NITIDO

Venue 2.0 Framework

Venue 2.0 is a front-end framework for building high-performance rich internet applications (RIA) that run seamlessly in a wide variety of Internet browsers. Its power, flexibility and ease make it a tool of choice for building large-scale user-centric application for the web.

Write Once, Perform Everywhere.

Introduction

Building browser compatible, performing, and large-scale rich internet applications is difficult. Issues of browser fragmentation, performance, skills availability and difficulties in maintenance are widely acknowledged. In fact, anyone, who has managed, written or in some way been involved in the development of any RIA, typically encounters:

- Fragmentation/Inconsistency of JavaScript, CSS and HTML support across browsers
- Deeply inter-twined technologies (Js, CSS, HTML) within application (business) code that 1) forces developers to know all three subjects in order to write code, 2) increases the number of lines of code and is error-prone.
- Massive maintainability issues. Same bugs are created, over and over (as much as 30% of the time) that results in untrusted code, lengthy QA cycles, reduced developer productivity and increased cost.

At Nitido we have recognized this challenge and been working at resolving it for years now. Our first effort is Venue 1.0 (described elsewhere). Venue 1.0 advanced the state of the art significantly and allowed us to create applications like Verizon's webmail and many others. From that experience we have advanced the way we build applications and thus evolved Venue 1.0 to Venue 2.0.

Venue 2.0 adds the following to Venue 1.0:

- Object Oriented (OO) UI development
- Clean and reusable UI abstraction through objects that have been pre tested on supported browsers.
- A robust communication and synchronization subsystem to handle client to server communication in large-scale environment's like Verizon's VOL.
- Many others.

Venue 2.0 improves on Venue 1.0 by providing key benefits:

- Shorten development cycles: The primary goal of Venue 2.0 is to ***shorten development cycle*** of RIA functional user interfaces.
- Lightweight: Venue 2.0 is designed for a small footprint, so as not to hinder the application performance in any way.
- Unobtrusive: Venue 2.0 is designed to avoid any constraints on the application's

business logic.

- No dependence on HTML: Venue 2.0 inherently removes dependence on having to write raw HTML code based on the fact that every UI construct is rendered internally as HTML elements.

- Extensible: Venue 2.0 provides hooks to extending the framework to fit your needs. In most cases, a simple matter of extending a class is all that is needed. What this means is that even though a feature may not be available in the framework ***"out of the box"***, it can most assuredly be added easily by extending the framework.

As an example, the following table compares and contrasts some of the differences between Venue 1.0 and Venue 2.0.

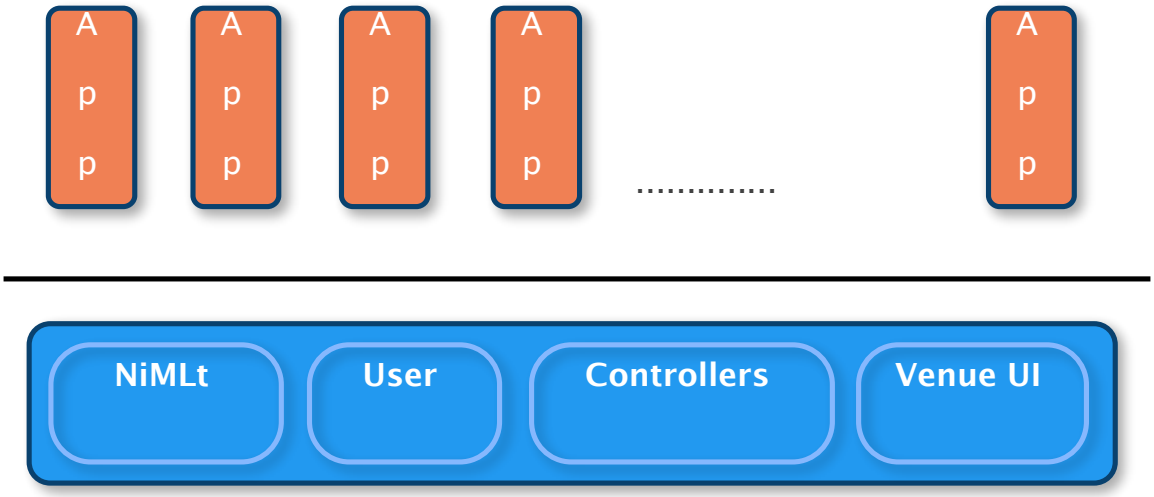
Venue 1.0	**Venue 2.0**
HTML templates required	No HTML templates needed

Code to create a Label:
 	
`var _label = document.createElement( "label" );`
 	
`_label.style.width = "100px";`
 	
`_label.id = "id-one";`
 	
`_label.innerHTML = "My 1st Label";`
 	
`Event.addListener( _label, "onlick", this.callMe1 );`
	Code to create a Label:
	
	`var _label = new NIM.ui.Label( { id: "id-one", label: "My 1st Label", styleClass: "myLabel" onclick: { fn: this.callMe1 } );`

Direct Manipulation of the HTML DOM	Manipulation of the HTML DOM handled internally but exposed through a set of APIs

Architecture

The Venue 2.0 architecture is depicted in the following picture:



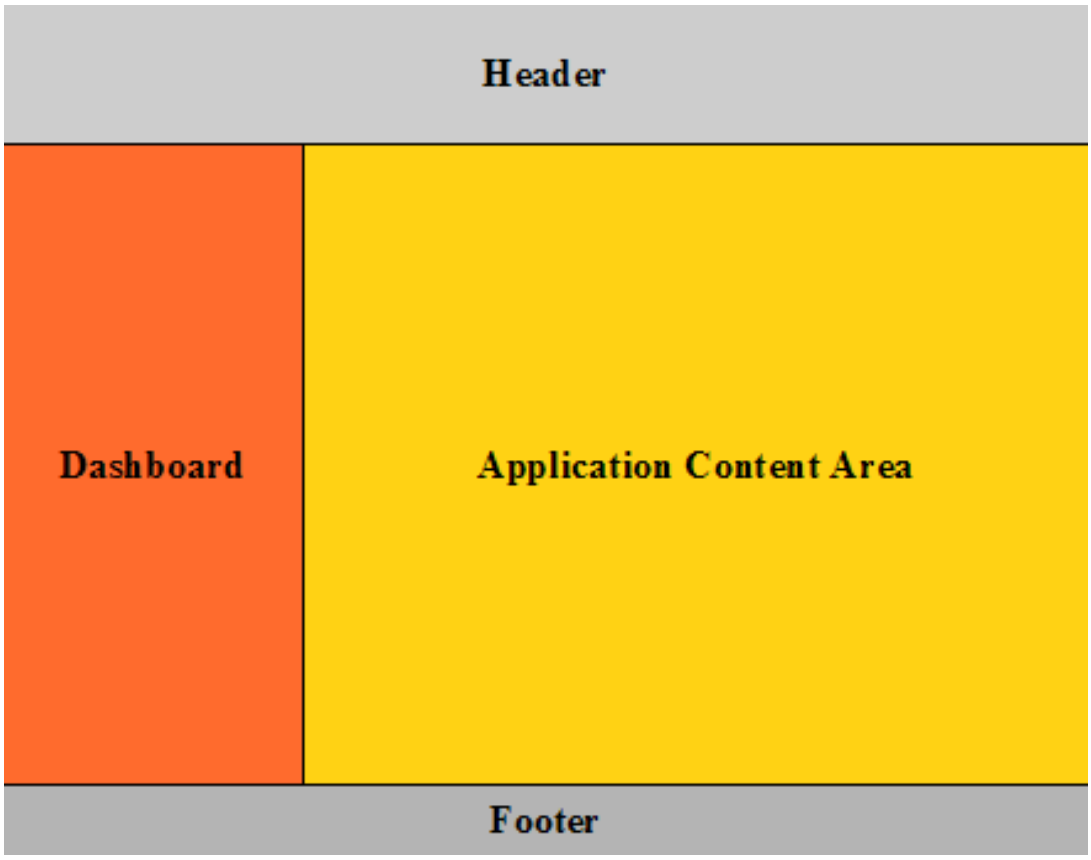
- App 1 … App N: Applications written on top of the Venue 2.0 framework.
- NiMLt: Provides access to core functions for authentication, authorization, communication, and logging.
- User: Object representation of an authenticated user.
- Controllers: Provides methods to perform CRUD operations of data objects.
- Venue UI: Provides a global Canvas object in which the application interface lives and Provides UI Constructs to build interfaces

Functional components of a Venue 2.0 Application

Venue 2.0 automatically provides a ***Dashboard*** and an ***Application Content Area*** for every application.

- **Dashboard:** The purpose of the dashboard is to provide an interface to navigate the collections/groups of items(ex: folders in email/albums for photos/groups for addressbook) that pertain to the given application. It is up to the developer if he/she would like to use it or not.

- **Application Content Area:** This area is where the application "content" is rendered and most of its interactions happen.



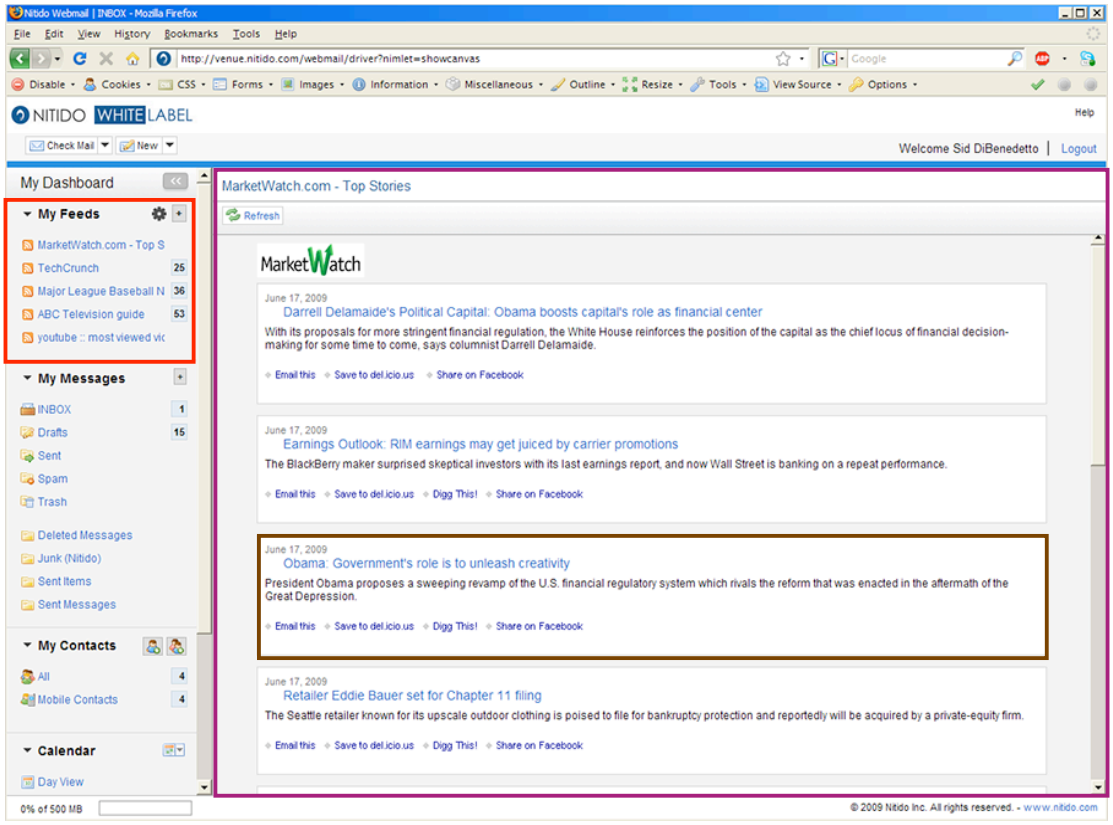
In addition, Venue 2.0 provides some UI constructs ***"out of the box"***. Below are some of the more widely used constructs which are seen and noted in screen shots(provided below) of a sample application.

Button, ChoiceGroup, Image, Label, Link, MediaPlayer, Menubar, Panel, PopupPanel, ScrollPanel, Table, TextField and many others.

Working Example Code

A sample RSS application that is written entirely using VenueScript can be found in **Appendix A**. The original source code of this RSS application written in Venue 1.0 was ***1650*** lines of code compared to ***365*** lines in Venue 2.0. That represents approximately a ***75%*** decrease in written code.

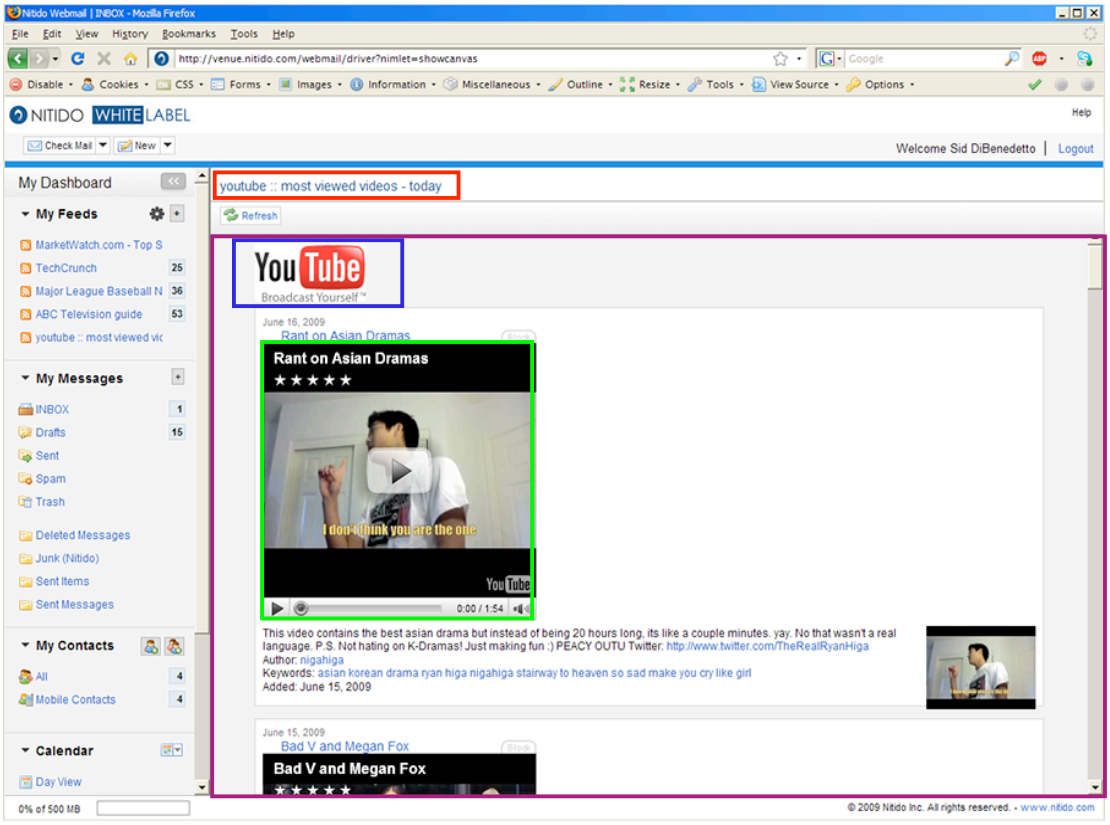
Below are some screen shots of the aforementioned RSS application code deployed within a Venue 1.0 Webmail instance.



In this screen shot, the red outline shows the dashboard instance provided by Venue 2.0 for the RSS application. In this case, the dashboard contains a list of RSS feeds and the number of unread/new articles within that feed. You will notice that in the dashboard, there are two icons (cogwheel and plus symbol) on the top right. These represent functionality native to the dashboard: ***manage*** my feeds and ***add a new*** feed (in this case).

The purple outline represents Venue 2.0's native application content area. This is where the application and its content gets rendered. The developer has free reign in this area since Venue 2.0 ***"sandboxes"*** the application here.

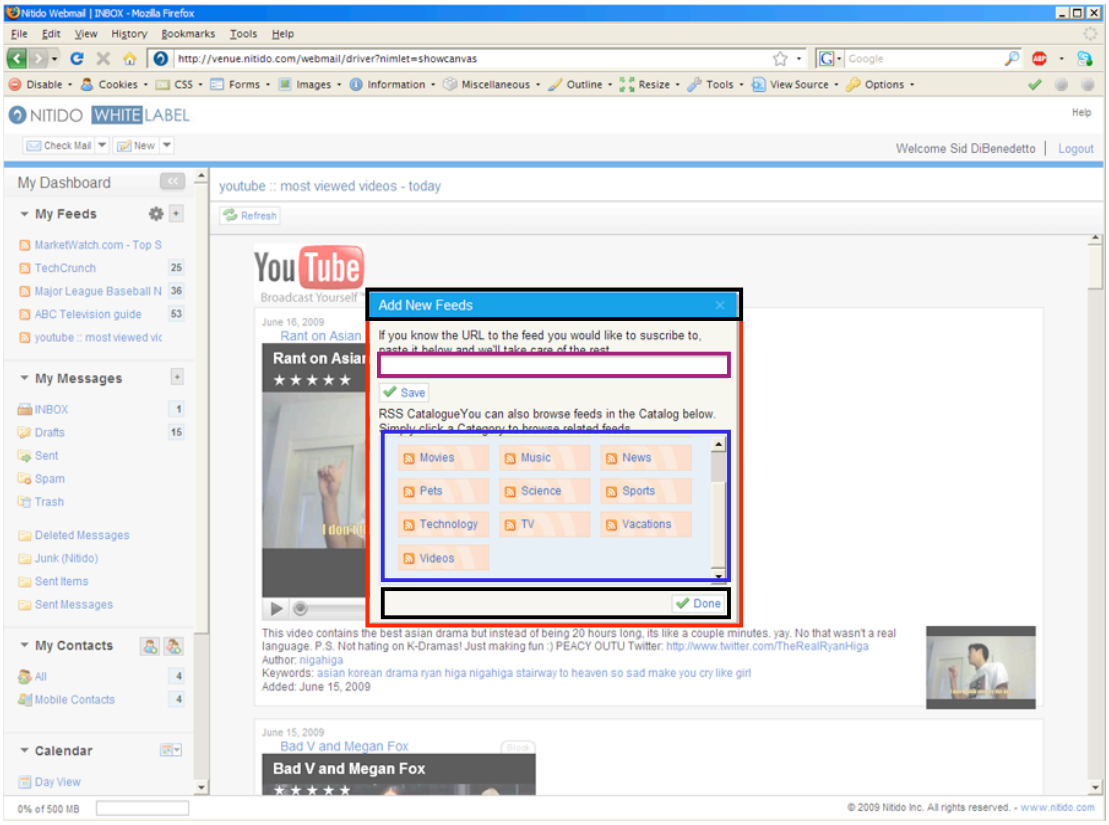
The brown outline is an example of the HTMLRenderer component. Basically, in this sample, the abstract of the article is in HTML format. So the developer just has to pass this content to the UI component and it will be rendered as is.



In this screen shot, the red outline shows an instance of a Label component. It is pretty much straight forward. The blue outline shows an instance of a Image component. It, too, is pretty much straight forward.

The green outline is an example of the MediaPlayer component. This is a special component in that it will render the proper type of player for the content given to it. This eliminates the need for developer to explicitly create a specific player.

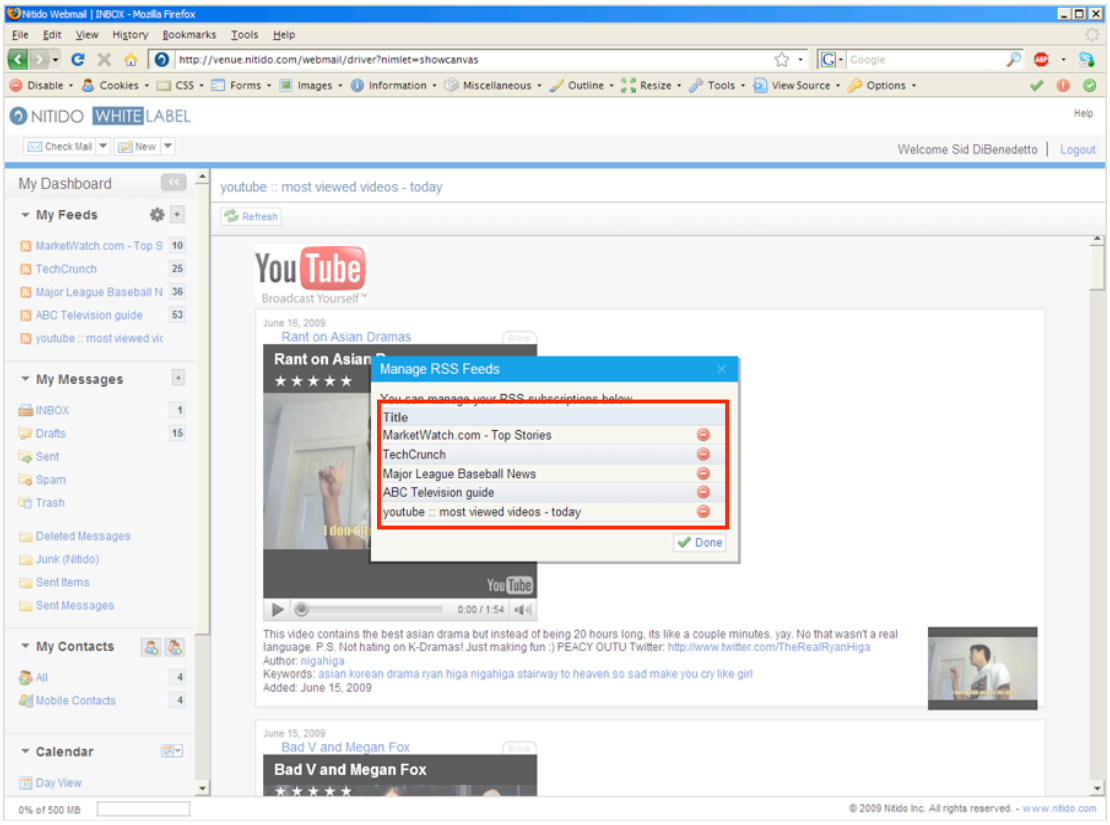
The purple outline is an example of the ScrollPanel component. This UI element alleviates the necessity for the developer to manually keep track of content in a Panel and enable scrolling when needed. It does all the calculations internally and enables/disables scrolling automatically.



In this screen shot, the red outline shows an instance of a PopupPanel. This element allows for the concept of model windows in a Venue 2.0 application. This UI component also provides specific layout support natively as outlined in black. The top black outline shows the header of the panel where the developer can specify a title. It also by default, allows the developer to state which window operation icon to display(minimize, maximize, close). The bottom black outline provides a Menubar component, in which the developer can name the buttons he/she would like to show.

The purple outline shows an instance of a TextField component. This UI element does also support a number of additional features including ***pre-populated default text*** and ***input masks.***

The blue outline is an example of the Table component. In this instance, the rendered output of the table is not a standard layout(has no header row and grid lines). It is also placed in a ScrollPanel element, which shows the nesting capabilities of UI contructs.



The red outline is another example of the Table component. Yet in this instance, the rendered output of the table follows a more traditional layout(has a header row and grid lines).

Appendix A

```
( function(){

    var user                        = nim.getUser(),
          rssCon                    = user.getController( "RSS" ),
          catalogPanel              = null,
          menuBarPanel              = new NIM.ui.Panel( {
styleClass: "actionsBar" } ),
          menuBar                   = new NIM.ui.MenuBar(),
          feedPanel                 = new NIM.ui.Panel( { id:
"feedViewer" } ),
          scrollPanel               = new NIM.ui.ScrollPanel( {
scrollX: false, scrollY:
                                                  true, autoHeight:
                                              true } ),
          articleList               = new NIM.ui.Panel( {
styleClass: "articleList" } ),
          titleBar                  = new NIM.ui.Panel( {
styleClass: "titleBar" } ),
          feedTitle                 = new NIM.ui.Label( {} ),
          feedImage                 = new NIM.ui.Image({}),
          categoryPanel             = null,
          catalogCategoriesTable    = null,
          categoryChoiceGroup       = null;

    // define application constructor
    NIM.applications.RSS = function(){
    };

    // extends the base application class
    NIM.extend( NIM.applications.RSS, NIM.core.BaseApplication, {

        // required API methods
        init: function(){

            var catalogMenuBar;

            //reference to the dashboard instance specific to
this application
            this.dashboard =
NIM.ui.DashboardManager.getDashboardInstance( this, { title:
                                                    "My Feeds"
                                                    } );

            menuBar.append( { label: "Refresh", icon: "refresh",
onclick: { fn:
                    this.refreshFeed, scope: this, obj: { arg1:
                    "Feed is Refreshed" } } }  );
```



```
            this.parent.append( titleBar );
            this.parent.append( menuBarPanel );
            this.parent.append( feedPanel );

            titleBar.append( feedTitle );
            feedPanel.append( scrollPanel );
            scrollPanel.append( feedImage );
            scrollPanel.append( articleList );
            menuBarPanel.append( menuBar );
            this.loadFeeds();
        },

        destroy: function(){
            user = null;
            rssCon = null;
        },

        //auto callback from dashboard when an item is selected
from the list
        handleDashboardSelection: function( selectedDashboardItem
){
            if ( selectedDashboardItem.id !=
rssCon.getSelectedFeed() ){
                // load feeds
                feedTitle.setLabel( "Loading " +
selectedDashboardItem.obj.feedObj.title
                        + "..." );
                this.loadFeedArticles(
selectedDashboardItem.obj.feedObj, true );
            }

            this.selectedNode = selectedDashboardItem;
        },

        //auto callback from dashboard to configure the add new
item popup-dialog
        handleAddNewItem: function(){
            if ( ! this.addNewItemPopupPanel ){
                // build catalog popup panel
                this.addNewItemPopupPanel = new
NIM.ui.PopupPanel(
                    {
                        title: "Add New Feeds",
                        icon: "icon",
                        buttons: [
                            { label: "Done", icon: "ok", type:
"done", onclick: { fn:
                      this.catalogDelegate, scope: this, obj: {
                      arg1: "custom argument for cancel" } } }
                      //'type' is used for pre-determined action
```



```
                        ],
                        width: "400px"
                    }
                );

                this.addNewItemPopupPanel.append( new
NIM.ui.Label( { label:
                    nimAM.catalogBrowser.TXT.webcomeText1, tag:
                    "p" } ) );
                this.feedTextField = new NIM.ui.TextField( {
width: "100%", icon: "rss",
                    styleClass: "rss-feed-url-input" } );
                this.addNewItemPopupPanel.append(
this.feedTextField );

                catalogMenuBar = new NIM.ui.MenuBar( { width:
"100%" } );
                catalogMenuBar.append( { label: "Save", icon:
"save", onclick: { fn:
                   this.addFeed, scope: this, obj: { txtFeed:
                   this.feedTextField } } } );
                this.addNewItemPopupPanel.append( catalogMenuBar
);
                this.addNewItemPopupPanel.append( new
NIM.ui.Label( { label:
                   nimAM.catalogBrowser.TXT.titleCatalog, tag:
                   "h2" } ) );
                this.addNewItemPopupPanel.append( new
NIM.ui.Label( { label:
                   nimAM.catalogBrowser.TXT.webcomeText2, tag:
                   "p" } ) );

                catalogPanel = new NIM.ui.ScrollPanel( { title:
"RSS Catalog", scrollX:
                   false, scrollY: true, autoHeight: true,
                   styleClass: "rss-catalog-list" } );
                catalogCategoriesTable = new NIM.ui.Table( {
cols: 3 } );

                this.addNewItemPopupPanel.append( catalogPanel );

                rssCon.loadCatalog( true, { obj: this, fn:
"renderCatalog" } );
            } else {
                this.addNewItemPopupPanel.show();
            }
        },

        //auto callback from dashboard to configure the manage
popup-dialog
        handleManage: function(){
```



```
            return {
                    title: "Manage RSS Feeds",
                    label: "You can manage your RSS subscriptions
below",
                    table: { columnDefs : [ { label: "Title" }, {
label: "" } ],
                     reordering: true },
                    buttons: [ { type: "done" } ] };
        },

        // helper/callback methods
        displayDashboardException: function(){
            Dashboard.displayError();
        },

        initDashboard: function(){
            var feeds           = rssCon.getFeeds(),
                dashboardItem,
                dashboardItems  = [],
                feed;

            for ( var i=0, j=feeds.length; i < j; i++ ){
                feed = feeds[i];
                dashboardItems.push( new NIM.ui.DashboardItem( {
icon: "rss", label:
                     feed.title, counter: feed.numofarticles, id:
                     feed.position, obj: { feedObj: feed } } ) );
            }

            this.dashboard.append( dashboardItems );
        },

        updateDashboard: function( args /* event arguments passed
by controller */ ){
            var feeds           = rssCon.getFeeds(),
                dashboardItem,
                dashboardItems  = [],
                feed;

            for ( var i=0, j=feeds.length; i < j; i++ ){
                feed = feeds[i];
                dashboardItems.push( new NIM.ui.DashboardItem( {
icon: "rss", label:
                   feed.title, counter: feed.numofarticles, id:
                   feed.position, obj: { feedObj: feed } } ) );
            }

            this.dashboard.clear();
            this.dashboard.append( dashboardItems );
        },
```



```
        handleDeleteItem : function( feed ) {
            rssCon.deleteFeed( feed.feedObj.position, { obj:
this, fn:
              "updateDashboard" } );
        },

        handleReorderItems: function( feedlist ) {
        },

        displayException: function( e ){
            canvas.error( "Error", e.msg, false, {
                buttons: [
                    {
                        text : "Ok",
                        styleClass : "roundButton"
                    }
                ]
            } );
        },

        populateArticlePane: function(){
            var feedObj     = rssCon.getFeedObject(
rssCon.getSelectedFeed() ),
                articles,
                article,
                articlePanel,
                articleDate,
                articleTitle,
                articleBody,
                articleEmbed;

            // get articles array from controller
            articles = rssCon.getArticles( { feedurl:
feedObj.link, position:
               rssCon.getSelectedFeed(), defaultFeed: false,
               page: 0 } );
            if ( articles ){
                articles = articles.getItems();
            }

            var articlePanels = [];

            // loop through and draw articles
            for ( var i=0, j=articles.length; i < j; i++ ){
                article = articles[i];

                articlePanel = new NIM.ui.Panel( { styleClass:
"feed-article" } );
                if ( article.date ){
                    articleDate = new NIM.ui.Label( { label: new
                     Date( article.date ).format( "MMM d, y" ),
```



```
                    tag: "h1", styleClass: "article-date" } );
                    articlePanel.append( articleDate );
                }
                articleTitle = new NIM.ui.Link( { label:
article.title, tag: "h2", href:
                    article.link, target: "_blank", styleClass:
                    "article-title" } );
                articlePanel.append( articleTitle );

                if( article.enclosure && (
article.enclosure.length > 0 ) ) {
                    for( var x = 0, y = article.enclosure.length;
x < y; x++ ){
                        articleEmbed = new NIM.ui.MediaPlayer(
article.enclosure[x] );
                        articlePanel.append( articleEmbed );
                    }
                }

                articleBody = new NIM.ui.HTMLRenderer( { content:
                         article.description } );
                articlePanel.append( articleBody );
                articlePanels.push( articlePanel );
            }

            // clear scroll panel's contents and redraw
            articleList.clear();

            feedImage.setSource( feedObj.imglink );
            feedTitle.setLabel( feedObj.title );

            articleList.append( articlePanels );

            // update counter of dashboard node
            this.selectedNode.setCounter( 0 );
        },

        loadFeeds: function(){
            rssCon.loadFeeds(true, { obj: this, fn:
"initDashboard" } );
        },

        refreshFeed: function(e, args){
            feedTitle.setLabel( "Refreshing " +
this.selectedNode.label + "..." );
            var feedObj = rssCon.getFeedObject(
rssCon.getSelectedFeed() );
            this.loadFeedArticles( feedObj, true );
        },

        loadFeedArticles: function( feed, purgeCache ){
```



```
            rssCon.setSelectedFeed( feed.position, { position:
feed.position }, true );
            rssCon.loadArticles( { feedurl: feed.link, position:
feed.position }, false,
                purgeCache, { obj: this, fn:
                "populateArticlePane" } );
        },

        /**
            RSS Catalog-related methods
         */
         catalogDelegate: function(e){
             // target is a reference to the Label object
             rssCon.loadCategory( target.id, { obj: this, fn:
"renderCategory" } );
         },

         renderCatalog: function(){
             var catalogs            = rssCon.getCatalog(),
                 categoryCatalog     = [],
                 catalog;

             for ( var i=0, j=catalogs.length; i < j; i++ ){
                 catalog = catalogs[i];
                 categoryCatalog.push( new NIM.ui.Link( { label:
catalog.title, icon:
                  "rss", styleClass: "catalog-category", id:
                  'catalog_' + catalog.id, onclick: { fn:
                  this.displayFeedCategory, scope: this, obj: {
                  catalog_id: 'catalog_' + catalog.id } } } ) );
             }

             catalogCategoriesTable.append( categoryCatalog );
             catalogPanel.append( catalogCategoriesTable );

             window.table = catalogCategoriesTable;

             this.parent.append( this.addNewItemPopupPanel );

             this.addNewItemPopupPanel.show();
         },

         // TODO: Figure out how to roll in a delegate function
for the entire
         // Catalog PopupPanel
         displayFeedCategory: function(e, args){
             var catalog_id = args.catalog_id.split( "_" )[1];
             rssCon.loadCategory( catalog_id, true );
         },

         renderCategory: function(){
```



```
            var category   = rssCon.getSelectedCategory(),
            feedList       = [],
            feed,
            feedCheckbox,
            categoryMenu,
            saveBtn,
            backBtn;

            // clear previous category list (if it exists)
            if ( categoryPanel ){
                categoryPanel.destroy();
            }

            var categoryListPanel = new NIM.ui.ScrollPanel( {
title: category.title,
                 scrollX: false, scrollY: true, height: "100%",
                 autoHeight: true, styleClass: "categoryListing"
                 } );
            var categoryListTable = new NIM.ui.Table( { cols: 1,
                 styleClass: "category-list-form" } );
            categoryListPanel.setTitle( category.title );

            for ( var i=0, j=category.feeds.length; i < j; i++
){
                feed = category.feeds[i];

                // TODO: CheckBox should be renamed to Checkbox
                feedCheckbox = new NIM.ui.CheckBox( { label:
feed.title, value:
                 feed.link, name: "feed_" + category.id + "_" +
                 i, checked: false } );
                feedList.push( feedCheckbox );
            }

            categoryListTable.append( feedList );
            categoryListPanel.append( categoryListTable );

            // store feedList to be accessed when clicking the
'save' button
            this.selectedFeeds = feedList.concat();   // not a
true reference

            // menu bar & buttons
            categoryMenu = new NIM.ui.MenuBar();
            saveBtn = { label: "Add", icon: "add", onclick: {
fn:
                 this.addFeedsFromCategory, scope: this, obj: {
                 arg1: "custom argument that gets passed to my
                 callback function" } } };
            backBtn = { label: "Back", icon: "back", onclick: {
fn:
```



```
                this.showCatalog } };
            categoryMenu.append( saveBtn );
            categoryMenu.append( backBtn );

            // panel that contains the entire categories content
(including menu bar)
            categoryPanel = new NIM.ui.Panel( { height: "100%" }
);
            categoryPanel.append( categoryListPanel );
            categoryPanel.append( categoryMenu );

            this.addNewItemPopupPanel.append( categoryPanel );

            // hide catalog and show category list
            this.showCategory();
        },

        showCatalog: function(){
            categoryPanel.setVisible( false );
            catalogPanel.setVisible( true );
        },

        showCategory: function(){
            categoryPanel.setVisible( true );
            catalogPanel.setVisible( false );
        },

        addFeed: function(e, args){
            var feedURL = args.txtFeed;
            if ( feedURL && feedURL.getValue() != "" ){
                // check if valid url
                //

                // add feed
                rssCon.addFeed( feedURL.getValue(), { obj: this,
fn:
                ["updateDashboard","showCatalog"] } );
                this.feedTextField.setValue( "" );
            } else {
                // alert user that field is empty
            }
        },

        addFeedsFromCategory: function(){
            var feedsToAdd = [],
                feed,
                i,
                j;

            for ( i = 0, j = this.selectedFeeds.length; i < j;
i++ ){
```



```
                feed = this.selectedFeeds[i];

                if ( feed.isChecked() ){
                    feedsToAdd.push( feed.getValue() );
                }
            }
            rssCon.addFeed( feedsToAdd, { obj: this, fn:
                ["updateDashboard","showCatalog"] } );
        }

} );  })();
```


Glossary

API	Application Programming Interface
CRUD	Create, Retrieve, Update and Delete – operations usually performed on data objects
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	Hypertext Mark-up Language
JS	Javascript
OO	Object Oriented
RIA	Rich Internet Application
UI	User Interface
VenueScript	Term used to describe the language used to develop applications on the Venue 2.0 Framework