

Arena Custom Module Development

Version 2011.1b
Last updated 10/18/2011
Maintained by the Community

Arena Community Documentation
Arena 2011.1



Revision History

Version	Date	Editor(s)	Description
1.00	6/10/2007	Nick Airdo	First, final version with input by David T, Jon E and the Arena development team.
2.00	10/29/2007	Nick Airdo	Updated for the Arena 2007.2 ("2.0") build.
2008.1	3/21/2008	Nick Airdo	Updated for Arena 2008.1 by Jesse Brown, Jon Edmiston, Jeff Maddox, Mac McGreger, Caleb Tucker, David Turner
2008.2	9/25/2008	Nick Airdo, Jesse Brown, David Turner, Jeff Maddox	Updated for the Arena 2008.2 release. See Release Notes for more significant changes.
2009.1	2/25/2009	Nick Airdo, Jesse Brown	Updated for the Arena 2009.1 release. See Release notes for more significant changes.
2011.1	9/15/2011	Nick Airdo	Redo of the Dev Environment Setup
2011.1b	10/18/2011	Nick Airdo	Include details on grouping module settings.

Table of Contents

RELEASE NOTES	5
WELCOME TO THE COMMUNITY	6
DEVELOPMENT ENVIRONMENT SETUP	7
Using Web PI to install Visual Web Developer Express	7
Obtaining Your Assigned ORGID	9
ARENA ARCHITECTURE	11
What Are Arena Modules?	11
Module Settings	12
The "My Portal" Framework	18
Context Sensitive Content	21
Security and Access Control	23
DEVELOPER GUIDELINES	24
Policies	24
Naming Conventions	25
Source Control & Comment Blocks	27
Globalization and Localization	28
MODULE CREATION EXAMPLE	29
Creating Custom Classes (Optional)	29
Creating Custom User Controls	30
Adding the New Module to Arena	31
Create New Pages	31
Create Module Instances on a Page	32
Module in Action	34
PACKAGING YOUR MODULE FOR THE COMMUNITY	35
Zip Folder Structure	35
Module Export	35
ARENA UI TOOLBOX	37
Arena:BooleanImageColumn	37
Arena:SelectColumn	37
Arena:DataGrid	38
Document Picker	40
Arena:ImagePopupColumn	41
ModalPopup	42
ModalPopupIFrame	43
OptGroupDropDownList	44
Person Search Window	45
Arena:DateTextBox	47
Arena:LookupDropDown	48
Arena:PhoneTextBox	50
Arena:ProfilePicker	51
UI LOOK AND FEEL	52
ARENA AGENTS	53
Agent Settings	53

Agent Setting Attributes	54
REFERENCES	55
The.Community Website	55
Code Generation	55
Arena UI Style Guide	55
Photoshop Image Templates	55
Arena Import/Export Instructions.....	55
API Documentation	55
VSS keyword Expansion.....	55

Release Notes

2011.1 & 2011.1b

- Rewrote the Development Environment Setup section
- Added a section on grouping module settings
- Simplified the table of contents

2009.1

- Added Arena:SelectColumn to UI Toolbox section
- Added Globalization and Localization to Developer Guidelines
- Added Custom Organization Setting Naming Guidelines

2008.2

- Added this Release Notes section
- Added the Welcome section
- Added the Context Sensitive Content section
- Added Report Setting to the module settings section
- Added Web Services to the naming convention section
- Added information about LINQ in the Create Custom Classes section
- Added Arena:PhoneTextBox to UI Toolbox section
- Added Arena:ProfilePicker to UI Toolbox section
- Removed Arena:PersonPicker to UI Toolbox section

Welcome to the Community

Allow me to speak on behalf of the entire Arena Community when I say "Welcome to the Community!" What is going on here has been a personal dream of mine for several years, and I'm ecstatic to know another developer or church has joined the team.

Please take a minute right now to add you and/or your team of developers to the [Community Developers](#) Wiki page. When you're done with that just jump over to the [Announcements Forum](#) and let us know you're here. That's certainly something we all want to hear about!

The Arena community is only as successful as you and the other community developers will make it. It's going to be your mission to learn, help others and contribute as much as possible. As a case in point, this document is actually the result of a community effort. It exists to help you learn about important development details and to get you quickly started in Arena Module development.

I look forward to hearing from you soon!

In Christ,

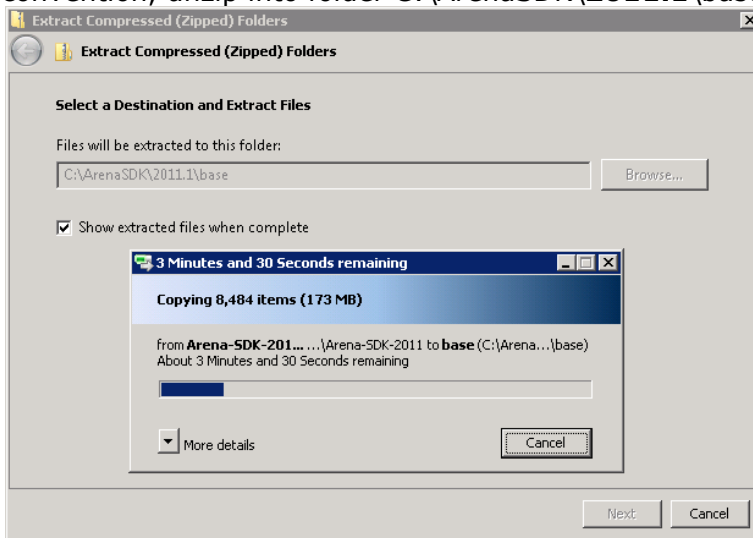


Nick Airdo

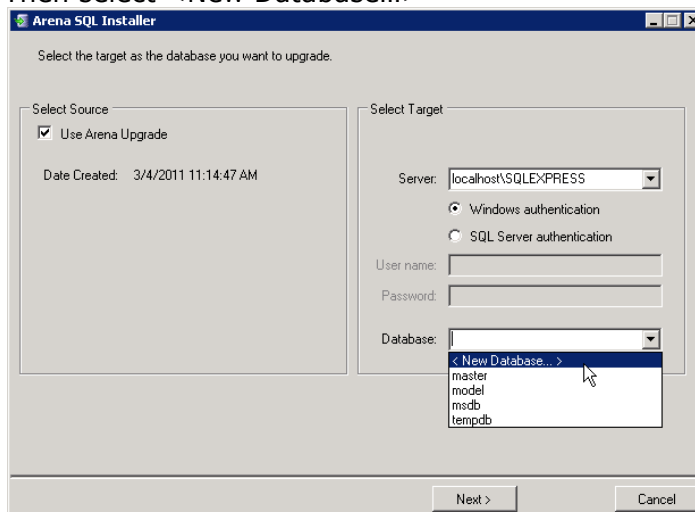
Development Environment Setup

Using Web PI to install Visual Web Developer Express

- 1) If you have Visual Studio installed, proceed to next step. If you don't own Visual Studio, you can install the free "Express" version using Internet Explorer and going to <http://bit.ly/WebDevExpress2010>. Once there you will use Microsoft's Web Platform Installer and select "Visual Web Developer Express 2010" (or later).
- 2) Download the [2011.1 Arena SDK from the community site](#) and then, by convention, unzip into folder C:\ArenaSDK\2011.1\base



- 3) Read the README.txt and follow the remaining steps which should include:
 - a) Run "SQL Installer\SQLInstaller no reports.bat"
 - i) if using SQL Express you should change "localhost" to "localhost\SQLEXPRESS"
 - ii) Then select <New Database...>



- iii) Again, by convention, use the name Arena_2011_1_Dev
- iv) Click "Next >", then "No" when asked if you want to define a specific user...

- b) Open "Web\Arena.sln" in Visual Studio
 - i) Click **Next >** to step through the conversion wizard. (No need to create a backup before converting)
 - ii) Click "Yes" if asked about "You are attempting to open a precompiled Web site."
 - iii) Do not allow VS to upgrade project to .NET Framework 4.0.
- c) Modify "Web\Arena\web.config" change the connection string as in:
"Data Source=(local)\SQLEXPRESS;Initial Catalog=Arena_2011_1_Dev;"
- d) Press F5 to run/debug Arena

Obtaining Your Assigned ORGID

Each organization that wishes to participate as an Arena Community Developer will be assigned a unique identifier, or an ORGID. The ORGID is to be used in all code items to avoid collisions in the various namespaces.

For example, "Church of the City, Memphis" might be assigned the identifier "cotcmem". In examples, if you see <ORGID>, then replace this with your organization id. Although the ORGID is case insensitive, it should be used with appropriate capitalization depending on code area (Namespace, table, sp, etc). Each area is covered in section

Naming Conventions.

Contact Arena support (1-888-77-Arena or support@arenachms.com) if your organization has not already been assigned an ORGID.

Arena Architecture

What Are Arena Modules?

Like other pluggable web application frameworks, Arena gives developers the ability to extend its core functionality through the development of custom "Modules". Typically, Modules consist of one or more .NET custom ASCX UserControls and one or more custom business object classes (in the form of assemblies) that are used by the UserControls. It is worth noting that the Arena Administration screens use the term "Modules" (🧩) to refer to only the UserControl component.

When properly designed, a module becomes a new generic piece functionality that can be reused in multiple places (instantiated on one or more pages) in the Arena application. For example, consider Arena's *Advanced Html Text* module. This single module represents the generic functionality of editing and displaying HTML in an area of a web page, however it is actually instantiated on hundreds or thousands of pages within the Arena installation. Each instance is able to distinguish itself from other *Advanced Html Text* instances (by its uniquely assigned module_id) and has its own set of associated data.

Additionally, a module can be written to have any number of configurable settings which are set by the Arena Administrator when they define new instances of your module.

Setting Name	Setting Value
Attendance Type ID <small>The Attendance Type ID to report on.</small>	<input type="text"/>
Allow Name Filter <small>Flag indicating if filtering by name should be available.</small>	<input type="radio"/> True <input checked="" type="radio"/> False
Area Detail Page (required) <small>The page that is used to display the area for a specific member.</small>	Area Details ...
Area Filter Parameter <small>If filtering by area, this is the querystring parameter name that will have the Area ID.</small>	<input type="text"/>
Communication Page <small>The page that is used for displaying a new communication.</small>	New Communication ✖ ...
Default Occurrence Type ID (required) <small>The default occurrence type ID that should be used when adding new occurrences.</small>	1
End Date <small>The end date to use for the report. Default is current date.</small>	<input type="text"/>
Gender <small>Optional parameter specifying a specific gender to display (0=Male, 1=Female)</small>	<input type="text"/>
Group Cluster Page <small>The group cluster page.</small>	Male Female (not set) ...

For example, the *"Small Group Tab Control"* module shown above requires the administrator to enter a default Occurrence Type ID and also allows the administrator the option of providing an End Date, Gender, reference to a related page and other settings. This is accomplished via the use of attribute declarations (a.k.a. module settings) in the UserControl.

Module Settings

A UserControl which extends the `Arena.Portal.PortalControl` class will usually have one or more module settings. Each setting has a title, description and a Boolean indicating whether the property is required. Good descriptive titles and descriptions is highly recommended since these are shown to the administrator when the module is added to a page/template.

Setting Name	Setting Value
General Settings	
Return Results Page Size <i>The number of items to display on each page of the result set (default = 10).</i>	<input type="text"/>
Styling	
Search Button CSS Class <i>CSS classname to use for the search button. Default 'searchBtn'</i>	<input type="text"/>
Search Button Image Path <i>Relative path to image for search button above results.</i>	<input type="text"/>
TextBox CSS Class <i>CSS classname to use for the search textbox. Default 'search-box'</i>	<input type="text"/>
WebService	
Search Server URL (required) <i>The URL of your MS Search Service (eg, http://mss01/_vti_bin/search.asmx)</i>	<input type="text" value="http://google.com/hidden-search-api"/>
WebService Account	

With the appropriate use of settings, a module can be used on multiple pages and each instance of the module can display different information. Grasping this concept is a key to a well designed Arena module.

To Group module settings by category names as shown in the image above, your module must also include the using `System.ComponentModel` directive and define a `Category` attribute with a grouping name argument for each module setting as seen here:

```
using System.ComponentModel;

// Styling Group
[TextSetting( "Search Button Image Path", "Relative path ...", false ), Category( "Styling" )]
public string SearchImagePathSetting { get { return Setting( "SearchImagePath", "", false ); } }

[TextSetting( "Search Button CSS Class", "CSS classname...", false ), Category( "Styling" )]
public string SearchButtonCSSClassSetting { get { return Setting( "SearchButtonCSSClass", "", false ); } }

[TextSetting( "TextBox CSS Class", "CSS classname ...", false ), Category( "Styling" )]
public string TextBoxCSSClassSetting { get { return Setting( "TextBoxCSSClass", "", false ); } }

// No group defined -- these go into a "General Settings" section automatically
[NumericSetting( "Return Results Page Size", "The number of ...", false )]
public string ReturnResultsPageSizeSetting { get { return Setting( "ReturnResultsPageSize", "10", false ); } }
```

A module setting's property name **MUST** be **exactly the same** as the actual setting name but end with the word "Setting" as shown in the examples.

The list below gives a brief description of each setting and example of how it might be used.

Boolean (True/False)

This setting will render as a true/false radio button and takes an additional option that specifies the default value.

```
[BooleanSetting("Enable Cool Option", "This is a true/false value. Defaults to true.", false, false)]
public bool EnabledSetting { get { return Convert.ToBoolean(Setting("Enabled", "true", false)); } }
```

Cluster

This currently renders as a numeric textbox. Use this if you need a value from smgp_group_cluster

```
[ClusterSetting("Cluster ID", "This is a cluster id", false)]
public string ClusterIDSetting { get { return Setting("ClusterID", "", false); } }
```

Cluster Type

This renders dropdown list of defined Cluster Types.

```
[ClusterTypeSetting("Cluster Type ID", "This is a cluster type id", false)]
public string ClusterTypeIDSetting { get { return Setting("ClusterTypeID", "", false); } }
```

Css

This currently renders as a textbox. Use this if you need a CSS file name.

```
[CssSetting("Css File Name", "This is a css file name", false)]
public string CssFileSetting { get { return Setting("CssFile", "", false); } }
```

Custom List

This will render as a drop down list using the values you specify. It accepts an array of display text and values for the dropdown, respectively.

```
[CustomListSetting("Custom List Value", "This represents a custom list of specific items", true, "", new string[] {
    "item 1", "item 2", "item 3" }, new string[] { "1", "2", "3" })]
public string CustomListSetting { get { return Setting("CustomList", "", false); } }
```

Date

This renders as a date picker textbox.

```
[DateSetting("Date Value", "This is a date", false)]
public string DateValueSetting { get { return Setting("DateValue", "", false); } }
```

Image

This currently renders as a textbox. Use this if you need an image file name.

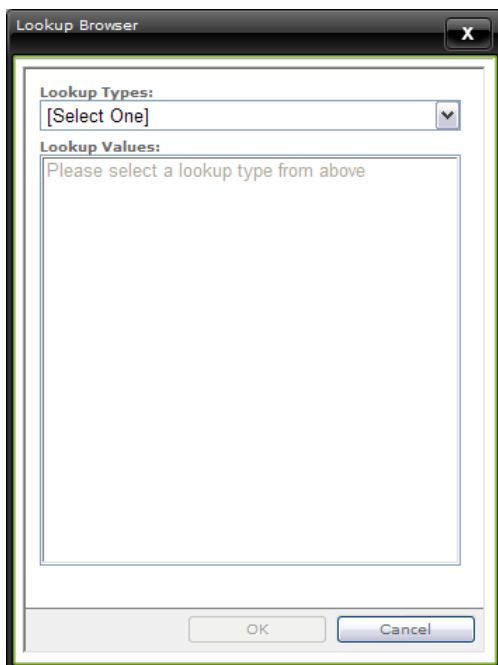
```
[ImageSetting("Image File Name", "This is an image setting", false)]
public string ImageFileNameSetting { get { return Setting("ImageFileName", "", false); } }
```

Lookup Value

This renders as a Lookup picker with a user selectable Lookup Type.

Lookup ID
This is a lookup id

(not set)

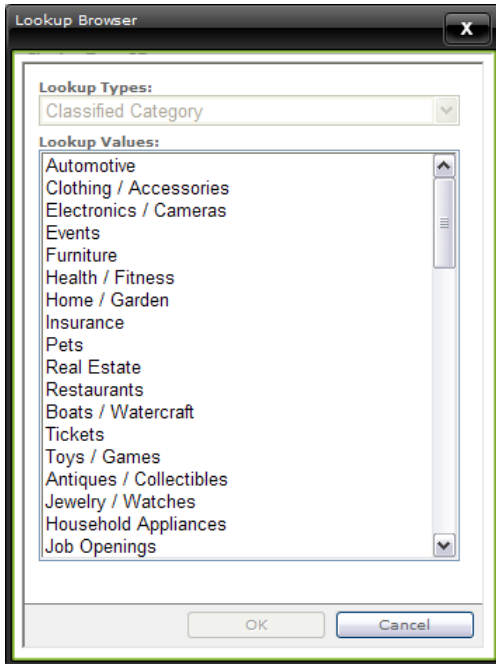


```
[LookupSetting("Lookup ID", "This is a lookup id", false)]  
public string LookupIDSetting { get { return Setting("LookupID", "", false); } }
```

Lookup (Constrained)

This renders as a Lookup picker that is constrained to a single Lookup Type. It accepts the GUID of a Lookup Type.

Lookup ID special (required)
This is a required lookup id constrained to a certain lookup type (not set)



```
[LookupSetting("Lookup ID special", "This is a required lookup id constrained to a certain lookup type", true,
"4C96A451-ADDE-435A-9D5D-D3609A909EB1")]
public string LookupID2Setting { get { return Setting("LookupID2", "", false); } }
```

Metric

This currently renders as a numeric textbox. Use this if you need an id from mtrc_mertic.

```
[MetricSetting("Metric ID", "This links to a metric id", false)]
public string MetricIDSetting { get { return Setting("MetricID", "", false); } }
```

Numeric

This renders as a numeric (either integer or decimal) textbox.

```
[NumericSetting("Numeric Value", "This is a numeric setting", false)]
public string NumericValueSetting { get { return Setting("NumericValue", "", false); } }
```

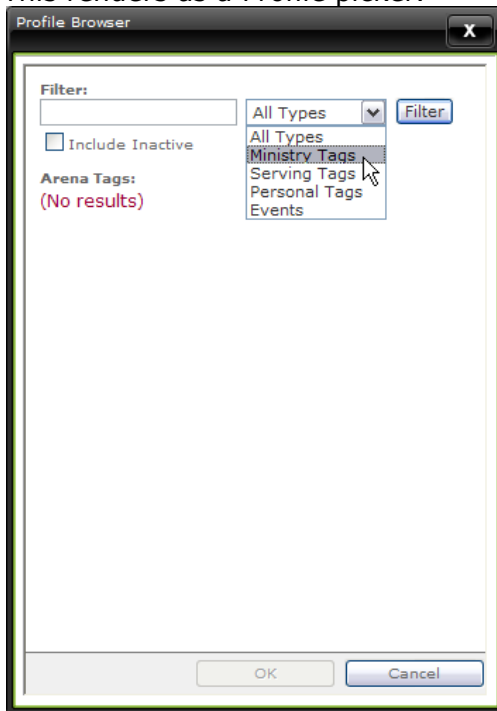
Page

This renders as a page picker. It accepts a default page ID if desired.

```
[PageSetting("Page ID", "This represents a page id that points to somewhere", false, 7)]
public string PageIDSetting { get { return Setting("PageID", "7", false); } }
```

Tag

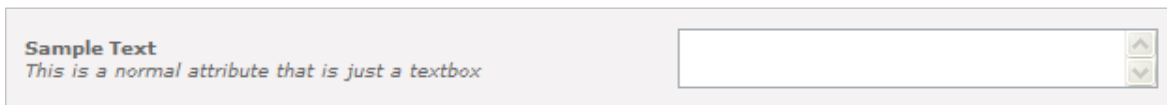
This renders as a Profile picker.



```
[TagSetting("Default Tag", "This links to a tag/profile", false)]
public string TagSetting { get { return Setting("Tag", "", false); } }
```

Text

This setting will render as a Textbox.



```
[TextSetting("Sample Text", "This is a normal attribute that is just a textbox", false)]
public string SampleSetting { get { return Setting("Sample", "", false); } }
```

List From SQL

This lets you specify a query to create a drop down of possible values. The query should be written with the "value" part as the first column and the "text" part as the second column. This base class has been updated to allow rendering of a ListBox for Multiple selections. To use, set the ListSelectionMode property of the attribute. The value will be a comma delimited list of your values.

```
[ListFromSqlSetting("Default Type ID", "The default type ID of this small group cluster.", true, "", "select
cluster_type_id, type_name from smgp_cluster_type order by type_name")]
public string DefaultTypeSetting { get { return Setting ("DefaultType", "", true); } }
```


Cluster Category (inheriting from ListFromSqlSetting)

This is a drop down of values from smgp_category. This was created by inheriting from the ListFromSqlSetting class. Developers can inherit from this class to create reusable setting types.

```
public class ClusterCategorySettingAttribute : ListFromSqlSettingAttribute
{
    private const string sql = "select category_id, category_name from smgp_category order by category_id";

    public ClusterCategorySettingAttribute(string name, string description, bool required) : base(name, description,
        required, "", sql)
    {
    }
}

[ClusterCategorySetting("Category ID", "The ID of the category of this small group cluster.", true)]
public string CategorySetting { get { return Setting("Category", "", true); } }
```

Report Setting

This setting allows you to pick a report or a report folder from reporting services. For example, in events, you specify a folder on the reporting server that holds ETicket reports. The above setting creates a picker UI instead of having to manually type in the path..

```
[ReportSetting("ETickets Folder", "The folder that contains ETicket Reports.", false, SelectionMode.Folders,
"/Arena/ETickets")]
public string ETicketReportURLSetting { get { return Setting("ETicketReportURL", "/Arena/ETickets", false); } }
```

The SelectionMode enum has the following options:

- Folders (only folders are selectable)
- Reports (only reports are selectable)
- All (both folders and reports are selectable)

Smart Page Settings

This new module setting type simplifies page relationships in module settings for certain scenarios such as a list page that goes to another detail page. Instead of requiring the user to explicitly set a page id for the detail, you can give a “hint” that the detail module should be below, beside, or above (in terms of hierarchy). So, if a setting isn’t explicitly set, the system will automatically find the related module.

```
[SmartPageSetting("Cluster Type Page", "The page used to show cluster types associated with this category.",
"UserControls/SmallGroup/ClusterTypeList.ascx", RelatedModuleLocation.Beneath)]
public string ClusterTypePageIDSetting
{
    get { return _clusterTypePageID; }
    set { _clusterTypePageID = value; }
}
```

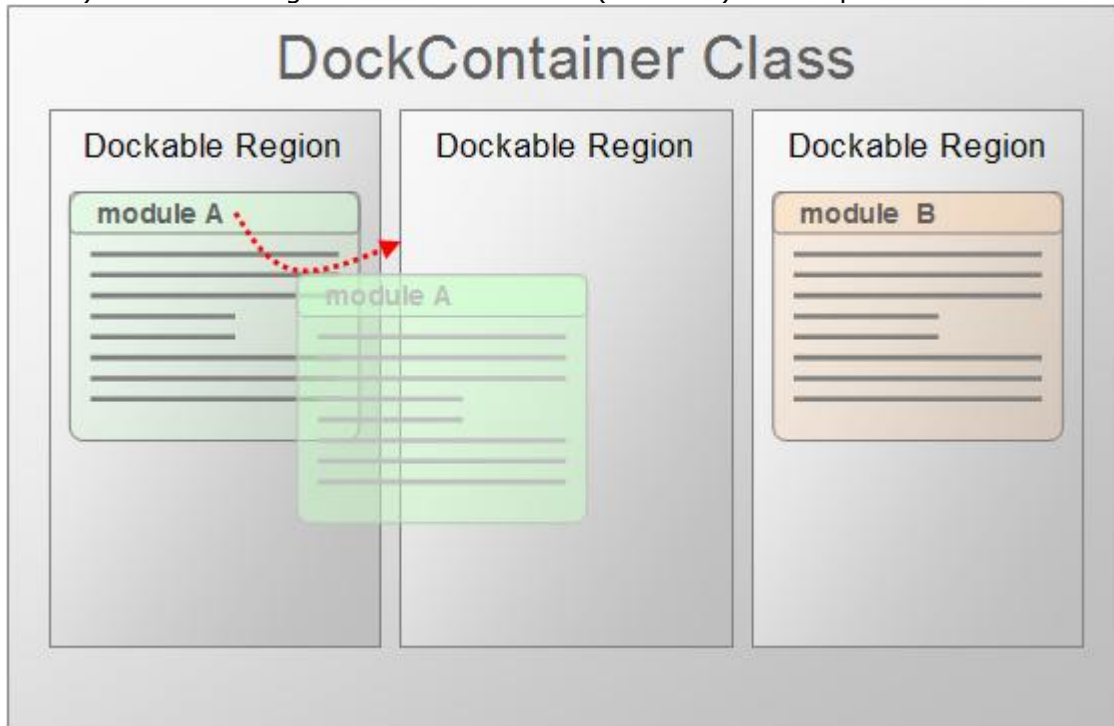
Note: the SmartPageSetting requires that you use a private property for get and set instead of calling Setting(...) in the get.

The “My Portal” Framework

The 2008.1 release of Arena introduces some new capabilities that developers can take advantage of to provide a richer “portal” experience for users that are similar to iGoogle, my.live.com or my.yahoo.com. This new framework functionality exists in the form of DockContainers, Dockable modules, and personalized module settings.

Note: If you are a new developer or new to Arena development, you might consider skipping this section until you are comfortable with the Arena framework.

Developers can create custom DockContainers (similar in nature to ASP.NET WebPart Zones) that define regions in which content (modules) can be placed or ordered.



See the MyPortal code examples for a reference implementation.

DockContainers

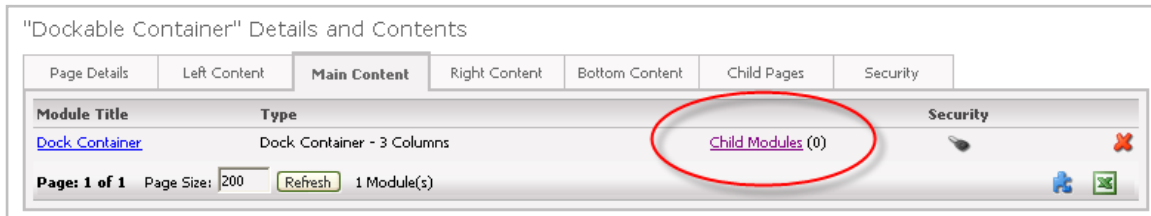
A DockContainer is a simple class that extends the **PortalControlDockContainer** and exposes one or more public properties that act as the dockable regions for dockable modules.

There are several docking style/behavior properties that can be set inside the overridden OnInit() method. They are listed in the table below.

Property	Description
DockContentCssClass	CSS class name that controls the style of docked content area (the area surrounding a dockable module instance).
DockHeaderCssClass	CSS class name that controls the style of the docked content's heading or title.

Property	Description
DockingStyle	This controls the appearance of the docking region when a dockable module is being dragged into its new place. Possible values <ul style="list-style-type: none"> • SolidOutline (default) • DashedOutline • TransparentRectangle • Shadow • Original • None
DraggingStyle	This controls the appearance of the dockable module as it is being dragged into its new position. Possible values <ul style="list-style-type: none"> • Original (default) • SolidOutline • DashedOutline • TransparentRectangle • Shadow • GhostCopy • None

When completed, the DockContainer is also an Arena module which is added to the Arena module list and set with the "Allows Child Modules" flag as true. Once an instance of a DockContainer module has been added to a page, dockable child modules can then be added to it.

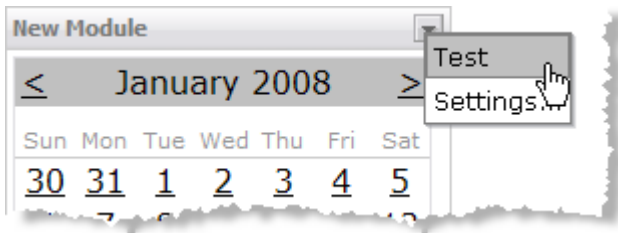


Dockable Modules

In their simplest form, Dockable modules are basically any Arena module that has been defined as a child module of a DockContainer module. They can be defined to appear (by default) to users in any of the DockContainer's dockable regions or they can be added to the list of available modules from which a user can choose to display on the page. Additionally these modules can be flagged as "mandatory" or "movable" by the administrator thereby preventing users from removing them from the page or moving them around the page, respectively.

If needed custom menu items can be added to the Dockable Modules' pull-down menu by implementing the **ICustomMenuControl** interface.

```
public partial class YOURMODULE : PortalControl, ICustomMenuControl
{
    ...
    // Implement the ICustomMenuControl interface if this module needs custom items to be displayed
    // in the pull down menu that is auto-generated with the dockable window.
    public IEnumerable<CustomMenuEntry> GetCustomMenuEntries()
    {
        // Only javascript commands are valid for this menu, so you can put a hidden button on the
        // page and get the javascript need to fire the button as in the example below.
        List<CustomMenuEntry> items = new List<CustomMenuEntry>();
        items.Add(new CustomMenuEntry
        (
            "Test",
            Page.ClientScript.GetPostBackEventReference(btnToCallFromScript, "")
        ));
        return items;
    }
}
```



Lastly, a Dockable Module can have Personalized Module Settings can be overridden by individual users if needed.

Personalized Module Settings

Any Module Setting that has the "[**PersonSpecific**]" attribute declaration will be treated by Arena as a module setting whose value can be overridden by a user.

```
[PersonSpecific]
[BooleanSetting("Show Services", "Flag indicating if services should be shown on the calendar.", false, true)]
public string ShowServicesSetting { get { return Setting("ShowServices", "false", false); } }
```

This kind of functionality is vital for creating highly configurable modules and for providing users with rich personalization capabilities.

Context Sensitive Content

A new feature in the 2008.2 release is the ability for a custom module to obtain the current "context" from the Arena Framework. This means that a module has the potential to *provide different content based on the current context of a page*.

Note: If you are a new developer or new to Arena development, you might consider skipping this section until you are comfortable with the Arena framework.

Consider this example illustration. A new *Advanced Html Text* module is on a page which also has the *Small Group Detail* module. Assuming the Small Group Detail module publishes its currently selected group GUID, the Advanced Html Text module can obtain this GUID (at runtime) can provide unique content that is specific to that group.

In order to achieve this with Arena 2007 each group would need to have their own page with their own Advanced Html Text module. The power that this loose coupling provides continues to grow as you consider additional situations. In the previous example the Advanced Html Text module is not tied to a group. The published context could have been from a profile/tag, in which case the Advanced Html Text module would have provided content for the profile/tag.

Imagine a Blog module using this same mechanism. That module could be used to provide a blog for **any entity** that publishes its context to the framework. The possibilities are endless.

Obtaining the Content Context

To obtain the current page's context you simply need to access the `ArenaContext.Current.ContentContext` property as shown in this example:

```
public void Page_Load(object sender, EventArgs e)
{
    // ArenaContext.Current is a static property, so it is available anywhere in the framework,
    // not just on PortalControls.
    if (ArenaContext.Current.ContentContext != null)
    {
        // Get content based on ArenaContext.Current.ContentContext.Guid
    }
    else
    {
        // Get normal content.
    }
}
```

Publishing a Custom Content Context

In order for a new custom class to publish to the ContentContext it must implement the `IKnownObjectTypeInterface` as shown in this example of an Assignment object.

```
#region IKnownObjectType Members

public Guid Guid
{
    get
    {
        return _guid;
    }
}

public string Table
{
    get { return "asn_assignment"; }
}

public object Instantiate(ArenaContext context)
{
    if (!string.IsNullOrEmpty(context.QueryString["assignmentid"]))
    {
        Assignment assignment = new Assignment(Convert.ToInt32(context.QueryString["assignmentid"]));
        if (!assignment.Allowed(OperationType.View, context.User, context.Person))
            assignment = null;
        return assignment;
    }
    return null;
}

#endregion
```

Notice how the object is instantiated based on the ArenaContext QueryString (it could be based off of any info available in the ArenaContext object) and how security is checked before returning the item. Null should be returned if the user does not have security rights to view the item.

The class must also be registered as a known type by adding to the "Custom Content Context Classes" lookup type as shown in this example:

Custom Content Context Classes

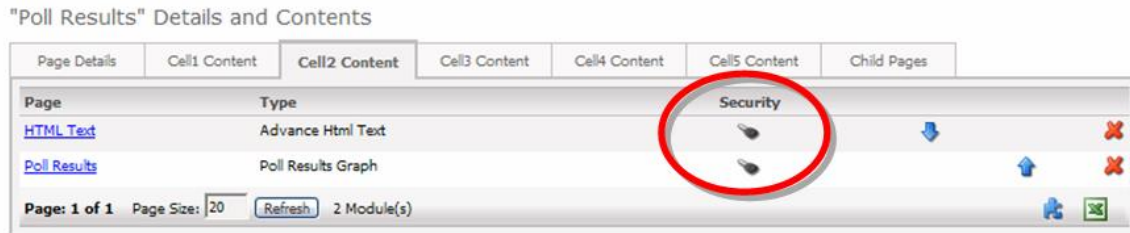
Classes that implement the `IKnownObjectType` interface and should be used as content context providers. In the 'Class Name' field, enter the fully qualified class name, e.g. 'Arena.Custom.COTCMem.MyClass'. In the 'Assembly Name' field, enter the assembly name, e.g., 'Arena.Custom.COTCMem'.

ID	Value	Class Name	Assembly Name
9731	Assignments	Arena.Assignments.Assignment	Arena.Framework

Page: 1 of 1 Page Size: 20 Refresh 1 Lookup Value(s)

Security and Access Control

The Arena framework provides the mechanism for assigning access roles to a module when adding them to a page as shown here.



The module developer is responsible to incorporate the necessary logic in the user control to determine if the current user has access for the current module instance.

This is achieved using the `CurrentModule.Permissions.Allowed` method as shown in this example:

```
bool_editEnabled = CurrentModule.Permissions.Allowed(OperationType.Edit, CurrentUser);
```

Use an appropriate operation type value from the table below:

Operation Types	Description
Approve	Ability to set the approve bit on an item.
Edit	Ability to edit the details of an item. Having edit does not imply view access.
Edit_Modules	
Edit_Notes	
Edit_People	
Edit_Registration	
Edit_Security	Ability to set role and person security on the item.
View	Ability to view the details of the item.

Also note that when your custom module accesses non-custom business objects/entities, the logic in the module must adhere to security policy settings for those entities. For example, when accessing Person Attribute entities and data, the `core_attribute_access` must be checked before allowing view and edit access to the data.

Developer Guidelines

Policies

The following policies are intended to set boundaries that will allow you to successfully integrate your custom modules into the Arena platform. Failure to adhere to these policies can cause your modules or changes to be destroyed during a future Arena update.

- Do NOT modify existing user controls (.ascx files) on the web site. You may add new .ascx and .ascx.cs files to the /UserControls/Custom/<ORGID> directory
- Do NOT modify existing pages (.aspx) files on the web site. You may add new .aspx and .aspx.cs files to the /<ORGID> directory.
- Do NOT modify existing style sheets. You may add new .css files to the /css/Custom/<ORGID>/ directory.
- Do NOT modify existing images. You may add new images to the /images/custom/<ORGID> directory.
- Do NOT modify the schema of existing database tables. Any schema changes to the base tables will be undone during an upgrade. You may create new tables following the naming conventions defined in the next section.
- Do NOT modify existing database stored procedures. Any changes to the base stored procedures will be overwritten during an upgrade. You may create new stored procedures using the naming conventions defined in the next section.
- Do NOT modify existing database views. Any changes to the base views will be undone during an upgrade. You may create new views using the naming conventions defined in the next section.
- DO create custom classes, tables, stored procedures, etc. using the naming conventions defined in the next section. This ensures that no naming conflicts will arise during an upgrade.

Naming Conventions

These naming conventions are critical to ensure that your custom modules do not interfere with any other installed Arena custom modules. Additionally, failure to adhere to these naming conventions can prevent successful future upgrades. If you fail to follow the naming conventions outlined below in custom development, your module will be rejected from the community site and Arena Support will not be responsible for any upgrade problems you encounter.

Database Tables

Custom database tables should be named as follows:

`cust_<ORGID>_<Project-ModuleName>_<ClassName>`

Example: cust_cotcmem_library_item

Database Stored Procedures

Custom stored procedures should use either "get" "save" or "del" as follows:

`cust_<ORGID>_<Project-ModuleName>_sp_get_<descriptivename>`

`cust_<ORGID>_<Project-ModuleName>_sp_save_<descriptivename>`

`cust_<ORGID>_<Project-ModuleName>_sp_del_<descriptivename>`

Example: cust_cotcmem_library_sp_save_libraryLoan

Example: cust_cotcmem_library_sp_del_libraryLoan

Example: cust_cotcmem_library_sp_get_libraryLoanByID

*Example: cust_cotcmem_library_sp_get_libraryLoanByIDAndStatus ***

** For multiple filtering criteria use "And" to concatenate filter items.

Database Views

Custom views should be named as follows:

`cust_<ORGID>_<Project-ModuleName>_v_<descriptivename>`

Example: cust_cotcmem_library_v_library_outstanding_loans

User Controls

Custom user controls (modules) should be placed in Arena in the UserControls/Custom/<ORGID>/<Project-ModuleName>/ directory. The namespace of the code behind file should be as follows:

`ArenaWeb.UserControls.Custom.<ORGID>.<Project-ModuleName>`

Example: ArenaWeb.UserControls.Custom.Cotcmem.Library

Note: You may create any number of folders underneath your UserControls/Custom/<ORGID>/<Project-ModuleName>/ directory.

Custom Classes

Custom classes must be in external projects in Visual Studio 2005. The namespace of the custom classes should be as follows:

For data layer classes:

`Arena.Custom.<ORGID>.<Project-ModuleName>.DataLayer`

Example: Arena.Custom.Cotcmem.Library.DataLayer

For entity classes:

Arena.Custom.<ORGID>.<Project-ModuleName>.Entity

Example: Arena.Custom.Cotcmem.Library.Entity

Note: You may create deeper namespaces underneath your Arena.Custom.<ORGID> namespace as needed.

New Pages (.aspx and .aspx.cs)

You should avoid adding new pages to Arena if possible. However, if it is necessary to create a new page, then you must follow the guidelines below:

New .aspx pages should be placed in the /<ORGID> directory.

Example: /Cotcmem/MyNewPage.aspx

The namespace of your pages should be as follows:

ArenaWeb.<ORGID>.<Project-ModuleName>

Example: ArenaWeb.Cotcmem.Library

Web Services (.asmx)

Your custom .asmx files should be placed in the WebServices/<ORGID> directory.

Example: /WebServices/Cotcmem/Outlook.asmx

The namespace of your pages should be as follows:

ArenaWebService.<ORGID>.<Project-ModuleName>

Example: ArenaWebService.Cotcmem.Outlook

Custom Organization Settings

Your custom Organization Settings should be named as follows:

<ORGID>.<Setting-Name>

Example: Cotcmem.PrintLabelProvider

Source Control & Comment Blocks

The use of an actual source control system such as SourceGear Vault, MS Visual Source Safe or a similar system is highly recommended even if you are in a single developer environment. All custom code artifacts should include the comment block as shown below that allows for [keyword expansion](#) by the source control system.

```

/*****
* Description:      <INSERTED BY CODE GENERATOR>
* Created By:      <INSERTED BY CODE GENERATOR>
* Date Created:    <INSERTED BY CODE GENERATOR>
*
* $Workfile: $
* $Revision: $
* $Header: $
*
* $Log: $
*****/

```

All public methods in your custom classes must be commented using standard `///` comment structure as shown in the following Example:

```

/// <summary>
/// Saves the library item to the database.
/// </summary>
/// <param name="userId">ID of person saving the record</param>
public void Save( string userId )
{
    ...
}

```

Always add comments above any changes you make to your released code so that other developers can quickly identify the reason for the change. Your comments should include lengthy details about the change as well as a change request, feature, or bug ID when applicable. These details only need to be provided for the first comment. Subsequent comments relating to the same change only need the bug ID and comments relating to the code directly below.

```

// BUG #376: No longer call the SaveMembers() by passing a
// person ID. Doing this was preventing the underlying class from
// properly determining the correct person at runtime via the
// context.
team.SaveMembers();

```

Globalization and Localization

Since Arena is being used by several cultures around the world, it is good practice to keep some things in mind when designing your modules; namely date/time and currency handling. Below are a few things you should be aware of:

- Always use a DateTime formatter that is culture aware such as any of the converters that come with .NET (ToShortDateString(), ToLongDateString(), etc.), a culture aware ToString format string (i.e. ToString("d") not ToString("MM/dd/yy")), or one of the **Arena.Utility** DateTime extenders below:
 - **IsEmptyDate()** returns whether the date is empty, meaning it equals either the max or min dates (.NET min/max, SQL min/max or defined as 1/1/1900 or 12/31/2099).
 - **ToShortDateString(hideEmptyDate)** returns "" if the date is empty, otherwise returns base.ToShortDateString()
 - **ToShortDateTimeString()** returns the short date and time strings. This is different than ToString() as it does not include the seconds for the time.
 - **ToShortDateTimeString(hideEmptyDate)** returns "" if the date is empty, otherwise returns ToShortDateTimeString()
 - **ToMonthDayString()** returns just the month and day in the correct format. It will remove the year and separator for display on the person selection control
- A new BoundColumn, DateTimeColumn, has been added to the Arena DataGrid which will hide the date if it is empty. Use the DataFormatString to specify the format (for date and time use "{0:d} {0:t}").
- The Arena DateTextBox is now culture aware and there are properties for validation and requiring a value.

For more information about ASP.NET Globalization and Localization see <http://msdn.microsoft.com/en-us/library/c6zyy3s9.aspx>.

Module Creation Example

This section walks through the creation of the sample Library Book Management (sample) Module to illustrate the steps required to develop a new module. Make sure you have set up your environment and have opened the Arena solution as described in section Using Web PI to install Visual Web Developer Express. You may also wish to download this sample Module from the community site. David Turner has also created an [excellent video demo of creating a module](http://www.david-turner.net/post/2007/11/Adding-a-Custom-Module-to-Arena.aspx) at <http://www.david-turner.net/post/2007/11/Adding-a-Custom-Module-to-Arena.aspx>

Creating Custom Classes (Optional)

If you need custom classes for your module (e.g., a new data layer class or an entity class), then you will need to create a new Visual Studio 2005 project and reference this project from the web site.

1) Right click on the Solution, and click on "Add New Project"

Select "Visual C#" then "Class Library".

Name the project "Arena.Custom.<ORGID>.<Project-ModuleName>", where <ORGID> is your assigned Arena organization id and <Project-ModuleName> is the logical name of your Module.

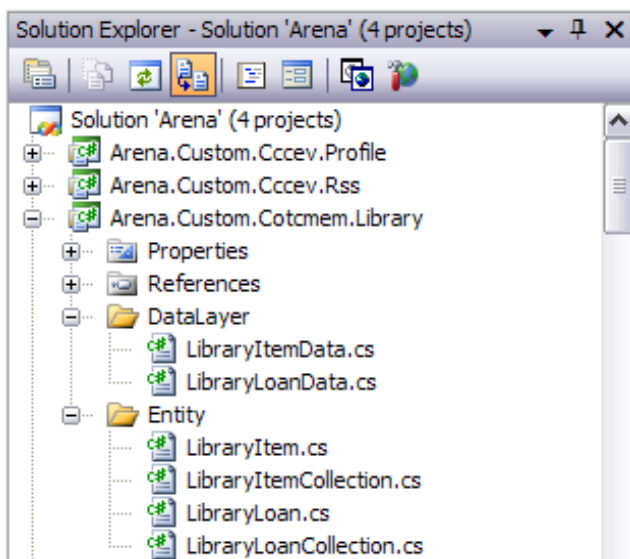
Right click on your web site in the solution explorer and select "Add Reference..."

Go to the "Projects" tab, select your project, and click "OK"

Within the custom project, add a reference to Arena.Framework (this is in your Arena \bin folder).

Note: It is highly recommended that you use a code generator and an Arena code template to generate the bulk of your code. Doing this ensures that your code follows the established patterns, reduces the likelihood of bugs, and makes your code easier to maintain. See Code Generation section for Arena code generation templates.

For our example we've created the following classes under the Arena.Custom.Cotcmem.Library project as shown in here:



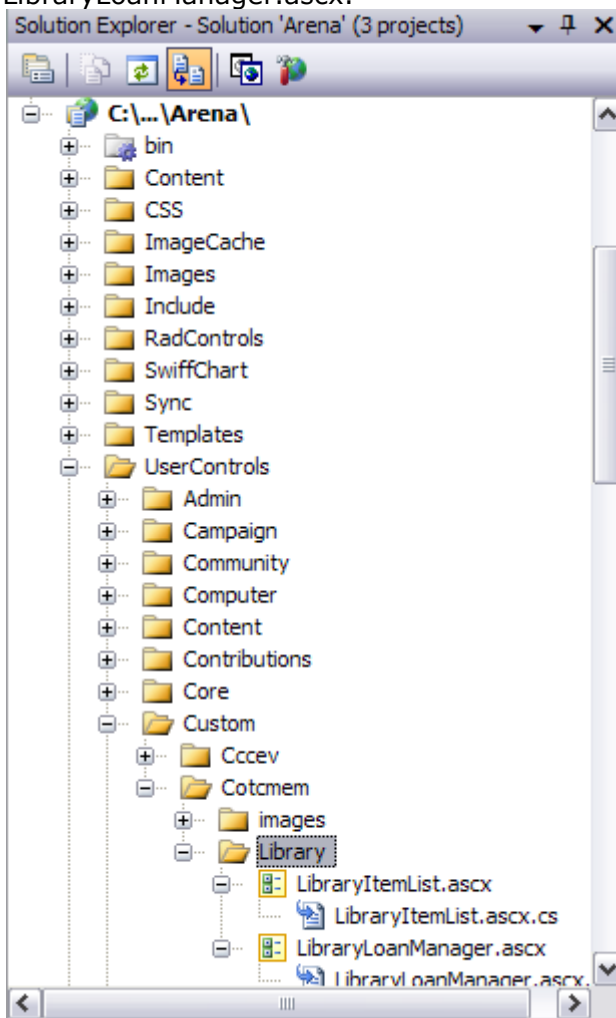
With the release of Arena 2008.2, the .NET 3.5 framework is now required. This means you can take advantage of features such as LINQ (<http://www.asp.net/LEARN/linq-videos/>) in your custom data layer classes.

Creating Custom User Controls

Next, we'll walk through the process of creating User Controls for your module by example. In this example we're using the "Library Book Management (sample)" module available on the community site.

Open your Arena website with Visual Studio 2005 and create a folder called `"/UserControls/Custom/<ORGID>/<Project-ModuleName>/"` in the root of your Arena folder.

Inside this new folder we've created two user controls, `LibraryItemList.ascx` and `LibraryLoanManager.ascx`.



Note: Images, stylesheets and other artifacts should be placed in the location as specified in the Policies section.

Adding the New Module to Arena

Under the Administration menu select the Modules item and then click the add module icon. Fill out the form ensuring you put the correct path to your user control in the URL field. Press update when finished. Do this for each UserControl of your Module.

Modules
ArenaChMS Modules

home > administration > modules

Name

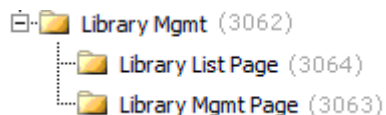
URL

Description

Field Hints

Create New Pages

If necessary, create any new pages you need for your new Module. See the Arena Administrator Manual for help creating new pages. For our sample module we've created the following three pages:

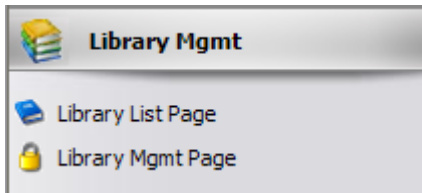


If desired, you can include your custom icons for your Modules by editing the Navigation Bar Icon and Icon Hover settings on the appropriate pages as shown:

Navigation Bar Icon
Image to display for this page when it is listed in the Navigation Bar control (default = 'top_folder.gif').

Navigation Bar Icon Hover
Image to display when mouse hovers over this page when it is listed in the Navigation Bar control (default = 'top_folder_sel.gif').

Provided the “Display In Nav” setting is checked for each page, your Module will appear in the navigation menu as shown here:



Set the Page Icon and Show Heading attributes in order to see the standard heading and a representative icon on the page.

Page Icon <small>Optional Image URL to use as the Icon for the page. Default is '~/images/#.jpg' where # is the current Page ID.</small>	<input type="text"/>
Show Heading <small>Optional Flag indicating if the Page Heading should be displayed on this page (True*/False).</small>	<input type="text" value="true"/>

Guidelines for Creating Top Level Navigation


The following guidelines should be used in order to manage the number of items in the top level of the left navigation bar:

- When the functionality is vastly different than everything else
- When only a subset of users will see it (i.e. Sports)
- When there will be more than 3-4 sub links (sub navigation pages)

Create Module Instances on a Page

Add your custom user controls (modules) to your pages as needed. See the Arena Administrator Manual for help with adding modules to pages.

For our sample, we will add the Library Book List module to the Library List Page and then add the Library Loan Manager module to the Library Mgmt Page as shown below.

Under the Administration menu select the Pages item and then navigate to the page in your page hierarchy. Select the Main Content tab and press the add module () icon. Next select the empty “New Module” item that was created.

"Library List Page" Details and Contents

Page Details	Left Content	Main Content	Right Content	Bottom Content	Child Pages				
Type of Module Library Book List <small>List of library books.</small>									
Title Library Book List									
Show Title <input checked="" type="checkbox"/>									
Content Frame Main									
Settings <table border="1"> <thead> <tr> <th>Setting Name</th> <th>Setting Value</th> </tr> </thead> <tbody> <tr> <td colspan="2">This module doesn't have any module settings</td> </tr> </tbody> </table>						Setting Name	Setting Value	This module doesn't have any module settings	
Setting Name	Setting Value								
This module doesn't have any module settings									
Details <div> <div></div> <div></div> </div>									
<input type="button" value="Update"/> <input type="button" value="Cancel"/>									

The Library Loan Manager needs to reference the page where the Library Book List module is located. This is accomplished by selecting the Library List Page as shown:

"Library Mgmt Page" Details and Contents

Page Details	Left Content	Main Content	Right Content	Bottom Content	Child Pages						
Type of Module Library Loan Manager <small>Used to manage library book loans</small>											
Title Library Loan Manager											
Show Title <input checked="" type="checkbox"/>											
Content Frame Main											
Settings <table border="1"> <thead> <tr> <th>Setting Name</th> <th>Setting Value</th> </tr> </thead> <tbody> <tr> <td> Book List Page (required) <small>The page used to list books.</small> </td> <td>Library List Page <input data-bbox="1187 1339 1219 1367" type="button" value="..."/></td> </tr> <tr> <td> Default Checkout Duration in Days <small>The default number of days a book is allowed to be checked out from the library before being due.</small> </td> <td>14 <input data-bbox="1187 1409 1219 1436" type="button" value="..."/></td> </tr> </tbody> </table>						Setting Name	Setting Value	Book List Page (required) <small>The page used to list books.</small>	Library List Page <input data-bbox="1187 1339 1219 1367" type="button" value="..."/>	Default Checkout Duration in Days <small>The default number of days a book is allowed to be checked out from the library before being due.</small>	14 <input data-bbox="1187 1409 1219 1436" type="button" value="..."/>
Setting Name	Setting Value										
Book List Page (required) <small>The page used to list books.</small>	Library List Page <input data-bbox="1187 1339 1219 1367" type="button" value="..."/>										
Default Checkout Duration in Days <small>The default number of days a book is allowed to be checked out from the library before being due.</small>	14 <input data-bbox="1187 1409 1219 1436" type="button" value="..."/>										
Details <div> <div></div> <div></div> </div>											
<input type="button" value="Update"/> <input type="button" value="Cancel"/>											

Press update and then select the Refresh Cache option from the Administration menu.

Module in Action

When you navigate to the Library Book List page on your site you will see the content created by the new module:

Title	ISBN	Author	
20,000 Leagues under the Sea	3643346345	Verne, Jules	✗
A Christmas Carol	7563454455	Dickens, Charles	✗
Aesop's Fables - Volume I	5454564554	Aesop	✗
Alice's Adventures in Wonderland	3435465433	Carroll, Lewis	✗
All's Well That Ends Well	4534645343	Shakespeare, William	✗
Around the World in 80 Days	5664353343	Verne, Jules	✗
David Copperfield	6564445544	Dickens, Charles	✗
Gulliver's Travels	2198545498	Swift, Jonathan	✗
Moby Dick	9451599459	Melville, Herman	✗
Oliver Twist	6546455544	Dickens, Charles	✗
Paradise Lost	1321984519	Milton, John	✗
Pride and Prejudice	3433452278	Austen, Jane	✗
The Adventures of Huckleberry Finn	4525234534	Clemens, Samuel Langhorne	✗
The Adventures of Tom Sawyer	2342343434	Clemens, Samuel Langhorne	✗
The Call of the Wild	6465334344	London, Jack	✗
The Iliad	2319846519	Homer	✗
The Island of Doctor Moreau	7865756655	Wells, H. G.	✗
The Odyssey	1268951984	Homer	✗
The Return of Sherlock Holmes	7546534344	Doyle, Arthur Conan, Sir	✗
The Time Machine	9876544455	Wells, H. G.	✗
The Waste Land	3219885198	Eliot, T. S.	✗
Ulysses	6564544564	Joyce, James	✗
War and Peace	9812198138	Tolstoy, Leo Nikoleyevich	✗

Page: 1 of 1 Page Size: 200 23 Book(s)

Navigate to the Library Mgmt Page to see the Library Loan Manager in action:

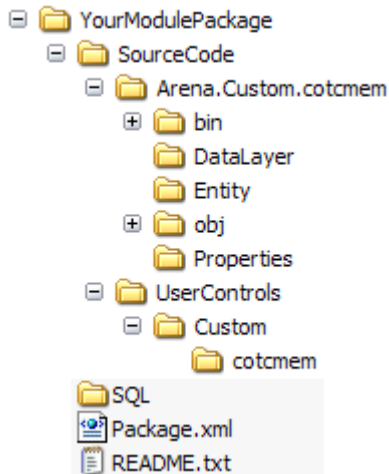
Actions

Packaging Your Module for the Community

If you intend on sharing your Module with the Community, you'll need to package it so that others can easily install and use it. Although not required, you are also encouraged to share your source code for the benefit of other community developers. The instructions illustrate how to create a package for your Module.

Zip Folder Structure

To package your Module you will need to create a zip file with the following folder structure:



1. Create a folder with the logical name of your Project/Module.
2. Inside, create the "SourceCode" and "SQL" folders.
3. Put the source code of any custom classes and UserControls under the SourceCode folder.
4. Put any required SQL scripts under the SQL folder.
5. Put the exported package XML file (see Module Export below), and a README.txt file containing any instructions you want to share regarding your Module into the root folder.
6. Zip up the package (starting from the base folder) and post it to the Community site.

Note: In order to simplify the installation of your Module, it is planned that a future version of Arena's Import function will allow others to easily run the SQL scripts you provide.

Module Export

This section gives a brief summary of how to export your module from Arena. For the full details read the "Arena Import/Export Instructions" (see 0 References section) found on the Community site.

1. Under Page Administration, use the export button on the page containing your module as shown here:

"Library Mgmt" Details and Contents

Page Details	Left Content	Main Content	Right Content	Bottom Content	Child Pages
Template ArenaChMS					
Parent Page Home					
Page Name Library Mgmt					
Display In Nav Yes					
Require SSL No					
Description This is the sample library mgmt folder					
<input type="button" value="Edit Details"/> <input type="button" value="Export..."/>					

2. Attach any assemblies for your custom classes (if any), CSS files, etc. in the Attach Additional Files area. The export utility automatically attaches module user control files (.ascx and .ascx.cx files and images used inside of these modules), so you do not need to manually attach these files.

Export - Windows Internet Explorer

Page to Export: Library Mgmt

NOTE: The template frame names (e.g., "Left Content", "Main Content", "Right Content") used on this page will be exported. The same frame names must exist on the target template when importing this page.

Special Instructions: ?

☒ Export child pages ?

Attach Additional Files: ?

bin/Arena.Custom.Cotcmem.Library.dll	...
	...
	...
	...
	...
	...
	...
	...

3. Press export and save the package XML file to the root folder where you are preparing to zip the Module.

Arena UI Toolbox

There are many tools in the Arena toolbox that will ease your custom development in the UI layer. Although not intended to replace the API documentation, the items listed in this section are some of the major parts you should be familiar with and should utilize to create a standard, consistent user interface.

Arena:BooleanImageColumn

This control creates a bound column that shows an image when true and nothing when false. Image is defined by ImagePath property; it defaults to the green checkbox.



Arena:SelectColumn

This control creates a check box column for use in selecting DataGridItems. You must have a DataKeyField set for your datagrid(eg. person_id). There are no other settings for this column. Just having one of these columns will give your users the ability to limit the grids functions (Excel, Word Merge etc) without any extra coding required.

<input type="checkbox"/>	Name	Status
<input type="checkbox"/>	Jesse	Member
<input type="checkbox"/>	Aderhold, Jesse-Wyatt	
<input type="checkbox"/>	Atkinson, Jesse	
<input checked="" type="checkbox"/>	Brown, Jab	Attendee
<input type="checkbox"/>	Brown, JAB	

You can access the **SelectedItems** property of the grid to get a comma delimited list of DataKeyFields. These will be the items that were checked by the user.

```
List<string> items = new List<string>(dgPersons.SelectedItems.Split(','));
foreach (var item in items)
{
    Person p = new Person(Int32.Parse(item));
}
```

Arena:DataGrid

Take advantage of the Arena DataGrid to automatically receive the benefits of a grid with paging, excel export, email hooks, etc. All displays of table data should use the Arena DataGrid.

Title	ISBN	Author	
20,000 Leagues under the Sea	3643346345	Verne, Jules	✖
A Christmas Carol	7563454455	Dickens, Charles	✖
Aesop's Fables - Volume 1	5454564554	Aesop	✖
Alice's Adventures in Wonderland	3435465433	Carroll, Lewis	✖
The Waste Land	3219885198	Eliot, T. S.	✖
Ulysses	6564544564	Joyce, James	✖
War and Peace	9812198138	Tolstoy, Leo Nikoleyevich	✖
Page: 1 of 1 Page Size: 200 <input type="button" value="Refresh"/> 23 Book(s)			

There are a few critical things to know to take advantage of the Arena DataGrid's extra features. The table below lists a few of the properties that you'll want to set. For a complete reference please see the DataGrid overview in the Arena Help file (see References section).

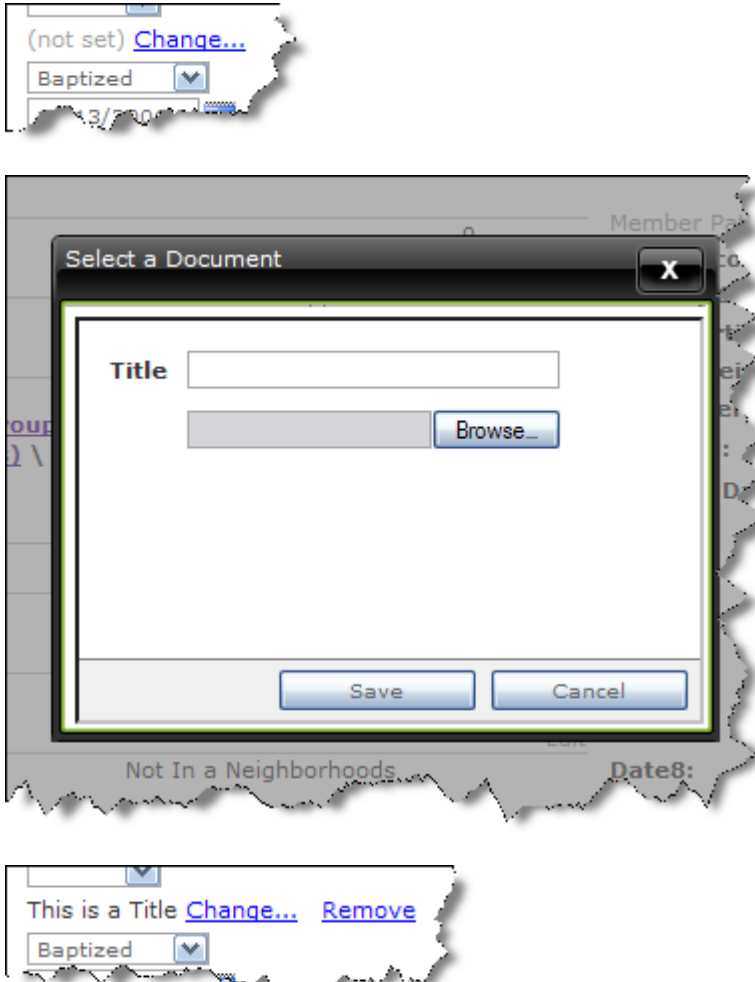
Property	Description
AddEnabled	Set to true to enable the add button (bottom right corner of grid).
AddImageUrl	Use this to define the image of the add button. Example "~/images/add_channel.gif"
AllowSorting	Set to true to enable sorting of the grid.
DeleteEnabled	Set to true to enable delete buttons for each row in the grid.
ExportEnabled	Set to true to enable the Export to Excel feature.
ItemType	Set this to be the logical name of the things that you'll be displaying in your grid. This name will be used in the hover-over of the Add button at the bottom right corner of the grid (provided the AddEnabled property is set to true).
MailEnabled	Set to true to enable the emailing functionality.
MergeEnabled	Set to true to enable the Microsoft Word Mail Merge (to generate address labels) feature.
MoveEnabled	Set to true to enable the re-ordering of items in the list (see SourceTableName, SourceTableKeyColumnName, and SourceTableOrderColumnName)
SourceTableName	The name of the table that will be updated if MoveEnabled is true and a record is moved up or down
SourceTableKeyColumnName	The primary key column name of the table that will be updated if MoveEnabled is true and a record is moved up or down

SourceTableOrderColumnName	The column that contains the sort order for the table that will be updated if MoveEnabled is true and a record is moved up or down
BulkUpdateEnabled	Set to true to enable Person Bulk Update functionality. You must also set the BulkUpdatePageUrl eg. Default.aspx?page=500. This should only be enabled on grids that show people. This can be very useful in conjunction with a SelectColumn.
PersonMergeEnabled	Set to true to enable Person Merge functionality. You must also set the PersonMergePageUrl eg. Default.aspx?page=500. This should only be enabled on grids that show people. This can be very useful in conjunction with a SelectColumn.
Event Handlers	Description
AddItem	Occurs when the Add icon button is clicked to add an item to the <u>DataGrid</u> control.
DeleteCommand	Occurs when the Delete button is clicked for an item in the <u>DataGrid</u> control.
ItemCommand	Occurs when any button is clicked in the <u>DataGrid</u> control.
ItemDataBound	Occurs after an item is data bound to the <u>DataGrid</u> control.
ReBind	Occurs when the DataGrid needs to rebind to the grid's DataSource. NOTE: This method is required for the sorting feature to work.

Note: In order for the footer to appear on the grid, you **must** enable one of the ExportEnabled, AddEnabled, etc footer items.

Document Picker

This control can be used to upload and view a file. It takes a DocumentTypeID (-1 if there is no type ID to be specified). This allows a document to be uploaded and associated with a type, using that type's security. The type determines what fields are available on the modal.



Other Pickers include LookupPicker, PagePicker, ReportPicker and ProfilePicker. They all are very similar in use. If you have any trouble with any of the pickers ask us on the community.

Arena:ImagePopupColumn

Using the Arena:ImagePopupColumn in an Arena:DataGrid, you can easily show a person's photo in a column.



```
<Arena:DataGrid
  ID="dgObject"
  Runat="server"
  DataKeyField="person_id"
  Width="100%">
  <Columns>
    <Arena:ImagePopupColumn
      DataField="photo_guid"
      HeaderText=""
      HeaderStyle-HorizontalAlign="center"
      ItemStyle-HorizontalAlign="center">
    </Arena:ImagePopupColumn>
    <asp:BoundColumn
      HeaderText="Name"
```

The DataField must specify a blob guid. The blob guid refers to the util_blob table's guid field, which uniquely identifies a picture.

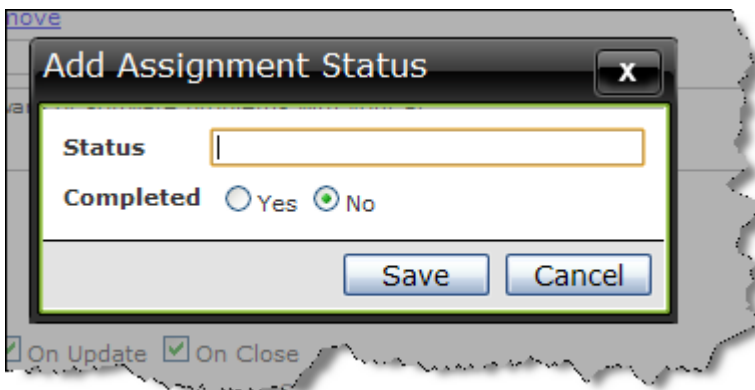
An easy way to write a query to get a person's photo is to join your custom query to the **core_v_personList** view. This view contains the column "photo_guid" so that you don't have to manually join to util blob.

```
SELECT * FROM core_v_personList p
INNER JOIN cust_cotcmem_my_people mp on p.person_id = p.person_id
ORDER BY person_name
```

ModalPopup

This control is used to show content in an Arena modal. It is divided into 2 sections for UI appearance only. You do not have to put your buttons inside the buttons panel. However, doing so will make your modal look like many of the Arena modals. Any content can be in any of the 2 sections.

```
<Arena:ModalPopup ID="mpAddStatus" runat="server" CancelControlID="btnCancelAddStatus"
Title="Add Assignment Status" DefaultFocusControlID="tbStatus">
  <Content DefaultButton="btnAddStatus">
    Content Goes Here
    <asp:TextBox ID="tbStatus" runat="server"></asp:TextBox>
  </Content>
  <Buttons>
    <asp:Button ID="btnAddStatus" runat="server" Text="Save" />
    <asp:Button ID="btnCancelAddStatus" runat="server" Text="Cancel" />
  </Buttons>
</Arena:ModalPopup>
```



You can call `mpStatus.Show()` in code or `mpStatus.show()` in Javascript to show the modal.

ModalPopupIFrame

This modal is used internally for all Arena Picker controls. It is a modal popup with an IFRAME inside. By defining a JSFunctionName, you can pass specific query string variables with each call to function.

For example:

```
JSFunctionName="openPage(pageId,otherId)"  
URL="default.aspx?page=#pageId#&someId=#otherId#"
```

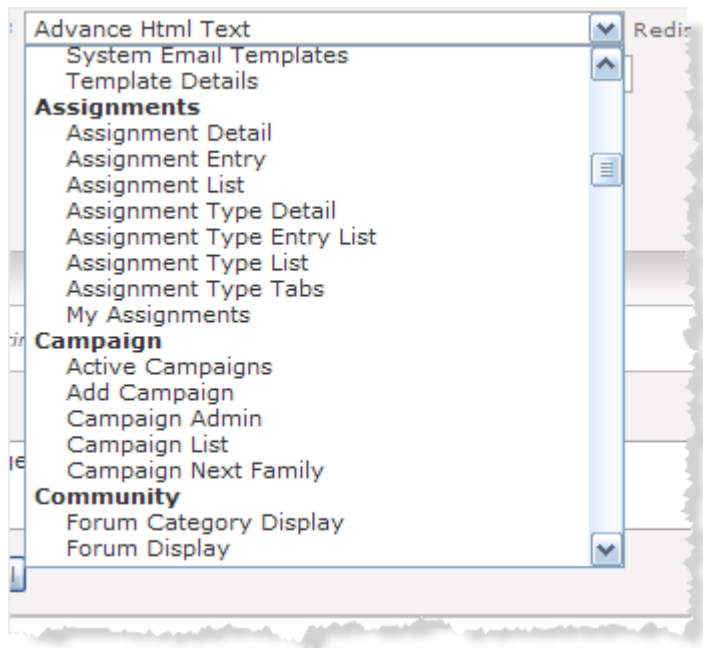
The control will generate the javascript function, openPage, that will open the modal and set the IFRAME to the url, replacing the fields wrapped with ## with the values passed in by the call to the function.

So `` would set the iframeURL to "default.aspx?page=7&someId=55"

It also sends over the modal Ids clientId in the querystring. This is helpful for the developer to interact and possibly close the modal from within the iframe.

OptGroupDropDownList

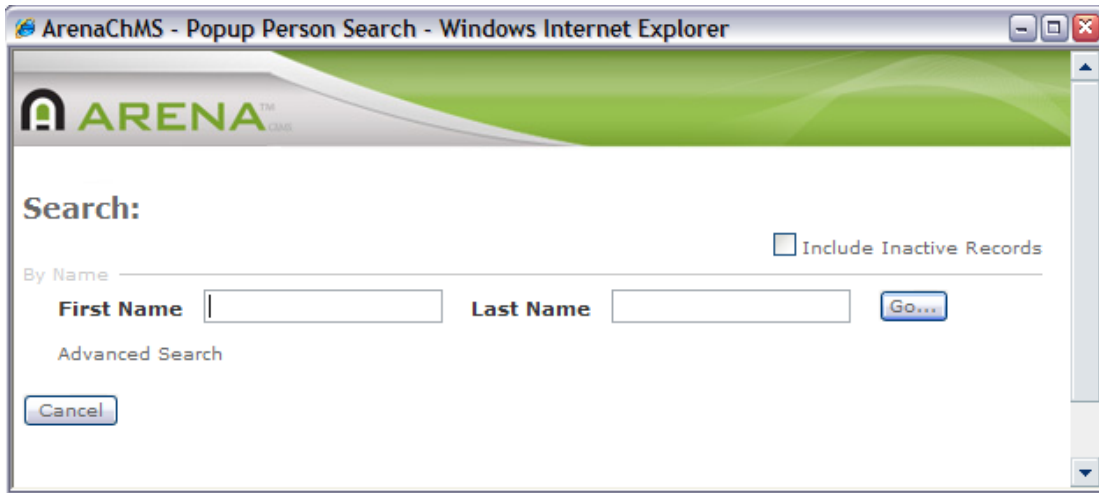
This control works very similar to a dropdown, but you can add grouping.



```
ddl.AddGroup("Group Name");  
ddl.Items.Add(new ListItem("key","value"));
```

Person Search Window

When a custom module needs the capability of searching for and selecting a person, use the 'person search window'.



Add a hidden "refresh" input-button and an input-hidden to your ACSX as well as a button from which the search window is opened:

```
<input type="button" id="btnRefresh" runat="server" causesvalidation="false" onserverclick="btnRefresh_Click"
style="visibility:hidden; display:none;" />
<input type="hidden" id="ihPersonList" runat="server" name="ihPersonList">
...
<asp:LinkButton ID="lbFindPerson" runat="server" OnClientClick="openSearchWindow();return false;" Text="Find..." />
```

The input-button is "pushed" automatically by the search window after a person is selected, and the input-hidden is used to hold a list of personIDs that were selected. The link button is used to initiate the opening of the search window.

In the code, register an openSearchWindow function as shown below and call this RegisterScripts method in the Page_Load:

```
protected void Page_Load(object sender, EventArgs e)
{
    ...
    RegisterScripts();
}

// Script for person search
private void RegisterScripts()
{
    StringBuilder sbScript = new StringBuilder();
    sbScript.Append("\n\n<script language='javascript'>\n");
    sbScript.Append("\tfunction openSearchWindow()\n");
    sbScript.Append("\t{\n");
    sbScript.Append("\t\tvar tPos = window.screenTop + 100;\n");
    sbScript.Append("\t\tvar lPos = window.screenLeft + 100;\n");
    sbScript.AppendFormat("\t\ttdocument.frmMain.ihPersonListID.value = '{0}';\n", ihPersonList.ClientID);
    sbScript.AppendFormat("\t\ttdocument.frmMain.ihRefreshButtonID.value = '{0}';\n", btnRefresh.ClientID);
    sbScript.Append("\t\tvar searchWindow =
window.open('Default.aspx?page=16','Search',height=400,width=600,resizable=yes,scrollbars=yes,toolbar=no,location=
no,directories=no,status=no,menubar=no,top=' + tPos + ',left=' + lPos);\n");
    sbScript.Append("\t\tsearchWindow.focus();\n");
    sbScript.Append("\t}\n");
    sbScript.Append("</script>\n\n");
    Page.ClientScript.RegisterClientScriptBlock(this.GetType(), "OpenSearchWindow", sbScript.ToString());
}
```

Lastly, add the method to handle when the search window pushes the refresh button and add the necessary logic to process the selected personID(s):

```
// Called automatically by the search dialog after a person id has been selected.
// The selected id will be in 'ihPersonList'.
protected void btnRefresh_Click(object sender, EventArgs e)
{
    // The search can possibly return multiple ids based on the module setting.
    if (ihPersonList.Value != "")
    {
        string[] personIds = ihPersonList.Value.Split(',');
        foreach (string id in personIds)
        {
            // Do what ever you need to do with the selected personID(s)
            // YOUR CODE GOES HERE
        }
        ihPersonList.Value = "";
    }
}
```

Arena:DateTextBox

When a custom module needs the user to supply a date value or time value, the Arena:DateTextBox control should be utilized.

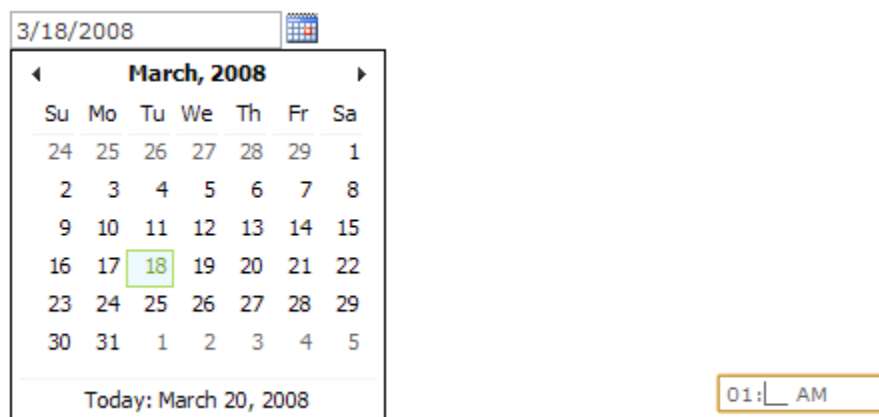


Figure 1 - example using Date format Figure 2 - example using Time format

Add the DateTextBox control into your ACSX as needed, replacing the id value shown below with an id appropriate for your situation:

```
<Arena:DateTextBox id="tbDueDate" CssClass="formItem" Width="100" runat="server" />
```

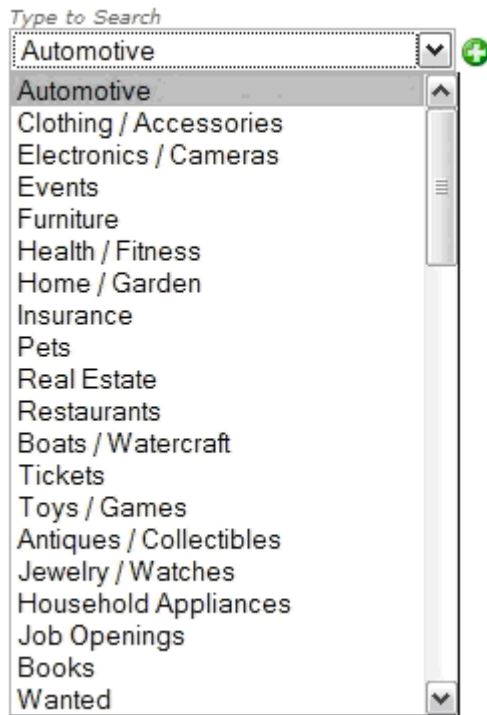
To set or get the date from the textbox simply access the SelectedDate property as shown in this example:

```
DateTime x = tbDueDate.SelectedDate;
tbDueDate.SelectedDate = DateTime.Now;
```

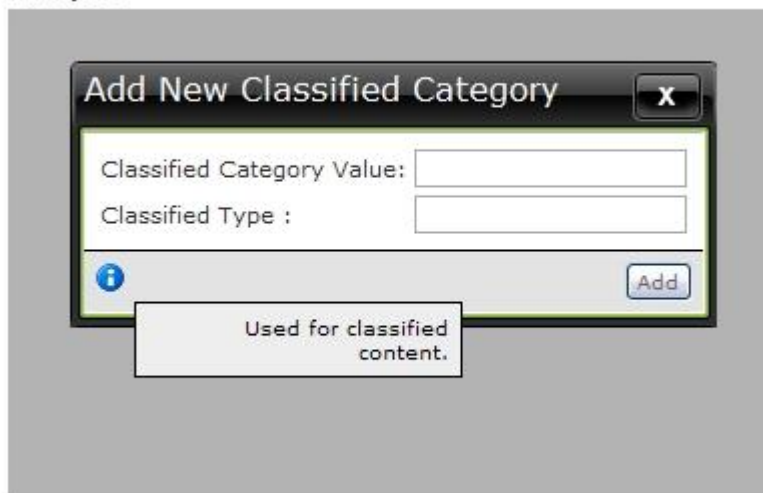
Property	Description
EmptyValueMessage	Message to display when required value is not supplied.
Format	Set to Date or Time to control what kind of value will be accepted. Default is Date.
InvalidValueMessage	Validation error message. Example "Start Time must be a valid time (hh:mm am/pm)!"
MaximumDate	Set to constrain the largest acceptable value. Default is 12/31/2099.
MinimumDate	Set to constrain the smallest acceptable value. Default is 1/1/1900.
Required	Set to true to require the user to supply a value.

Arena:LookupDropDown

This is a new control that populates a drop down list with a collection of lookup values based on a given lookup type. There's an option in the control to show "+" button next to the drop down that allows you to add a new value to the lookup type without having to go to Administration : Lookups to add a new value.



Example:



The Markup:

```
<Arena:LookupDropDown ID="lookupDropDown" runat="server"
SearchEnabled="true" AddEnabled="true" IsValueRequired="true"/>
```

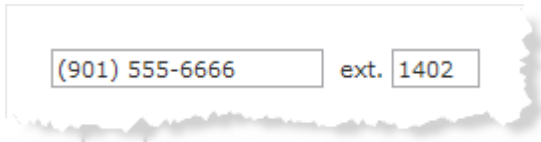
Pretty standard markup, I could have set these properties in code but I set them here to reduce the code.

The Code:

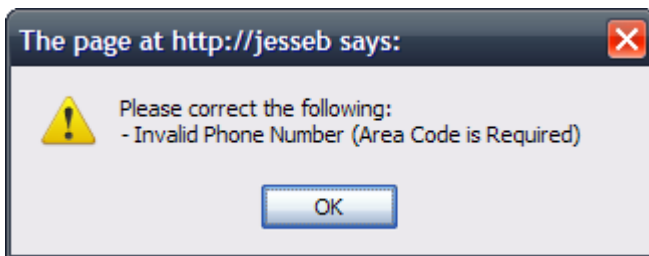
```
protected override void OnLoad(EventArgs e)
{
    lookupDropDown.LookupTypeGUID = SystemLookupType.ClassifiedCategory;
    //lookupDropDown.LookupTypeID = 9;
    if (!Page.IsPostBack)
        lookupDropDown.DataBind();
    base.OnLoad(e);
}
```

Arena:PhoneTextBox

This control is used to help Phone number fields conform to a standard format. It renders a textbox with javascript to help format phone numbers, with and without area codes. It also has the ability to render an extension textbox and label.



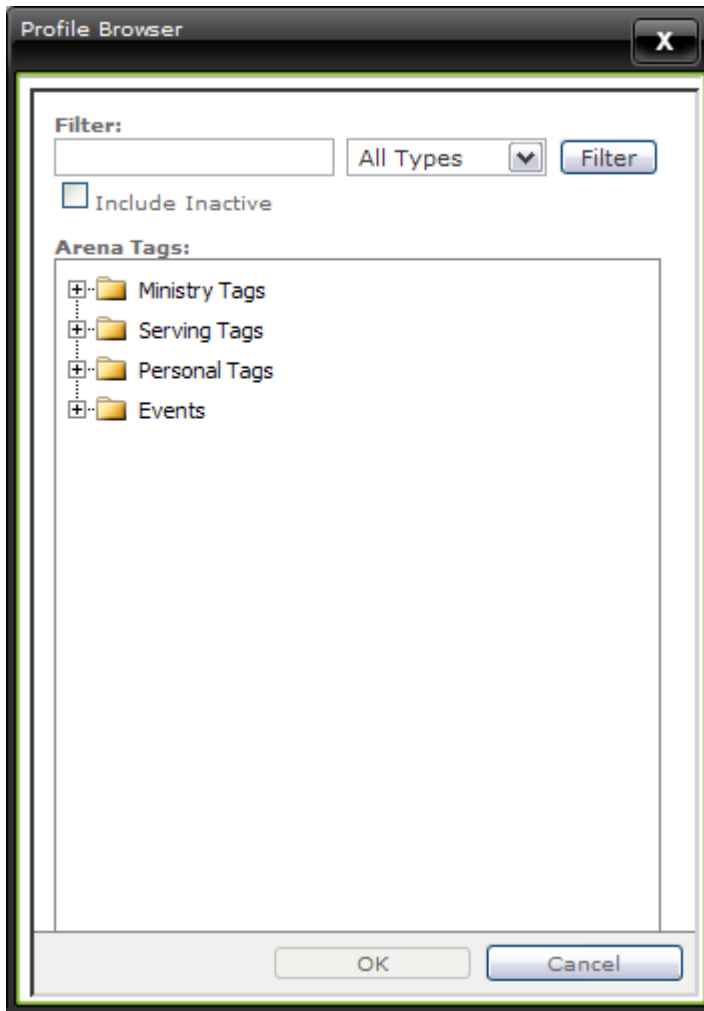
Clients who have supplied a **PhoneExpression** org setting with a regular expression can limit what kind of phone numbers are accepted in Arena. For example "`\"(?\\d{3})?[/s\\.\\-]?\\d{3}[/s\\.\\-]?\\d{4}.*`" requires a 10 digit phone number. They can also supply a PhoneExpressionError setting to provide what error is displayed if a phone number fails the given expression. By setting an expression the client can prevent international numbers as well.



```
<Arena:PhoneTextBox ID="tbPhone" runat="server" CssClass="formItem" ShowExtension="true" Required="true" EmptyValueMessage="Phone number is required!" />
```

Arena:ProfilePicker

This will render a modal picker of the tag structure from which the user can choose a tag/profile. It can be constrained to only show a specific type of tag.



```
<Arena:ProfilePicker ID="aProfilePicker" runat="server" CssClass="formItem"></Arena:ProfilePicker>
```

Simply set the type of profile type you wish to display and call the DataBind method.

```
aProfilePicker.ProfileType = Arena.Enums.ProfileType.Serving;  
aProfilePicker.DataBind();
```

UI Look and Feel

The look and feel of the User Interface is as important as the back-end design and functionality of a Module. Therefore custom Modules should have a high quality UI which is both pleasing to the eye and consistent with the existing look, feel and behaviors in Arena.

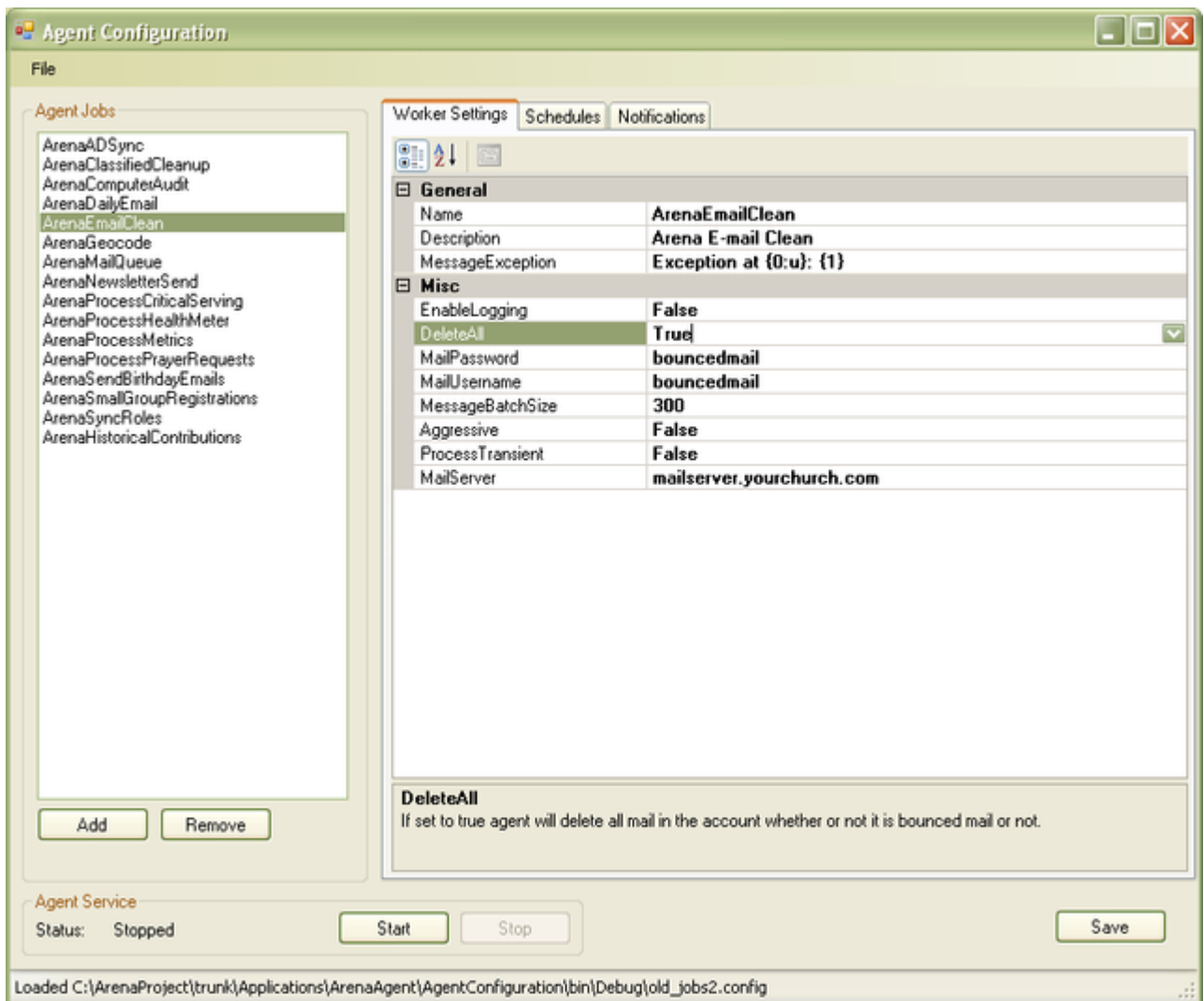
- Use the Arena UI Style Guide (see References) for all Module development. This document will show you how to properly structure your XHTML using standard container guidelines which will allow for the best CSS layout/style control.
- Follow existing UI patterns whenever possible.
- Use existing styles from existing style sheets whenever possible.
- Use the Photoshop image templates available on the Community site to create custom header and navigation icons (see 0). We recommend that you purchase images for your icons from [IconExperience](http://www.iconexperience.com/) (<http://www.iconexperience.com/>). Although there are other places that have high quality icons, they will not match the exact feel of the ones used in Arena.

Arena Agents

Arena Agents or more specifically "AgentWorkers" are classes that perform a batch type process or function which can be scheduled to run on a routine basis. The SDK includes all the official Arena Agents.

Your custom agent will inherit from the abstract AgentBase.AgentWorker class.

An administrator will typically configure an instance of your Agent using the Agent Configuration utility.



Agent Settings

Like Module Settings, Agent Settings are used to configure instances of your custom Agent.

Boolean (True/False)

This setting will render as a true/false radio button and takes an additional option that specifies the default value.

```
[BooleanSetting("Delete All", "If set to true agent will delete all mail in the account whether or not it is bounced mail or not.", true, true)]
public bool DeleteAll { get { return _deleteAll; } set { _deleteAll = value; } }
```

Numeric

This setting will render as a numeric textbox.

```
[NumericSetting("Message Batch Size", "Max number of e-mails to process with each running of the agent (recommended 300).", true)]
public int MessageBatchSize { get { return _messageBatchSize; } set { _messageBatchSize = value; } }
```

Text

This setting will render as a Textbox.

```
[TextSetting("Mail Username", "POP3 account to login to.", true)]
public string MailUsername { get { return _mailUsername; } set { _mailUsername = value; } }
```

Agent Setting Attributes

Agents Settings can be decorated with several attributes that are utilized by the Agent Configuration utility.

```
[Browsable(false)]
```

This attribute will prevent the setting from being displayed by the Agent Configuration utility.

```
[Description( "<Your description goes here.>" )]
```

The description attribute will be displayed for each selected property in the Agent Configuration utility.

References

The.Community Website

- <http://community.arenachms.com>

Code Generation

- CodeSmith Templates:
http://community.arenachms.com/files/folders/arena_codesmith_templates/default.aspx

Note: To avoid conflicts, do not name any of your table column names "ID" as the Arena CodeSmith templates will generate code that uses it internally.

Arena UI Style Guide

- <http://community.arenachms.com/files/folders/documents/entry1546.aspx>

Photoshop Image Templates

- <http://community.arenachms.com/files/folders/documents/entry184.aspx>

Arena Import/Export Instructions

- <http://community.arenachms.com/files/folders/documents/entry275.aspx>

API Documentation

- <http://community.arenachms.com/files/folders/documents/entry232.aspx>

VSS keyword Expansion

- http://msdn.microsoft.com/library/default.asp?url=/library/en-us/guides/html/vstskexpand_keywords.asp