|  |  |
| --- | --- |
|  | |
| **Verio MarketPlace**  **Application Construction & Alpha Release**  **Portal Framework**  **Design** | |
| **Version:** | 1.0 |
| **Release Date:** | 31stMay 2012 |
| **Author:** | InterraIT |
| **Approver:** | Verio Inc. |
| User of the document is responsible to ensure use of current version of the document.  This document is for restricted circulation. No part of this document shall be reproduced, stored in retrieval system or transmitted in any form or by any means – recording, photocopying, electronic and mechanical without prior permission of InterraIT. | |
| Interra Information Technologies, Inc.  25 Metro Drive, Suite 550, San Jose, CA - 95110  Ph.: +1 408 451 1700 Fax: +1 408 441 7495  Web site: <http://www.InterraIT.com> | |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Release Date** | **Author** | **Nature of Change** |
| 1.0 | 31st May2012 | InterraIT | First Resdflease |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1 Introduction 6](#_Toc326167320)

[1.1 Overview 6](#_Toc326167321)

[1.2 Requirements & Scope 6](#_Toc326167322)

*[1.3](#_Toc326167328)* [SoftwareRequirement 6](#_Toc326167328)

[1.4 Definitions, Acronyms & Abbreviations 6](#_Toc326167329)

[1.5 Assumptions, Constraints & Dependencies 7](#_Toc326167330)

[1.5.1 Assumptions 7](#_Toc326167331)

[1.5.2 Constraints/Limitation 7](#_Toc326167332)

[2 High Level Design 8](#_Toc326167333)

[Component Design 8](#_Toc326167334)

*[2.1](#_Toc326167335)* [8](#_Toc326167335)

[2.2 Request flow - sample 8](#_Toc326167336)

[2.3 Portal MV(MVP)C Design 9](#_Toc326167337)

[2.3.1 Framework View (MVP Design) 11](#_Toc326167338)

[2.3.2 Framework Controller 14](#_Toc326167339)

[2.3.3 Framework Model 14](#_Toc326167340)

[3 Low Level Design 15](#_Toc326167341)

[3.1 Portal framework Entity Relationship diagram 15](#_Toc326167342)

[3.2 DB Entities and DAO Layer 16](#_Toc326167408)

[3.3 DB Views 16](#_Toc326167409)

[3.3.1 Service Layer 17](#_Toc326167415)

[3.3.2 Controller Layer 17](#_Toc326167416)

[3.3.3 View Layer/GXT 17](#_Toc326167418)

[3.3.4 Data caching layer 17](#_Toc326167419)

[3.4 Framework Package Structure 18](#_Toc326167420)

[3.4.1 Package component summary 18](#_Toc326167421)

[3.4.2 Package component description 19](#_Toc326167422)

[3.4.3 Configuration files structure 19](#_Toc326167423)

[3.5 Authentication and Authorization 19](#_Toc326167455)

[User Management 21](#_Toc326167460)

[3.5.1 21](#_Toc326167462)

[3.5.2 Role Management 21](#_Toc326167464)

[3.5.3 Authentication 21](#_Toc326167468)

[3.5.4 Access control 21](#_Toc326167469)

[3.5.5 Authorization 23](#_Toc326167470)

[3.6 Branding Design 23](#_Toc326167472)

[3.6.1 Profiles 23](#_Toc326167473)

[3.6.2 Styling and Image Resources 23](#_Toc326167474)

[3.6.3 Customized Controls, Labels and Tooltips 24](#_Toc326167477)

[3.7 Globalization Design Implementation 25](#_Toc326167501)

[3.7.1 Locale 25](#_Toc326167502)

[3.7.2 Date Formats 25](#_Toc326167503)

[3.7.3 Currency Formats 25](#_Toc326167504)

[3.8 User Interface Components 26](#_Toc326167506)

[3.8.1 Form / Containers/Control Implementation 26](#_Toc326167507)

[3.8.2 Menus Implementation 29](#_Toc326167521)

[3.8.3 Messages Implementation 32](#_Toc326167522)

[3.8.4 GXT Controls Implementation 32](#_Toc326167532)

[2.2.2.1 Class Diagram 32](#_Toc326167534)

[2.2.3.1 Sequence Diagram 32](#_Toc326167535)

[3.9 Session management 33](#_Toc326167536)

[3.10 Caching Design 34](#_Toc326167537)

[3.11 Integration with HAT (Help Authoring Tool) 34](#_Toc326167538)

[3.11.1 Database Design 34](#_Toc326167539)

[3.11.2 View Integration with HAT 34](#_Toc326167540)

[2.2.4.1 Context Sensitive Help 34](#_Toc326167541)

[3.12 Logging 35](#_Toc326167542)

[3.12.1 Local Logging 35](#_Toc326167543)

[2.2.5.1 AOP Implementation 35](#_Toc326167544)

[2.2.6.1 Types of Logging 35](#_Toc326167545)

[3.12.2 Remote Logging 35](#_Toc326167546)

[2.2.7.1 Service Invocation 35](#_Toc326167547)

[2.2.8.1 Remote Logging Sequence Diagram 35](#_Toc326167548)

[3.13 Exception Management 36](#_Toc326167549)

[3.13.1 Exception Hierarchy 36](#_Toc326167550)

[4 Portal Application Design Reference 37](#_Toc326167554)

[5 OSS-B (SOA) Integration Design 38](#_Toc326167555)

**List of Tables**

*[Table 1: Abbreviations](#_Toc323190948)* [3](#_Toc323190948)

*[Table 2: Framework Package Structure](#_Toc323190949)* [3](#_Toc323190949)

**List of Figures**

*[Figure 1: High Level Component View](#_Toc325517402)* [16](#_Toc325517402)

[Figure 2: Access Control depiction 41](#_Toc325517403)

*[Figure 3: Common vs. Custom Theme Content](#_Toc325517404)* [46](#_Toc325517404)

*[Figure 7: Portal Design Reference](#_Toc325517405)* [67](#_Toc325517405)

*[Figure 8: Portal Integration Architecture](#_Toc325517406)* [68](#_Toc325517406)

# Introduction

## Overview

OSSB Portal framework will provide a generic implementation of GXT controls and widgets that would be customised for common properties like maxLength, height, spacing etc. In addition to that, there will be branding and globalization done to have labels, colour themes, logos/images, fonts based on profile and locale of the logged in user. There will be provision for role based access on various controlsand menus.It would work as a thin layerof GXT control robust enough to support any portal application that would consume its API’s.

In addition, portal framework will have message templates and tokens defined for both Email/SMS communication services. There will be branding and globalization done to have tokens,templates customized based on profile and locale.

## Requirements & Scope

Following are the features covered in this framework:-

* **Branding/Profile Management**

Branding would involvehaving the colour theme, logos/images, fonts etc. displayed based upon profile of logged in user. Also user will get to see the controls on the screen based upon the profile.

* **Globalization**

As part of globalization the framework will support and labels of various controls on screen, tooltip etc. in language based upon individual’s locale. But from initial implementation perspective we’ll be limiting our scope to English language only.

* **User Role Access Control**

User will get to see control on screen based upon his/her roles as part of User Role access control.

## SoftwareRequirement

Here is the software stack we plan to use

* GXT 3.0
* GWT 2.4
* Java 6+
* Spring 3.x ( MVC, Web Services)
* Hibernate Core 3.5.x
* Oracle 11g
* Oracle JDBC driver
* Jackson library 1.9.1 ( For JSON)
* JAXB 2.x ( for parsing & creating xml data structure)
* JBoss 7.x Application Server
* Apache Jackrabbit 2.2.9 (WebDAV Server)
* Apache CXF/JBoss Rest Easy (for REST implementation)
* EhCache2.4.6 (For business objects Caching)

## Definitions, Acronyms & Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| GWT | Google web toolkit |
| GXT | Ext-GWT (extension of GWT) |
| ORM | Object Relation Mapping |
| GUI | Graphic user interface |
| Subsystem | Logical parts of the software system |
| UI | User interface |
| SOA | Service oriented architecture |
| WebDAV | Web-based Distributed Authoring and Versioning |
| XML | Extensible mark-up language |
| JSON | Java script Object notation |
| GUI | Graphical User interface |
| CSS | Cascading style sheets |
| OEM | Original equipment manufacturer |
| MVP | Model view presenter |
| MVC | Model view controller |
| REST | Representational state transfer |
| AJAX | Asynchronous JavaScript and xml |
| HTTP | Hypertext transfer protocol |
| DAO | Data access object |
| JPA | Java persistence API |
| JAXRS | Java API for RESTful Web Services |
| JAXB | Java API for XML Binding |
| HTML | Hypertext mark-up language |
| JDBC | Java database connectivity |
| SSO | Single sign-on |
| MIME | Multipurpose Internet Mail Extensions |

*Table 1: Abbreviations*

## Assumptions, Constraints & Dependencies

### Assumptions

Portal framework design is based on following assumptions.

* <*Assumption 1*>
* <*Assumption 2*>

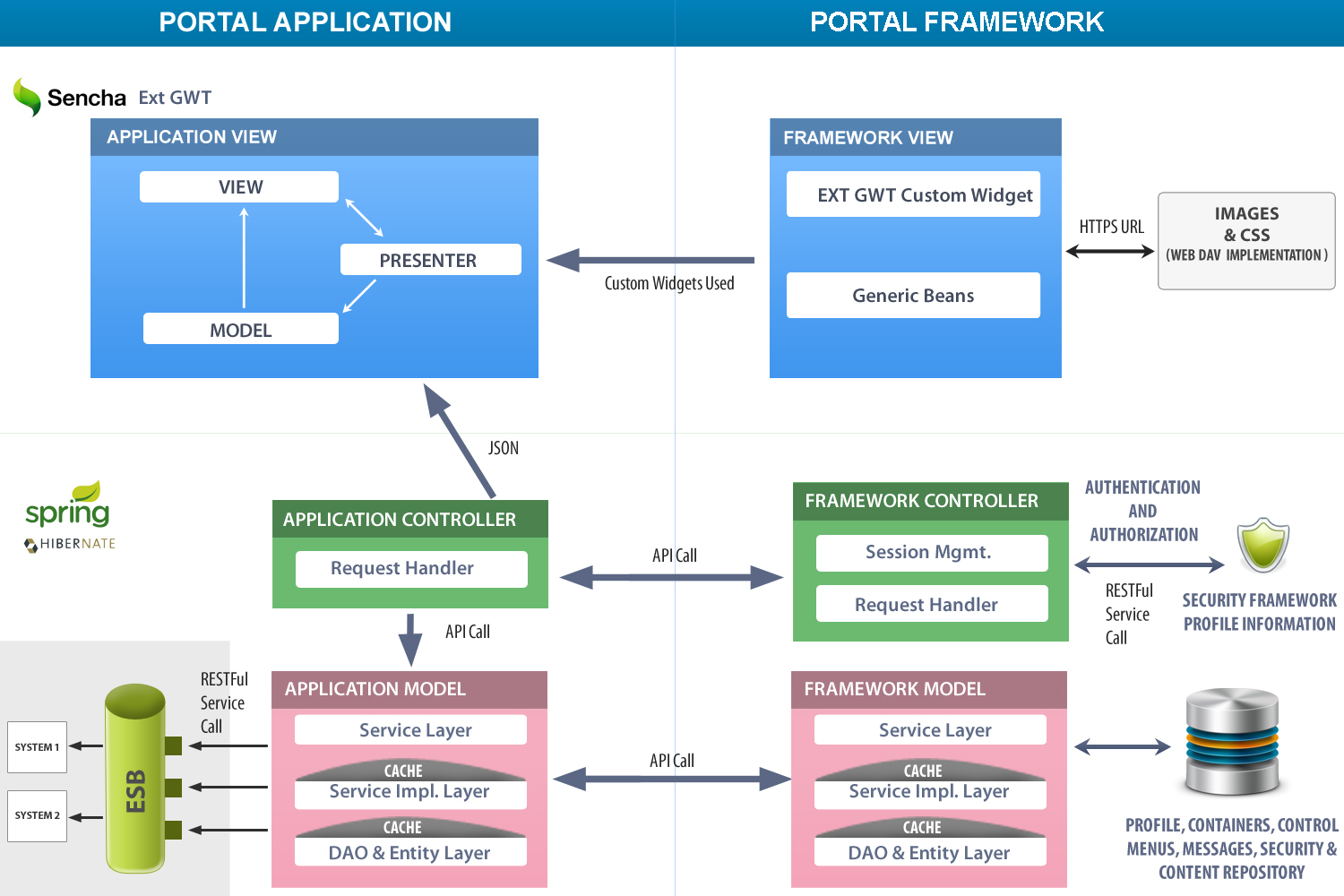
### Constraints/Limitation

Portal framework design has the following constraints:

* + Any GUI related changes like creating a new layout, additions/removal of containers/controls resulting in GUI change are not part of framework and will beunder the purview of actual portal development and might require development based upon the kind of change.
  + Any customization done in labels, controls in term of content, localization, access control resultingin GUI changes willbe under the purview of actual portal development and might require development based upon the kind of change.

# High Level Design

## Component Design



*Figure 1: High Level Component View*

## Request flow - sample

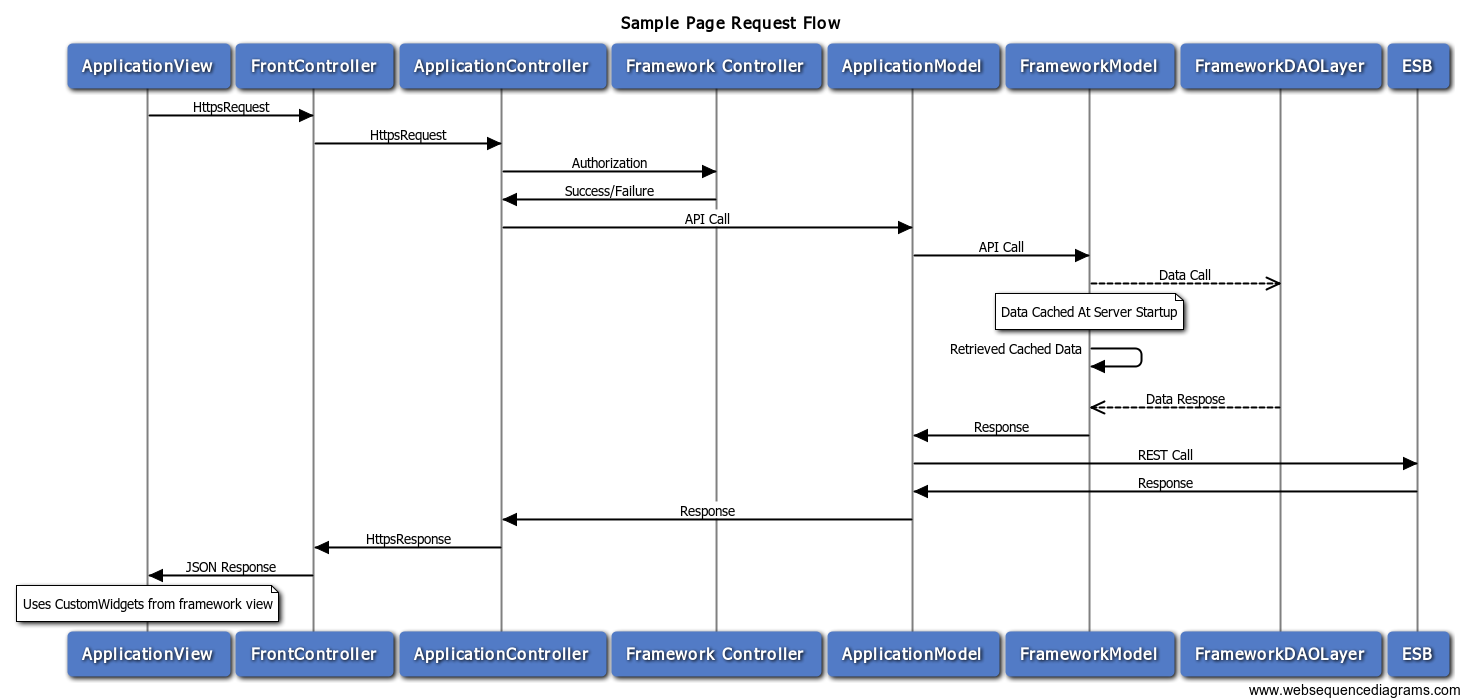


Figure 2: Sample request response flow

## Portal MV(MVP)C Design

The overall OSS-B portal architecture is based on the MVC architectural pattern, with the view component being further decomposed into multiple components based on MVP (model-view-presenter) design pattern.

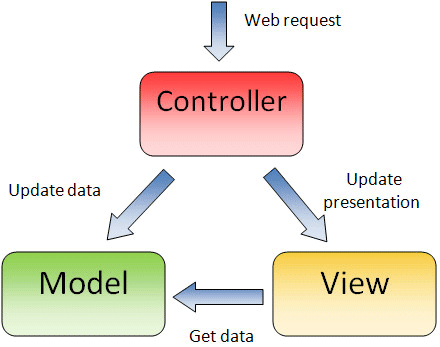
Out of this the portal framework will encompass below components:

* **Custom Widgets**

This will encompass a set of custom widgets having properties like length, height etc. customized and available to individual portals in form of jar file. The custom widgets will be used by view layer to render controls on screens of portals.

* **Framework Server**

The framework server will be responsible for caching all the framework metadata (controls, menus, globalization data and access control) and provide this data to the portals based upon request parameters like user, profile, role etc. The framework server also will be available as jar/api to individual portals.



*Figure 3: Core MVC Architecture*

### Framework View (MVP Design)

The MVC View is implemented using model-view-presenter pattern.

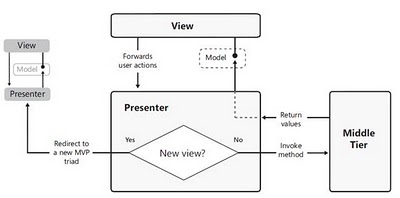


Figure 4: Core MVP Architecture

##### View

This view is typically an instance of a class that implements the model and holds reference to an instance of the presenter. Between views and presenters, there is a one-to-one cardinality (though the number of presenter classes will be minimized by creating a hierarchy and reusing the code).

The initial web view of the application is being segregated out of the Ext-GWT implementation and only the dynamic part of the application will be rendered through this View part. Initial page will be rendered based upon the URI being entered which is resolved by the Apache web server to fetch the correct set of Html/Css files required for the logged in user from HttpDav Server.

The view part of the Framework is composed of:-

**Ext-GWT Widgets:**Widget refers to an onscreen component (created in html on the web page) that will provide a user control to either depict/input some sort of information from the user. All the widgets solely will be used from the Ext-GWT library only.

**Custom Widgets:** Based on re-usability needs, a set of custom widgets need to be created (on top of the default Ext-GWT widgets) to simplify the development work.

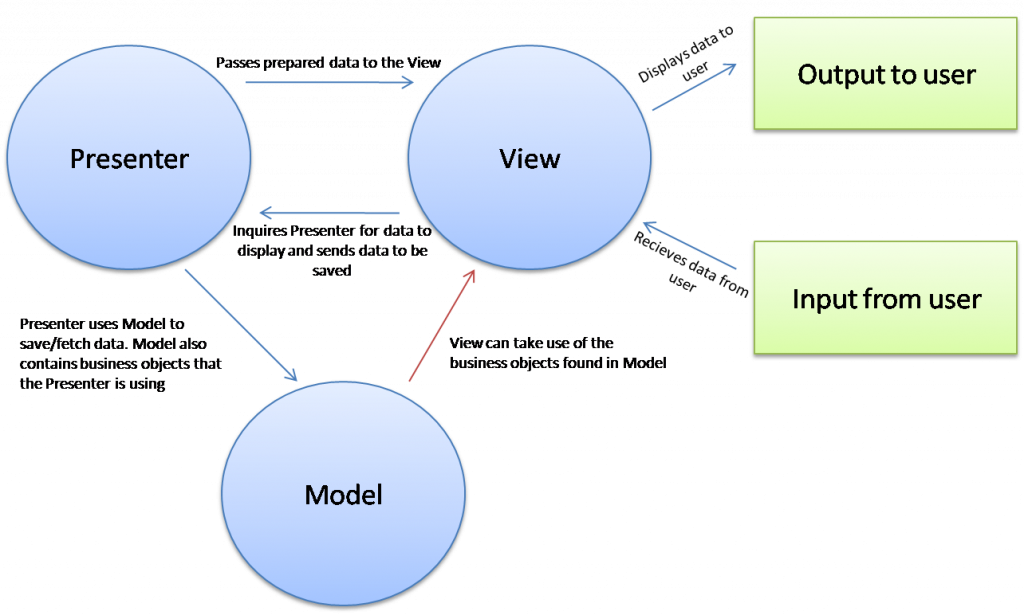
##### Presenter

The presenter uses view object to grab input values and prepare a call to the middle tier through a list of sub-controllers. Each sub-controller performs a specific tasks related to the request. It can include performing some business logic over the components or invoking a server side function through http based interfaces. After the response is received, the presenter passes data back to the view. The Presenter is implemented by standard set of classes provided by Ext-GWT (or will be extended to provide custom functionality depending upon the business use case).During initialization, the presenter is fed with the input view, with the events acting as the linkages between them. Once an event is fired from the UI, the presenter receives the event with the view as input and forwards the request to appropriate sub-controller, which modifies the model to reflect the changes back in the UI.

##### Model

The model is the representationof data being worked on in the view. In Ext-GWT model are being represented by a set of classes whose values are attached to a Widget at the time of initialization. Thus, the model is typically a domain level object. It holds application data and provides methods to consistently access it.

A model does not provide an association with the User Interface views. This allows the model to be reused amongst many portal windows having different user interface requirements.   
The following diagram represents how the model interacts with the other MVP components:



*Figure 4: Core MVP Architecture Data Flow*

### Framework Controller

As the OSS-B portal Framework architecture is based on Spring MVC, the Dispatcher Servlet provided by Spring will acts as the Framework controller (thus implementing the front controller pattern). All the requests originating from the UI will be directly handled by this controller.

Controller workflow:

* The Client requests for a Resource in the Web Application as an http request.
* The request is analysed by the Spring Security Filters to determine all the aspects of the security are being addressed in the HTTP Request & forwards to the Spring front Controller.
* The Spring Front Controller, which is implemented as a Servlet, will intercept the Request and then will try to find out the appropriate Handler Mappings configured through the annotations.
* The Handle Mappings is used to map a request from the Client to its Controller object by browsing over the various Controllers defined in the controller’s package.
* With the help of Handler Adapters, the Dispatcher Servlet will dispatch the Request to the Controller.
* The Controller processes the Client Request and returns the Model in the format as desired by the UI back to the Front Controller.

The data returned will be looked upon by security filter (if any case, the modification needs to be either to the data or to the headers passed in the response).

### Framework Model

In Spring MVC context, a model represents the data that will be passed to and from an operation (defined in a web controller) on a resource invoked by a unique Restful URI associated with the resource.

All the models will be represented by a set of beans arranged in a proper set of packages; with the individual model has no information about the type of resources with which it can be linked with. This helps in clearly separately the line of responsibility for each of the component and allow the model classes to be reused anywhere in the application, e.g. the Product JPA Entity bean can be reused as the model bean, to return back the details of a specific product.

All the models will be populated by various Spring based services written on top of the Business services & External services. Using Spring, the same set of services can be exposed as a web service, if required directly.

All the data stored in various OSS-B systems will be exposed via wrappers. All the wrappers are themselves encapsulated in a separate package “**External Services**”. All the data stored in the OSS-B Portal database will be made accessible by the Spring Dao layer (implemented with **HibernateDaoSupport**).

# Low Level Design

## Portal framework Entity Relationship diagram

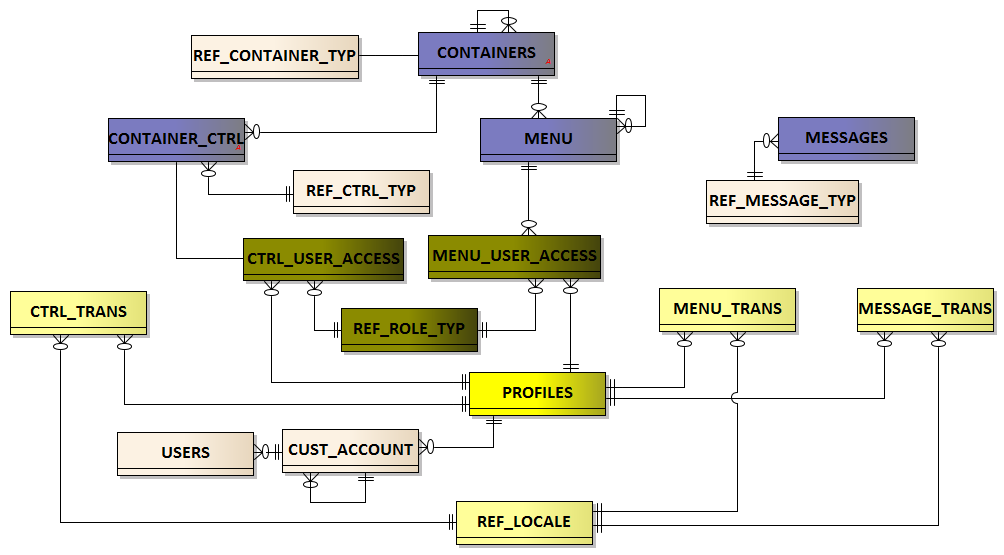


Figure 1: Portal framework- Entity relationship diagram



## DB Entities and DAO Layer

Here are the database entities proposedas part of Portal framework.

* **Containers**

Containers are placeholders for customizable controlsthat would make building blocks for the portals to be developed using the portal framework. There is a provision for parent/child relationship to maintain hierarchy.

* **Controls**

Controls are UI components with ability to specify custom properties like text length limit,

height etc. with ability to have locale based captions.

* **Trans Tables**

Trans tables are responsible for holding locale based data for Control, Menus and messages

for globalization and profile based customization.

* **UserRoleAccess**

UserRoleAccessprovides provision for Brandingof various controls, menus on the basis of

user role and profile.

* **Menu**

To hold data for various Menus to be used by different portals.

* **Message**

This entity suggests various types of messages to be associated with the control to make it

more INFORMATIVE.

* **Profile**

Profile will be used for globalization and other customization and access control.

* **User**

To hold user attributes like profile, account, role etc. This again will be useful for

authorization, customizations and globalization.

* **Customer Account**

To hold data for various accounts and corresponding profiles, and to maintain account

Parentchild relationship. This is useful for brandingpurpose.

* **Templates**

This entity suggests various types of message templates available for outbound Email/SMS communication services.

## DB Views

Following views have been created and would act as entity objects to support DAO implementations with the hibernate DAO Support:-

* **VW\_CONTAINER\_CTRL**

Provides container, control relationship related data.

* **VW\_CTRL\_ATTR**

Provides data for all translate-able (profile, locale based) as well as non-translatable attributes related to controls.

* **VW\_CTRL\_ACCESS**

Provided access data for various roles and profiles for all the controls.

* **VW\_MENU\_DATA**

Provides container, menu relationship related data along with parent child relationship among menus.

* **VW\_MENU\_ATTR**

Provides data for all translate-able (profile, locale based) as well as non-translatable attributes related to menus.

* **VW\_MENU\_ROLE\_DATA**

Provided access data for various roles and profiles for all the menu items.

* **VW\_MESSAGE\_PROP**

Provides data for all translate-able (profile, locale based) as well as non-translatable attributes related to messages.

* **VW\_MESSAGE\_TEMPLATES**

Provides data for message templates with their details.

* **VW\_MESSAGE\_TEMPLATE\_ATTR**

Provides data for all profile, locale based attributes related to message templates.

* **VW\_MESSAGE\_TEMPLATE\_PROFILE\_DATA**

Provides data for all profile based templates.

## Service Layer

This layer contains a set of APIs that are to be used by various portals to access data setup around branding, globalization and user access control. The data is cached by cache services.

* **ContainerControlService**

Providescontainer control related details to various portals.

* **MenuItemsService**

Providesmenus related details to various portals.

* **MessageService**

Providesmessage details.

* **MetaDataLoadService**

Initializes different components to different portals.

* **UserLoginService**

Provides logged in user’s details, user controls information, user role details and validates it.

* **MessageTemplateService**

Provides message templates related details for outbound communication services.

## Controller Layer

Controller later is not applicable to portal framework server, but individual portals will have a controller layer that will make sure of framework details on access branding and customization related data.

## View Layer/GXT

The layer contains set of custom widgets to be used by various portals.

## Data caching layer

This layer sets the data in cache and is to be loaded and server start up with ability to refresh it on demand:-

* **CntrCtrlCacheService**

Caches container control related information.

* **MenuCacheService**

Cached menu details data.

* **MessageCacheService**

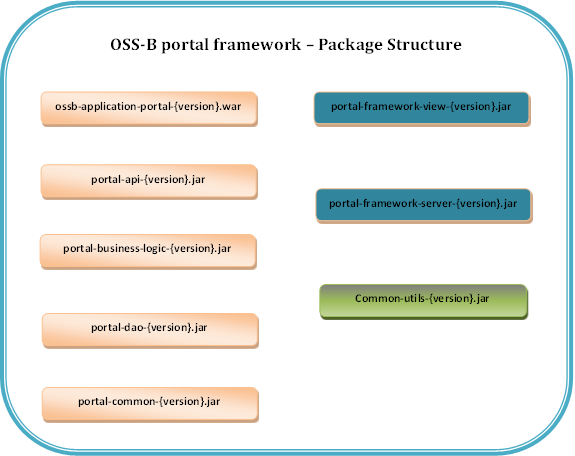
To cache various messages to be used in portals.

* **MessageTemplateCacheService**

Caches various message templates details with data.

## Framework Package Structure

### Package component summary











*Figure5:*Portal Framework Client Package Structure

### Package component description

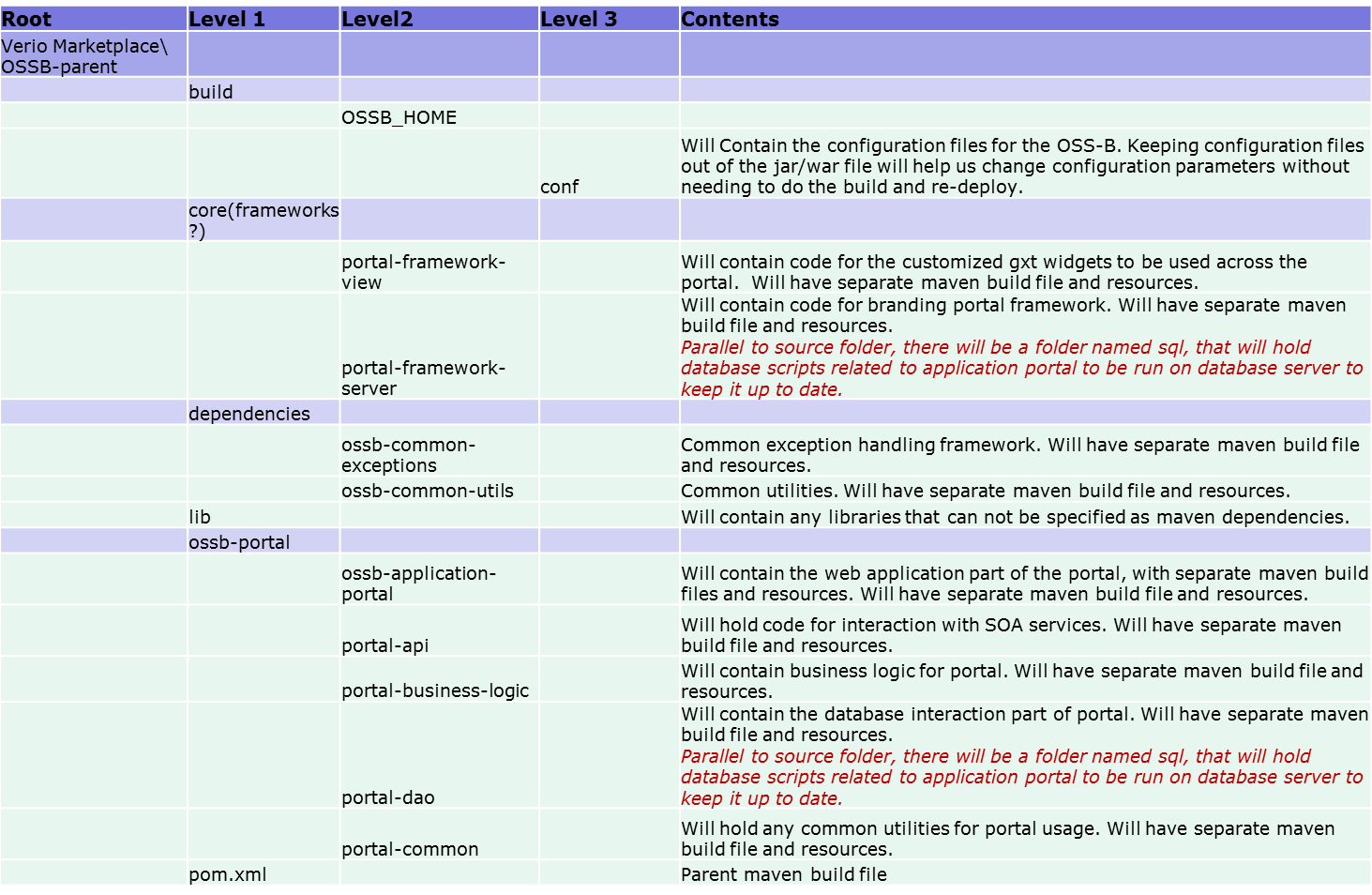


Figure 1: Description of package components

### Configuration files structure

##### Dynamic configuration

The configuration files having propertiesthat are dynamic in nature and are likely to change outside release are to be placed outside build archive (jar/war file). Example of these properties include interfacing end points to any component, e.g. Webdav server URL, database connection properties, jmx server URL and other properties. Dynamic refreshfacility will be available to be able to change these properties and have into effect without a need to restart the server. Folder structure for configuration will be as follows.

**The Configuration Tree**

/etc  
 |   
 +--- other\_app  
 | |  
 | |  
 |  
 |  
 +-- marketplace  
 | |  
 | +----- party  
 | | |  
 | | +------ mvc  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +---- services  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +--- messaging  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +---- logging  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | | +------- logback.xml  
 | | |  
 | | +---- security  
 | | +------- dev.properties  
 | | +------- qa.properties  
 | | +------- staging.properties  
 | | +------- prod.properties  
 | |  
 | +---- product  
 | | |  
 | | +------ mvc  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +---- services  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +--- messaging  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +---- logging  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | | +------- logback.xml  
 | | |  
 | | +---- security  
 | | +------- dev.properties  
 | | +------- qa.properties  
 | | +------- staging.properties  
 | | +------- prod.properties  
 | |  
 | +----- order  
 | | |  
 | | +------ mvc  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +---- services  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +--- messaging  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +---- logging  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | | +------- logback.xml  
 | | |  
 | | +---- security  
 | | +------- dev.properties  
 | | +------- qa.properties  
 | | +------- staging.properties  
 | | +------- prod.properties  
 | |  
 | +--- promotion  
 | | |  
 | | +------ mvc  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +---- services  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +--- messaging  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | |  
 | | +---- logging  
 | | | +------- dev.properties  
 | | | +------- qa.properties  
 | | | +------- staging.properties  
 | | | +------- prod.properties  
 | | | +------- logback.xml  
 | | |  
 | | +---- security  
 | | +------- dev.properties  
 | | +------- qa.properties  
 | | +------- staging.properties  
 | | +------- prod.properties  
 | |  
 | +- other component  
 |  
 +--- other\_app  
 |  
 |

##### Not so dynamic configuration

The configuration parameters that are unlikely to change outside a release cycle will be placed along with code and will be part of the build archive (jar/war file).



## Authentication and Authorization

Authorization and Authorization has been handled in two ways:-

* **Identity Server**

This entity would validate the user who has logged on based upon user/password provided, and return roles associated with the user.

* **Framework level**

Existence of a user in framework database and corresponding role and profile would be used to decide access to various sections of portals.

### User Management

Stored in table.portal framework database, users are managed in users table.

Stored in identity server, user details are provided by the WSO2 identity server.

### Role Management







Every user is associated with one or more roles and access id granted based upon profile, role combination.Roles related data is managed in identity server.

### Authentication

Every user request will be redirected to identity server which will authenticate theuser and return the roles the user has been associated with. The identity server will optionally (to be established) return theprofile and account of the user.

### Access control

Access control is driven by profile, role combination of any given user. Various controls and menus in different screens will be permissioned based upon profile, role combination. Every child profile will inherit access from all its parents for the same role. If there is no entry for an element (control or menu) in access control tables, it would mean access not being there. Here is an access control illustration.

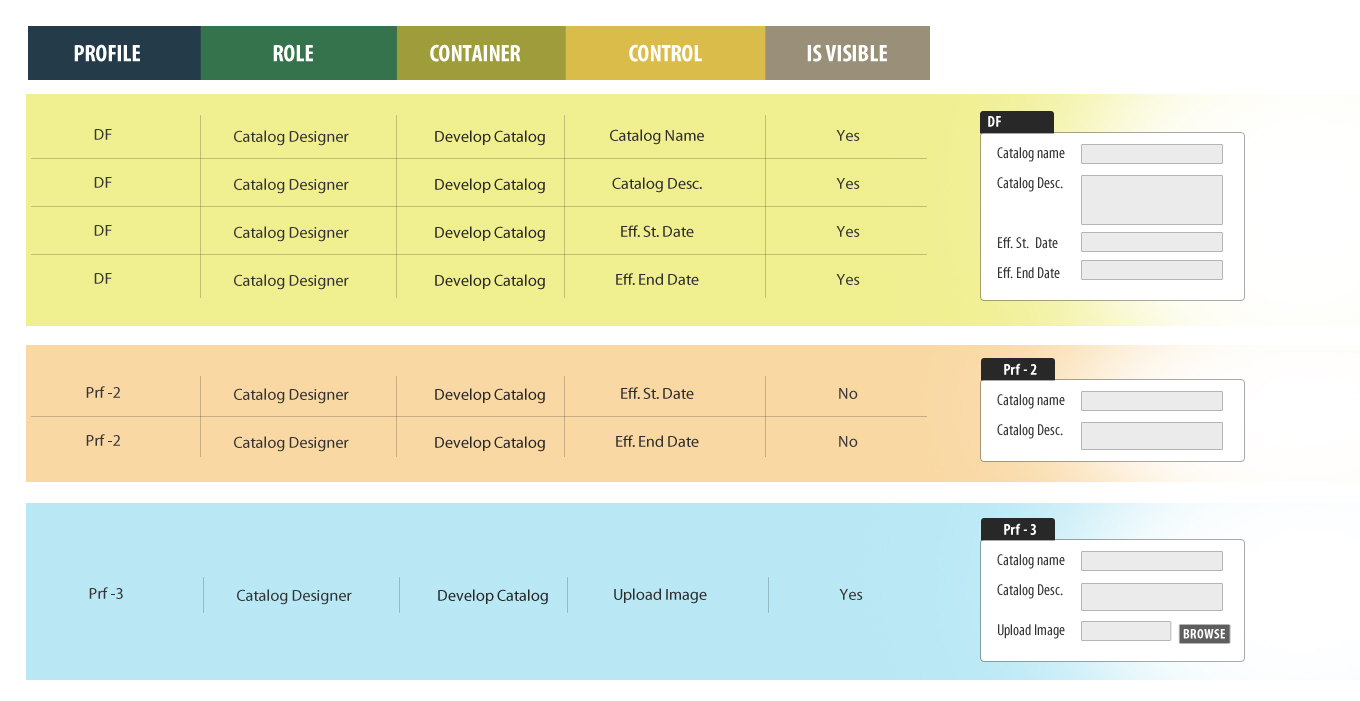


Figure 2: Access Control depiction

Here a user with “Catalog Designer” role, with DF (“default”) profile will have access to all four controls as listed in yellow highlighted table.

Similarly a “Catalog Designer” with Prf-2 as profile will get to see just two control as the other two have been removed from access control.

And a “Catalog Designer” with Prf-3 profile will have access to one additional control.

### Authorization

Once authenticated, the user will get to see relevant sections in the requested module based upon the roles and profile returned by identity server for theuser as described above.

## Branding Design

### Profiles

Profiles form the basis for the branding of elementary widgets like buttons, menus.

It will also create the platform for the styling of the widgets using custom images and css(colors, fonts) files.Along with customisation, profiles will facilitate the access control on the menus and controls as described in access control section.









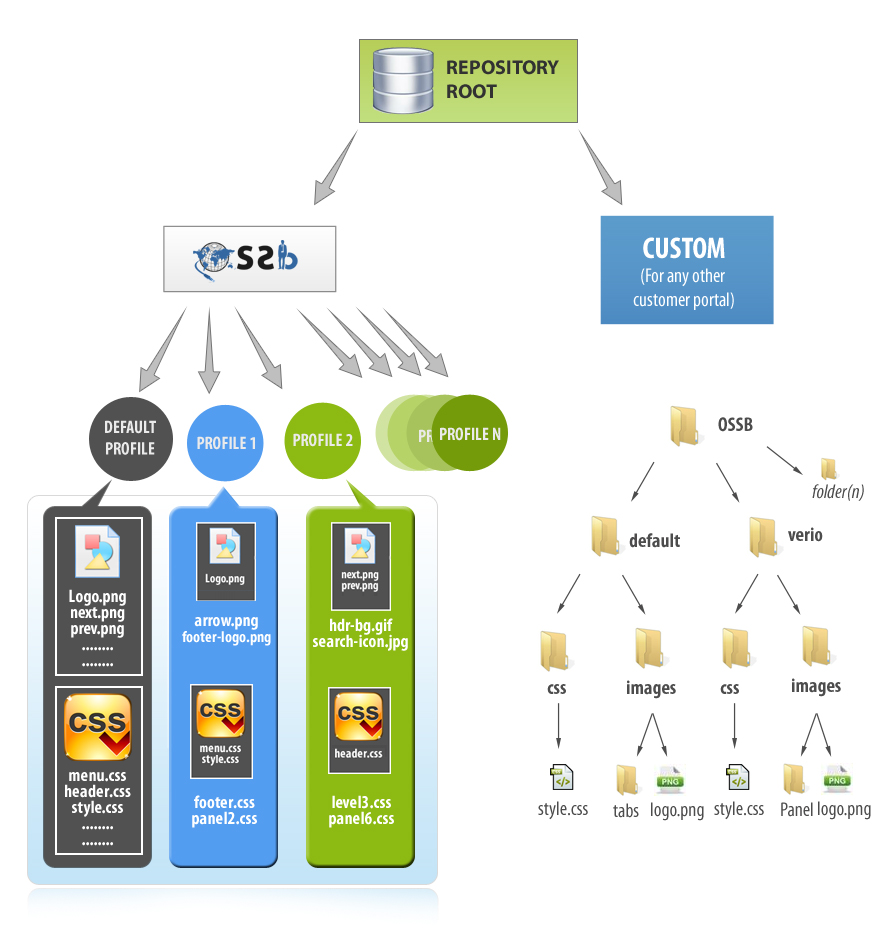


### Styling and Image Resources

Profilesplay an integral role while deciding the User interface of the OSS-B portal to the end User. The portal framework provides a structured approach to manage the various themes and made it easy for the end user (OEM/Reseller) to use any of the available themes. A Single theme available in the system comprises of

* Multiple Cascading Style Sheets (CSS) present in “**CSS**” folder within the theme.

Images



*Figure 3: Common vs. Custom Theme Content*

At any point of time when the themes need to be displayed, the request will be checked to see through headers and URL parameters, if a profile has been customized for an individual OEM/Reseller then instead of the defaultCSS the new formatted CSS will be returned, so that the new theme will get displayed.

The above diagram depicts how each profile will have its own set of static content.

##### Content Repository

Portal UI in addition to serving the dynamic contents by interacting with various systems also provides a panel to manage all the static contents of the application for the user. The static contents can be either the images, cascading style sheet etc. All the static contents are being arranged in such a way that each profile will have a folder under common folder.

OSS-B Portal application will make use of a Clustered environment of Content Repository Server (open sourced Apache jackrabbit software, an implementation of Http WebDav protocol).

Portal application will make use of Http WebDav client library in order to access/modify the resources present in the system.

All the requests for any of the static resources will be diverted to the clustered group of Jackrabbit servers by the front facing web servers. This will help in reducing the load on the application servers as well as providing easy management of the resources via the Jackrabbit GUI console or the portal admin panel.

### Customized Controls, Labels and Tooltips

As part of branding, each screen in the portal will be defined and container as controls where every page will become a container, containing further containers, which willeventually containcontrols.

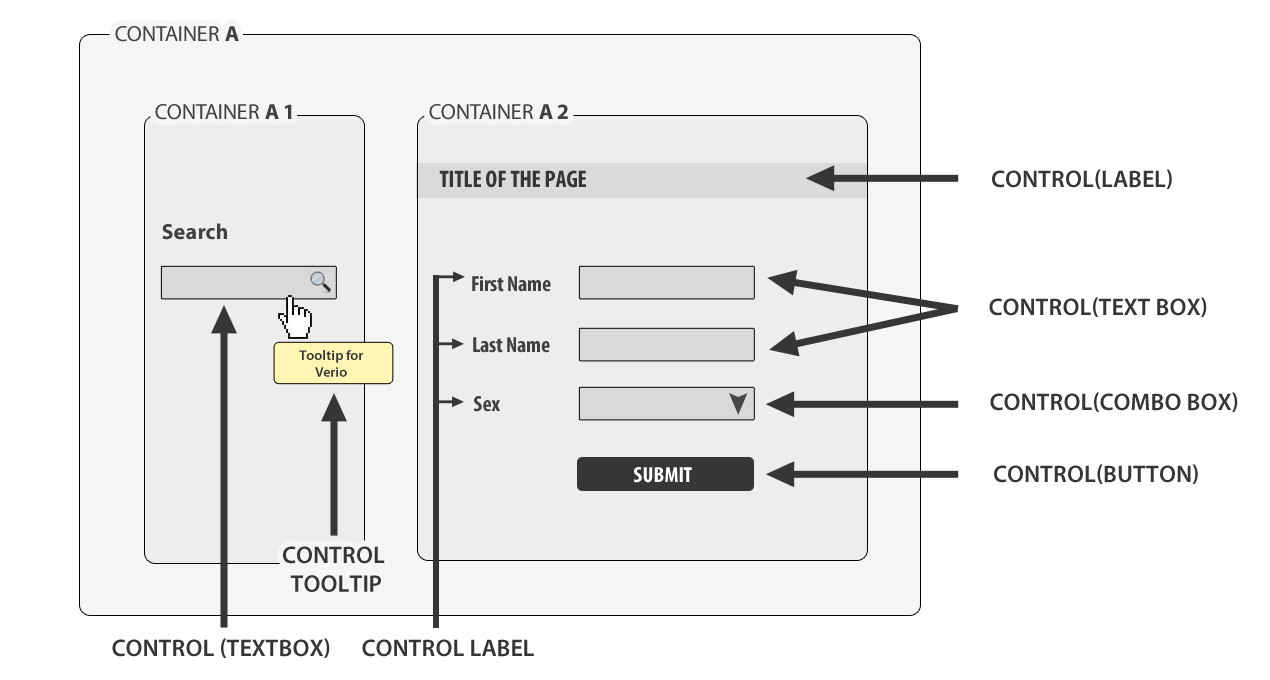


Figure 5: Depiction of container and controls

These containers and controls will be registered in framework database and will be customized based upon profiles and locale.Information like maxLength,is\_madatory will be kept in database as well which will aid in validation checks. Users will be given access to controls based upon profile and role.

##### Database Design

Following table structure would support customised control labels and tooltips

CTRL\_TRANS

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN\_NAME** | **TRANS\_VALUE** | **PROFILE\_NAME** | **LOCALE\_NME** |
| CTRL\_LABEL\_ID | Username | VERIO | en\_us |
| CTRL\_LABEL\_ID | User Name | YellowBook | en\_us |

As an example we can see the label name being customized based upon profile.

##### Object Structure

Please refer section 3.8..1.2.



## Globalization Design Implementation

### Locale

Locale indicates multilingual content that will appear on screen controls. It could be in form of labels, tooltip, help text etc.

##### Database Design

Globalization will be achieved by configuring multilingual content in database based upon locale with a profile.



CTRL\_TRANS

|  |  |  |  |
| --- | --- | --- | --- |
| **COLUMN\_NAME** | **TRANS\_VALUE** | **PROFILE\_NAME** | **LOCALE\_NAME** |
| CTRL\_LABEL\_ID | Username | VERIO | en-US |
| CTRL\_LABEL\_ID | **ユーザー名** | VERIO | Ja-JP |



### Date Formats

<*To be elaborated in Sprint 3>*

### Currency Formats

<*To be elaborated in Sprint 3>*

## User Interface Components

### Form / Containers/Control Implementation



##### Database Design

Based upon the relationship defined among container controls, here is DB schema depicting relationship among variousentities around Forms/Containers/Controls. The tables given below are from understanding perspective and do not map to DB table as is. Similarly column names and data have been expanded to make it interpretable.

A profile is assigned to an account and if no profile is given for an account, same is inherited from parent account.

|  |  |  |
| --- | --- | --- |
| **Cust Account** | **Parent Account** | **Default Profile** |
| ACC500155 | Null | Verio |
| ACC1166 | ACC500155 | Null |

Each UI page is a container, which can further can containers to nth level. Containers are essentially placeholders for controls objects.

|  |  |  |
| --- | --- | --- |
| **Portal** | **Container Code** | **Parent Container** |
| Customer Management | CMCA | Null |
| Customer Management | CMCASSearchTBar | CMCA |
| Customer Management | CMCAACntctTBar | CMCA |
| Customer Management | CMCAACntctDtal | CMCA |
| Customer Management | CMCAACntctGrid | CMCA |

Controls are UI components for user interaction and are associated with containers.

|  |  |  |  |
| --- | --- | --- | --- |
| **Portal** | **Container Code** | **Control Code** | **Control Type** |
| CMS | CMCASSearchTBar | SAccount | ToolBoxButton |
| CMS | CMCAACntctTBar | CntctNEW | TextButton |
| CMS | CMCAACntctTBar | CntcSAVE | TextButton |
| CMS | CMCAACntctTBar | CntRESET | TextButton |
| CMS | CMCAACntctTBar | CnDELETE | TextButton |

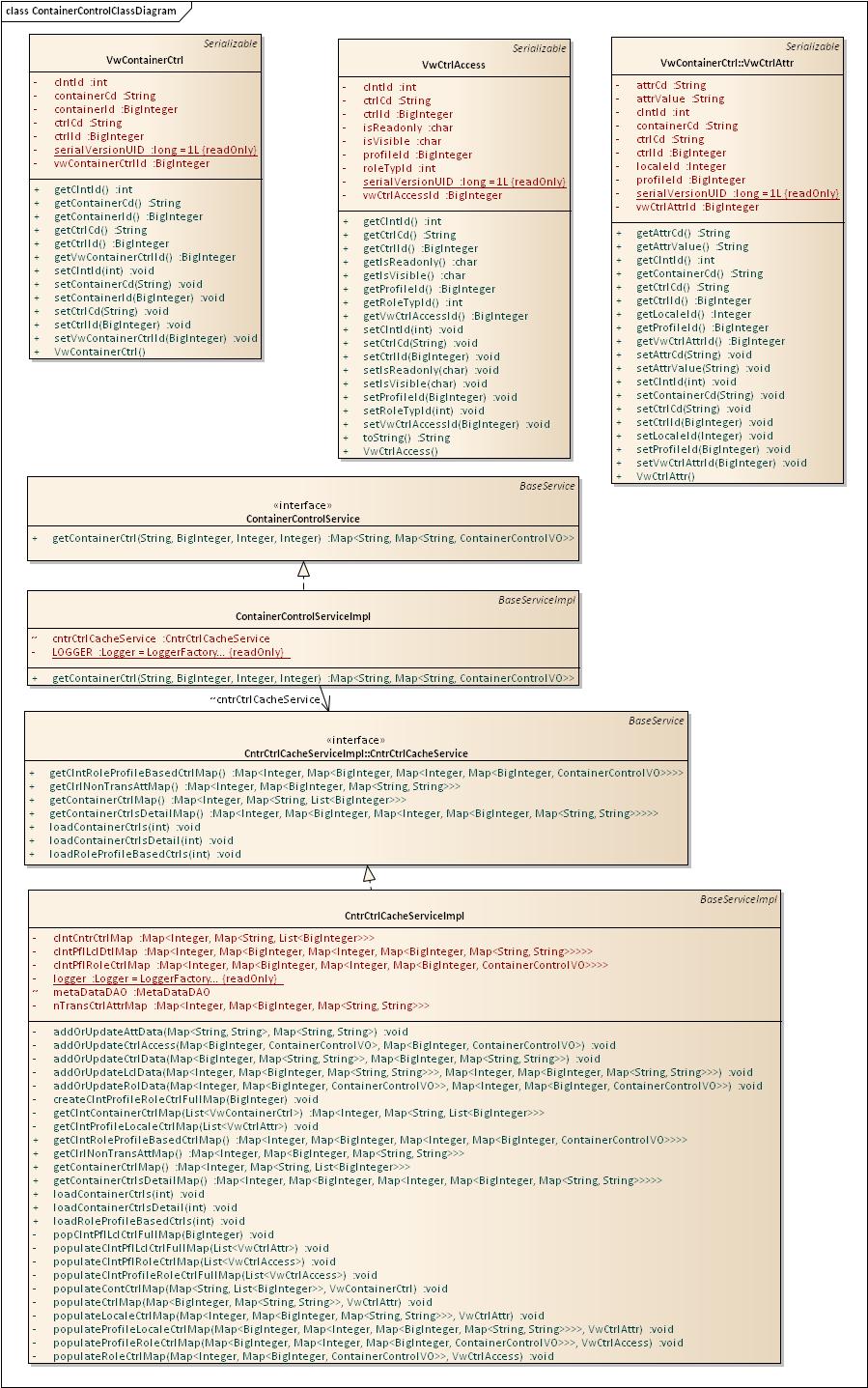
Controls are permissioned based upon profiles and roles.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Portal** | **Profile** | **Role** | **Control Code** | **IsVisible** | **IsReadOnly** |
| Cust. Mgmt. | Verio | Cust. Admin | CntctNEW | Y | N |
| Cust. Mgmt. | Verio | Cust. Admin | UserName | Y | N |
| Cust. Mgmt. | Verio | Cust. Admin | UserLName | Y | N |
| Cust. Mgmt. | Verio | Cust. Admin | UsrFName | Y | N |
| Cust. Mgmt. | Verio | Cust. Admin | CLstName | Y | N |
| Cust. Mgmt. | Verio | Cust. Admin | CDtoBrth | Y | N |
| Cust. Mgmt. | Verio | Cust. Admin | CPymntOP | Y | N |

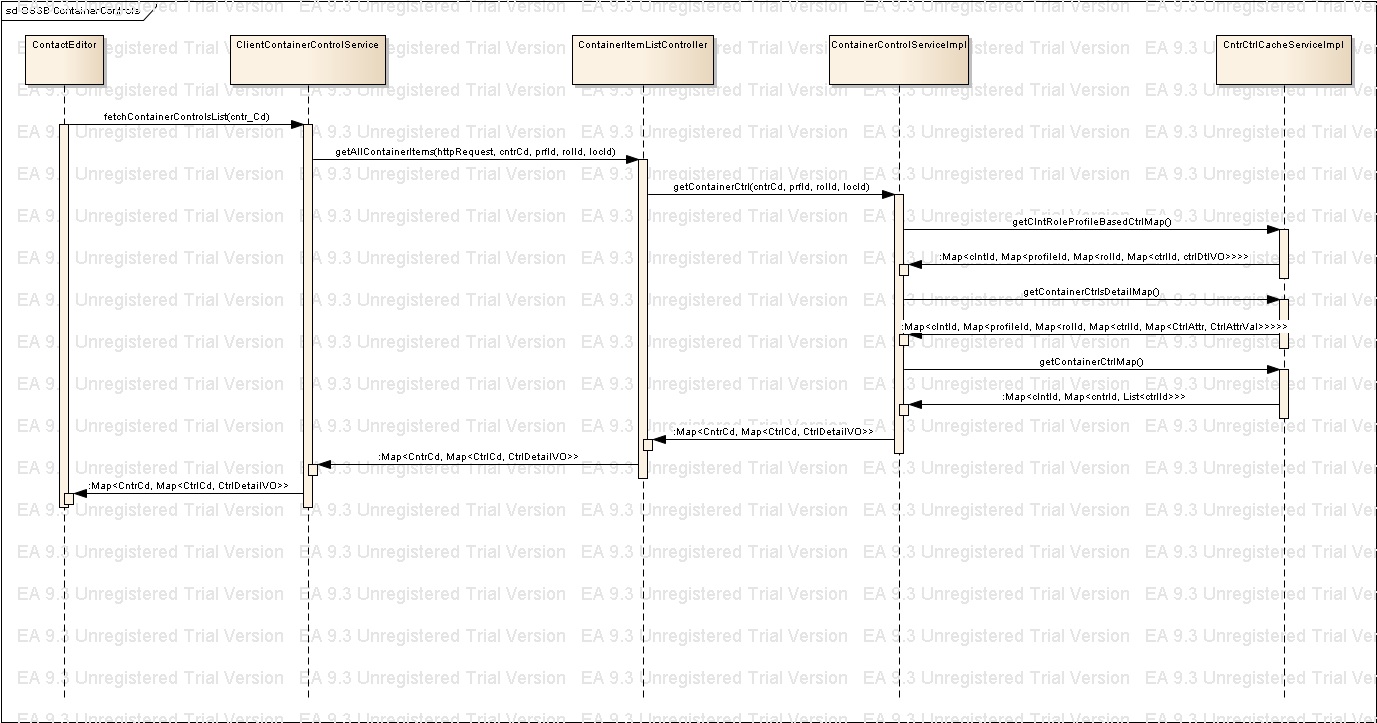
Various attributes like Label, Tooltip can be customized based upon profile and locale.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Portal** | **Container Code** | **Control Code** | **Profile** | **Locale** | **Attribute Code** | **Attribute Value** |
| Cust. Mgmt. | CMCASSearchTBar | SAccount | Verio | en\_US | CTRL\_LBL | Add |
| Cust. Mgmt. | CMCASSearchTBar | SAccount | Verio | en\_US | CTRL\_TIP | To add a record |
| Cust. Mgmt. | CMCASSearchTBar | SAccount | Verio | en\_US | CTRL\_LBL | Create New |
| Cust. Mgmt. | CMCASSearchTBar | SAccount | Verio | en\_US | CTRL\_TIP | To create a new record |

##### Class Diagram



##### Sequence Diagram







### Menus Implementation

Linking of menu to any page is similar to container, control association to a page. Like controls, menu is linked to a container associated with a given page and labels, structure, order, click action, click data for Menu items is configured in database. Menu labels are customizable based upon profile and are locale driven as well.

Below is a illustration of an example menu.

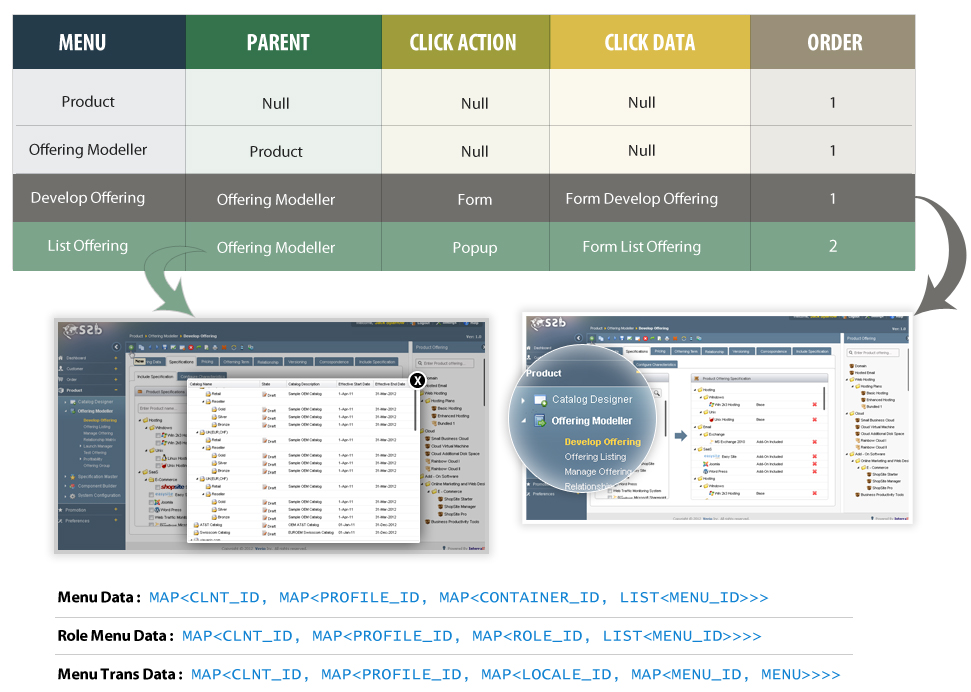


Figure 6: Menu data illustration

##### Database Design

Here is DB schema depicting relationship among various entities around Menus.

Menu, Submenu with ordering , click action and association with containers.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Portal** | **Container Code** | **Menu Item** | **Parent Menu Item** | **Item Order** | **On Click Action** | **On Click Data** |
| Cust. Mgmt. | CMSMainMenu | Party | Null | 1 | null | null |
| Cust. Mgmt. | CMSMainMenu | Party Relations | Party | 1 | MainWindow | CMNewPartyForm |
| Cust. Mgmt. | CMSMainMenu | Customer Account | Null | 2 | null | null |
| Cust. Mgmt. | CMSMainMenu | Account | Customer Account | 1 | MainWindow | CMNewAcccountForm |
| Cust. Mgmt. | CMSMainMenu | User | Customer Account | 2 | MainWindow | CMNewUserForm |
| Cust. Mgmt. | CMSMainMenu | Contacts | Customer Account | 3 | PopUP | CMNewContactForm |
| Cust. Mgmt. | CMSMainMenu | Payments Info | Customer Account | 4 | MainWindow | CMNewpayInfoForm |



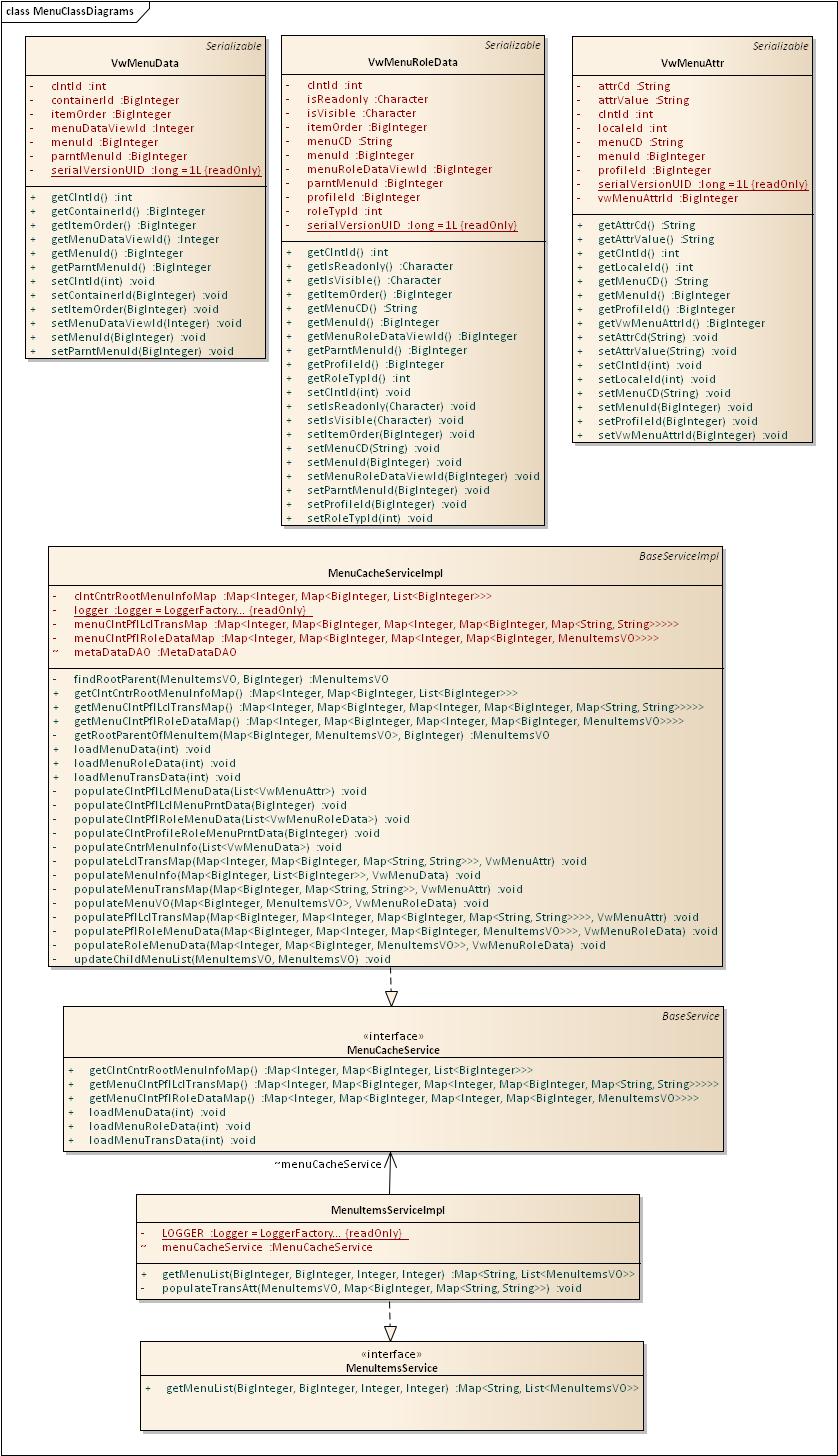
Profile, role based access control to Menu items.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Portal** | **Profile** | **Role** | **Menu Item** | **IsVisible** |
| Cust. Mgmt. | Verio | Cust. Admin | Party | Y |
| Cust. Mgmt. | Verio | Cust. Admin | Party Relations | Y |
| Cust. Mgmt. | Verio | Cust. Admin | Customer Account | Y |
| Cust. Mgmt. | Verio | Cust. Admin | Account | Y |
| Cust. Mgmt. | Verio | Cust. Admin | User | Y |
| Cust. Mgmt. | Verio | Cust. Admin | Contacts | N |
| Cust. Mgmt. | Verio | Cust. Admin | Payments Info | Y |

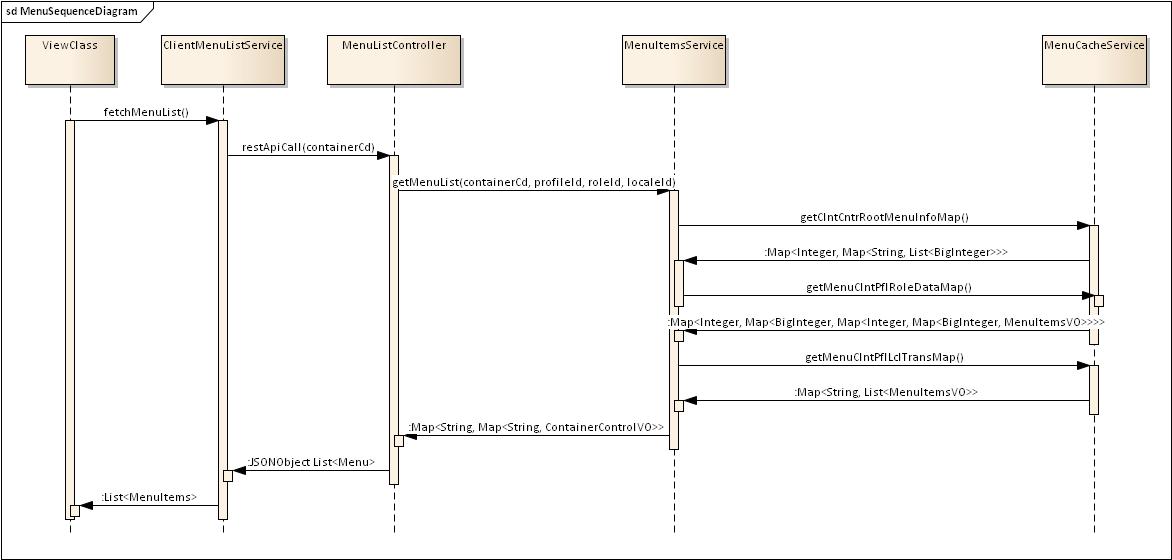
Attributes like label etc. can be customized based upon profile and locale.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Portal** | **Profile** | **Locale** | **MenuCode** | **AttributeCode** | **AttributeValue** |
| Cust. Mgmt. | Verio | en\_US | Party | Label | New Party |
| Cust. Mgmt. | Verio | en\_US | PartyRel | Label | Party Relation |
| Cust. Mgmt. | OEM | en\_US | Party | Label | Create New Party |
| Cust. Mgmt. | OEM | en\_US | PartyRel | Label | Party Relationship |

##### Class Diagram



##### Sequence Diagram



### Messages Implementation

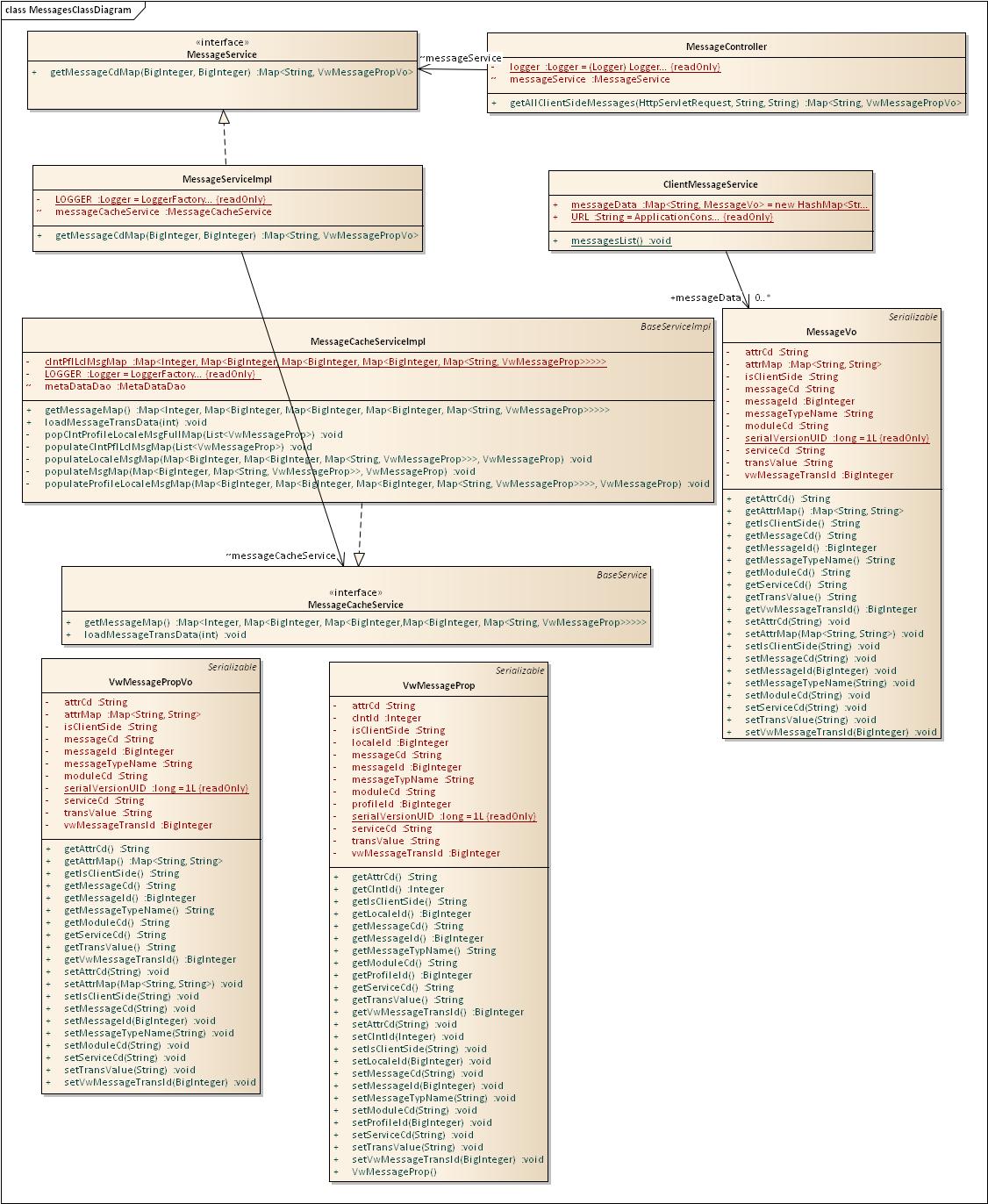
Framework supports the feature of customizing messages to be displayed to user. Some of the message types include validation messages, alerts, confirmation messages and error messages etc.

##### Database Design

Database design support messages to available for different applications

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Portal** | **Profile** | **Locale** | **Message Code** | **Message Type** | **AttributeCode** | **AttributeValue** |
| Cust. Mgmt. | Verio | en\_US | CMS-UR-003-E | Error | MsgTitle | Error Occurred |
| Cust. Mgmt. | Verio | en\_US | CMS-UR-003-E | Error | MsgDesc | Some error occurred during processing, please contact system administrator. |
| Cust. Mgmt. | OEM | en\_US | CMS-UR-003-E | Error | MsgTitle | System Error |
| Cust. Mgmt. | OEM | en\_US | CMS-UR-003-E | Error | MsgDesc | Oops! Some error occurred while processing your request, please try again later or call customer care. |

##### Class Diagram



### GXT Controls Implementation

#### Class Diagram

#### Sequence Diagram

<to be elaborated>

## SSO/Session management

<To be elaborated is sprint 3>

## Caching Design

The container/control, menu and message information for various profile/roles and profile locale combinations is being cached as java objects in the framework server layer at server start up. For every profileall this information is also inherited from parent account’s profile at start-up itself to enable faster response time on user request to render various pages on GUI.

Here are java objects that are being cached:

**General:**

// All downward account-profiles of a given account-profile.

Map<AccountProfile, List<AccountProfile>>acctPflChildMap;

// Parent account-profile of a given account-profile

Map<AccountProfile, AccountProfile>pflParentMap;

**Container/Control:**

// Master container-control list

Map<Integer, Map<String, List<BigInteger>>>clntCntrCtrlMap;

// Accessible control elements per profile/role

Map<Integer, Map<AccountProfile, Map<Integer, Map<BigInteger, ContainerControlVO>>>>clntPflRoleCtrlMap;

// Locale based data for all the elements based upon profile and locale.

Map<Integer, Map<BigInteger, Map<Integer, Map<BigInteger, Map<String, String>>>>>clntPflLclDtlMap;

//Non locale based attributes for all the controls

Map<Integer, Map<BigInteger, Map<String, String>>>nTransCtrlAttrMap;

**Menus:**

// Root menu list of all the containers

Map<Integer, Map<BigInteger, List<BigInteger>>>clntCntrRootMenuInfoMap;

//Menu details (with hierarchy) per profile-role

Map<Integer, Map<BigInteger, Map<Integer, Map<BigInteger, MenuItemsVO>>>>menuClntPflRoleDataMap;

//Menu lables and other locale based data per profile-locale

Map<Integer, Map<BigInteger, Map<Integer, Map<BigInteger, Map<String, String>>>>>menuClntPflLclTransMap;

**Messages:**

// profile locale based data for messages

Map<Integer, Map<BigInteger, Map<BigInteger, Map<BigInteger, Map<String, VwMessageProp>>>>>clntPflLclMsgMap;

## Integration with HAT (Help Authoring Tool)

### Database Design

### View Integration with HAT

#### Context Sensitive Help

*<To be elaborated in Sprint 3>*

## Logging

### Local Logging

#### AOP Implementation

As part of this, entry and exit points for individual methodswill be captured using around advice.

*<To be elaborated in Sprint 3>*

#### Types of Logging

*<To be elaborated in Sprint 3>*

### Remote Logging

#### Service Invocation

*<To be elaborated in Sprint 3>*

#### Remote Logging Sequence Diagram

*<To be elaborated in Sprint 3>*

## Exception Management

### Exception Hierarchy

Exception would be handled by creating base exception class that would be inherited by all

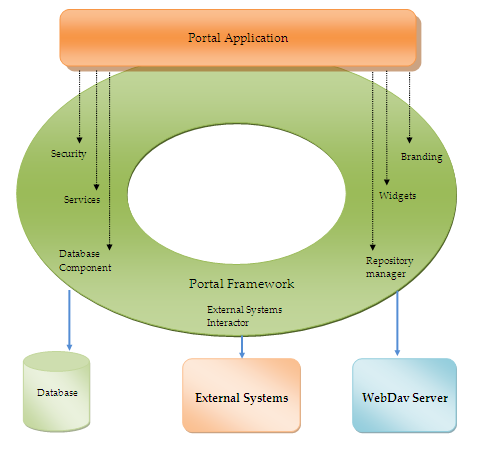
Child exceptions.It would contain error status codes that would represent the exception type.

Following types of exceptions can be handled:-

1. Data (Dao)related exceptions
2. Framework related exception

# Portal Application Design Reference

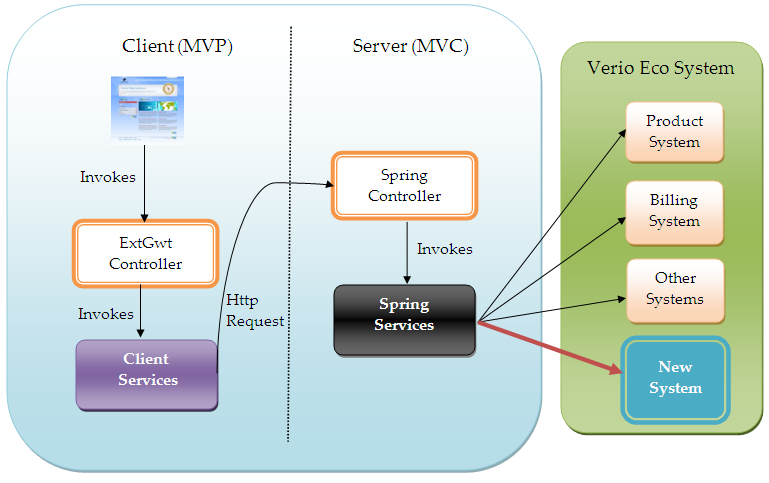
Portal application sits on top of portal framework architecture and will make use of the various components to render the UI. As the framework architecture is based on loosely coupled components spread around multiple layers, it will make use of the components in a standard fashion to maximize the usage of the components.



*Figure 7: Portal Design Reference*

# OSS-B (SOA) Integration Design

OSS-B Portal Framework provides a simple way to integrate a new application or system in the portal application by using the services provided by the new system.



*Figure 8: Portal Integration Architecture*

The above diagram represents how inclusion of new system can integrate with components of OSS-B Portal application in a systematic manner.

Steps to accommodate the services of a new system:

* Add the new system in Verio ecosystem with the services exposed by the new component available to the OSS-B portal application in a consistent predefined manner. (The protocol and the format of communication should be clearly defined)
* Create a new module in the “**spring services**” component to encapsulate a single business process that may include either directly accessing the services of new system or a sequential combination of other existing module services in addition to achieve a single business use case functionality.
* Create a new module or modify an existing module in “**client services**” component to include the new functionality.
* Render the existing UI or change the UI as per the change in data models or with the introduction of new data models

# 