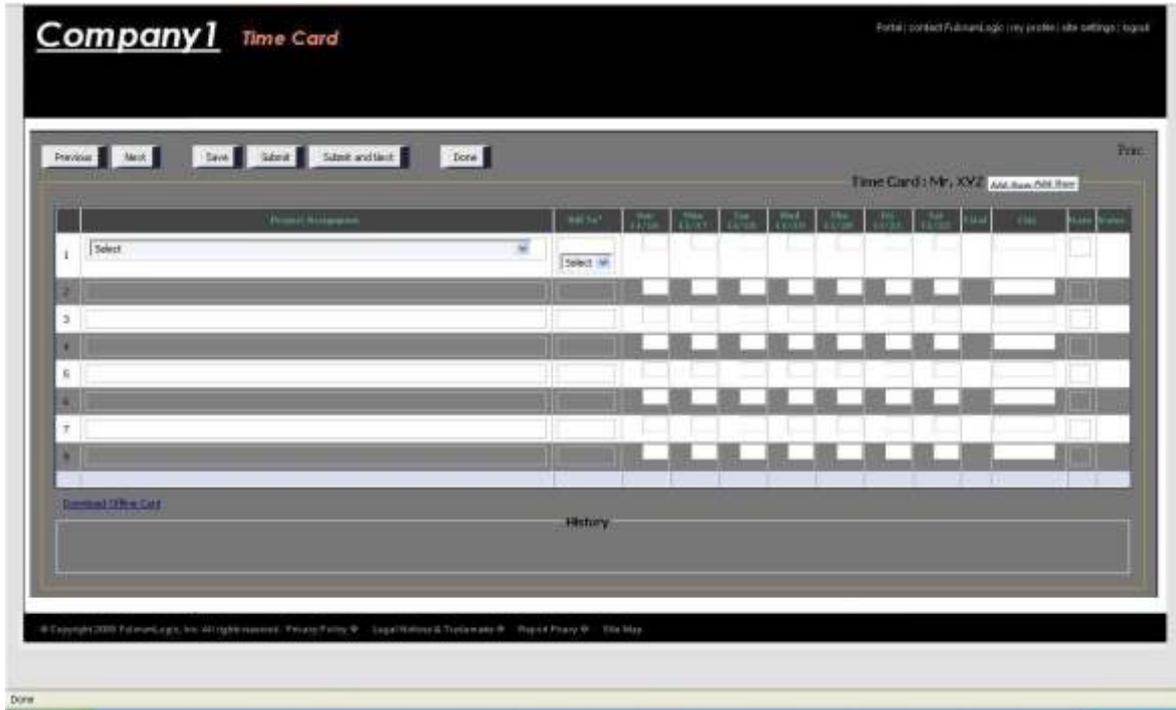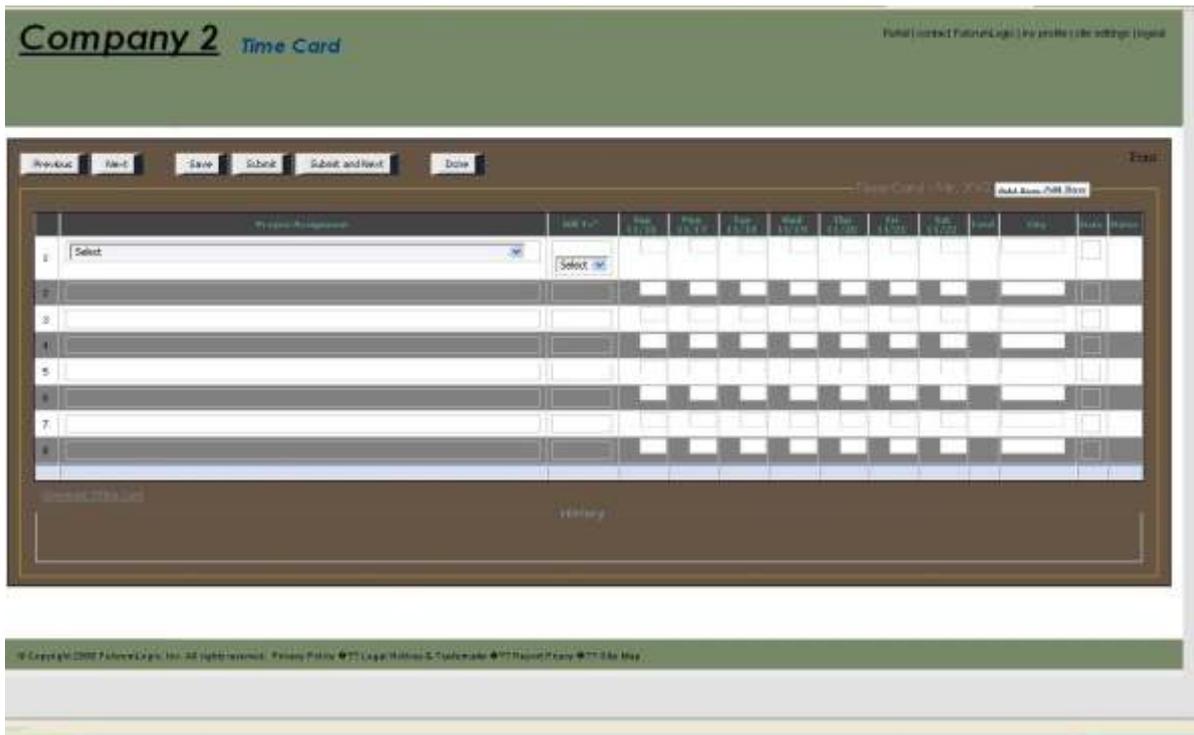
Company 1:

Figure 5: Layout for Company 1
Figure 6: Layout for Company 2

Separate and Secure Data Tier:

Implementing a Multi-Tenant database:

Key considerations while designing the multi-tenant database are consolidation, scalability and extensibility. Each tenant can have its own extensions for base tables as per the requirement, forming the separate logical schema for that tenant. Query transformation can be used to map these single-tenant logical schemas to one multi-tenant physical schema. Available physical memory can limit the scalability of the database.

The simplest approach to implement multi-tenancy would be to create private database structure for each tenant. In this approach the database structure is replicated and customized according to the tenant's requirement. Accordingly query transformation becomes very simple but it puts overhead to physical memory as the number of tables keeps growing when new tenants are added. Hence this approach gets expensive in terms of memory.

Second approach could be to use Universal tables. Universal table is a very generic structure containing data for all the tenants. Table schema includes the tenant column and the large number of generic columns considering requirements of all tenants. Tenant column is used to identify which tenant the row belongs. Columns from the logical source tables of each tenant are mapped to generic columns in universal table which allows different tenants to extend same table in different ways. Keeping all values altogether in one table avoids the reconstruction overhead. The disadvantage of this approach is that rows are unnecessarily wide containing null values for those generic columns which do not belong to the tenant.

Base-Extension table layout can be the better approach to overcome the above mentioned disadvantages. Here extensibility can be achieved by splitting the universal table in base and extension table. Base table's schema will have minimum required generic columns those used by all tenants. A separate extension table is created for additional requirements of particular tenant. This extension table can be shared by other tenants having the same extension requirement, which avoids creating unnecessary tables with the same schema.

To map the extension table to base table, a global Extension Mapping table is used with columns, Tenant_Id and Extension_Table. Extension_Table column returns the name of the Extension table to refer. To avoid creating a separate mapping table for each base table and its extension tables, one more column, Ext_Mapping is added to the global Extension Mapping table as well as the Base table. Ext_Mapping column value in Base table indicates the extension for that tenant. If null, there is no extension of base table

for the tenant and so no mapping required. Tenant_Id and Ext_Mapping columns in the Global extension mapping table are used together to get the value of the corresponding extension table for that tenant.

For our sample scenario of a Timesheet application there are 4 tables i.e. Tenant, Project, Timesheet and Employee. Example below explains the mapping for Project and Timesheet tables.

Tenant

Tenant_ID	Tenant_Name
T1	Tenant1
T2	Tenant2
T2	Tenant3

Project

Tenant_ID	SN	Proj_ID	Proj_Name	Client	Ext_Mapping
T1	PRJAA001	P1	AAA	Client1	Project
T1	PRJAA002	P2	BBB	Client2	Project
T2	PRJAA003	P1	CCC	Client1	Project
T3	PRJAA004	P1	XXX	Client1	Project

Project_Ext_Tenant1

SN	Manager
PRJAA001	Mark
PRJAA002	David

Project_Ext_Tenant2

SN	Manager	Domain
PRJAA003	John	Banking
PRJAA004	Peter	Logistics

Timesheet

Tenant_ID	SN	Date	Emp_ID	Proj_ID	Hours	Ext_Mapping
T1	TMSAA001	10-5-2008	E1	P1	8	Timesheet
T1	TMSAA002	11-5-2008	E1	P1	7	Timesheet
T2	TMSAA003	10-5-2008	E77	P1	8	Timesheet
T3	TMSAA004	10-5-2008	E34	P1	6	Null

Timesheet_Ext_Tenant1

SN	Billable	Task_Description
TMSAA001	Billable	Change analysis
TMSAA002	Non-Billable	Fixing layout issues

Timesheet_Ext_Tenant2

SN	Task_Description	Location
TMSAA003	Portlet development	Pune

Extension_Mapping

Tenant_ID	Extension_Table	Ext_Mapping
T1	Project_Ext_Tenant1	Project
T2	Project_Ext_Tenant2	Project
T3	Project_Ext_Tenant2	Project
T1	Timesheet_Ext_Tenant1	Timesheet
T2	Timesheet_Ext_Tenant2	Timesheet

SOA implementation for Middle tier:

The below figure describes about implementing SOA that has been designed by identifying the reusable assets. The two tiers in the middle are a medium for the interaction between the client tier and the database tier and by doing so it improvises the efficiency and scalability and even minimizes the effect and scope of changes. It has been categorized into four tiers.

Figure 10 :
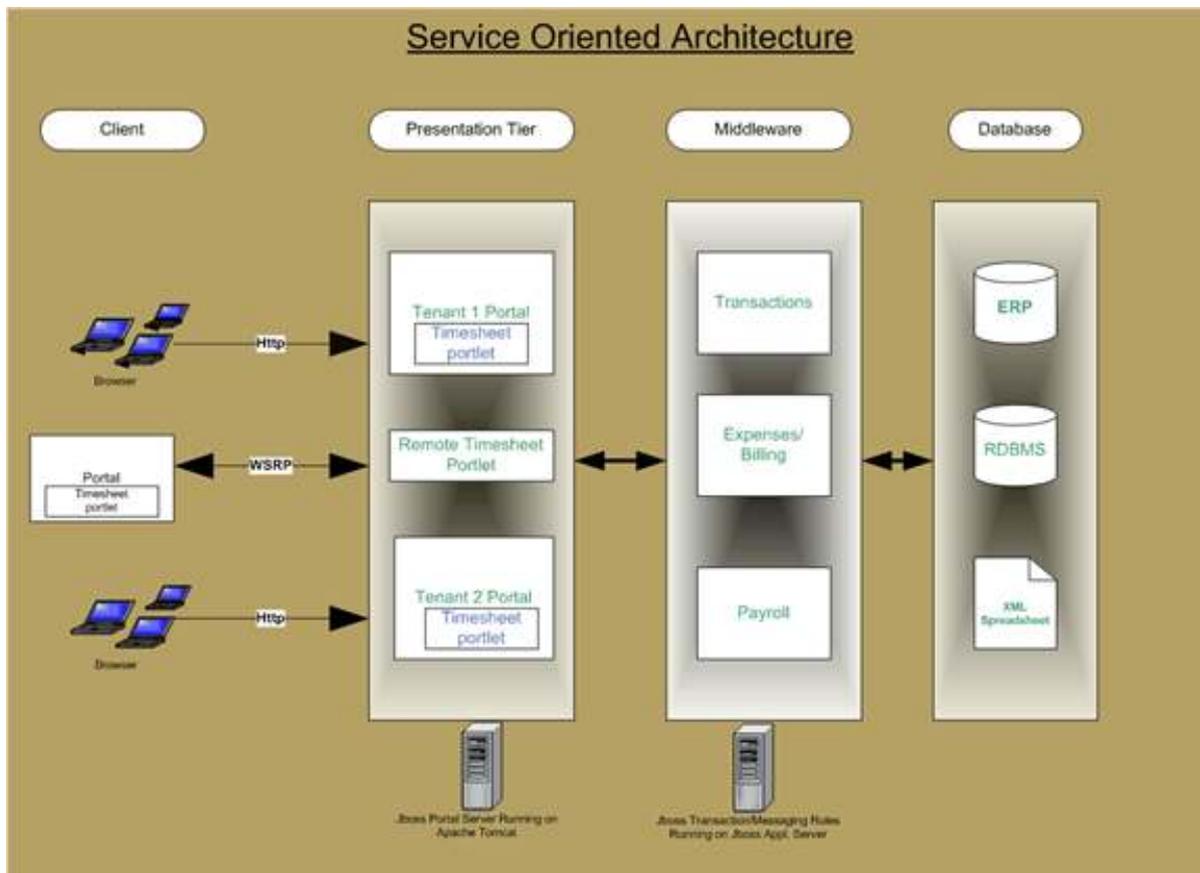
1) Client Tier   2) Presentation Tier   3)Middleware and   4)Database Tier

The Client Tier is responsible for interacting with the user. It renders Http/WSRP requests.

The Presentation Tier has all the available portlets along with remote portlets.

The client tier can talk to the Presentation Tier using Http or WSRP (Web Services for Remote Portlet).

We have our Middleware as the Business Tier where all the components are being placed like (Transactions, Expense/Billing and Payroll). The Presentation Tier may have to change depending on the client requirement but it would insulate the Middleware from any changes. The Business Tier is responsible for implementing the business services and making them available to the Presentation Tier. Its also responsible to interact with the Database Tier. We can also implement JAXB for reading XML related data.

The Database Tier is one where the shared enterprise resources such as database systems, XML spreadsheet etc. reside.

JBoss also provides support for WSRP which stands for Web Services for Remote Portlets. JBoss Portal provides a complete support of WSRP 1.0 standard interfaces and offers both consumer and producer services. WSRP support is provided by the portal-wsrp.sar service archive, included in the main jboss-portal.sar service archive. JBoss Portal does not however, by default, expose local portlets for consumption by remote WSRP consumers.

Abbreviations:

SaaS   -   Software as a Service

JSR   -   Java Specification Request

UI   -   User Interface

SOA   -   Service Oriented Architecture

LDAP   -   Lightweight Directory Access Protocol

JEMS   –   Jboss Enterprise Middleware Suite

WSRP   -   Web Services for Remote Portlet

HTTP   -   Hyper Text Transfer Protocol