akumina

InterChange Content App SDK

Developer Guide

Version 1.1 (May 2016)

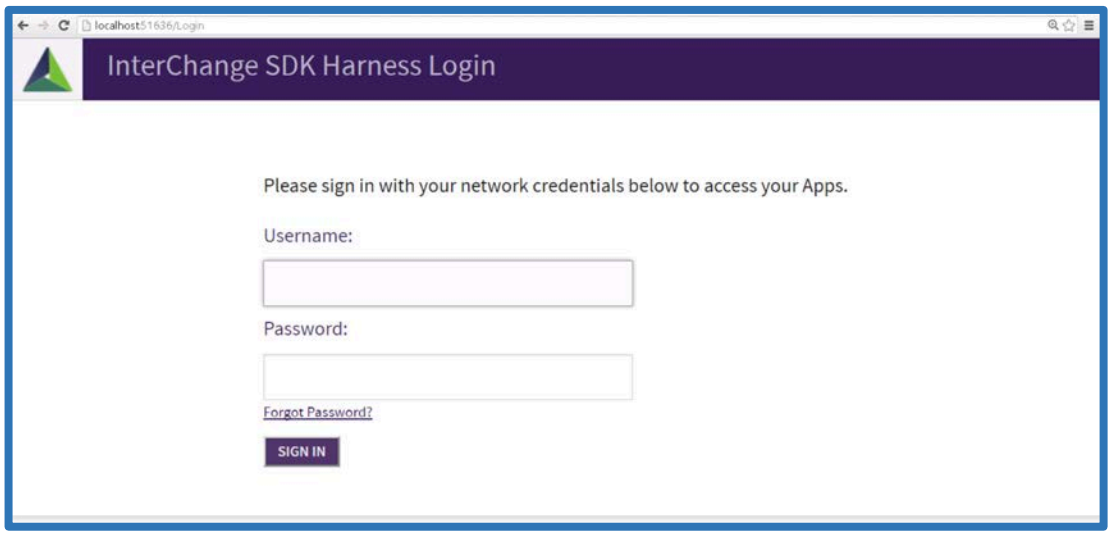
Table of Contents

Main Components of the SDK

The SDK Harness

This provides a mini-MVC application that provides the authentication and debugging environment to build your application. It contains the default functionality required by InterChange.

You can access the login page by navigating to /Login



Areas (MVC Concept)

When using the Content App SDK, your Content App essentially is an 'Area' inside of an MVC application. Both the Harness and InterChange interact with your Content App through this technology. To read more about Areas, visit the following MSDN link.

IContentApp Interface

This interface is the main interface used by InterChange in order to register your Content App for use. It also exposes functionality from the Harness to abstract crucial functionality from the core framework.

```
namespace Akumina.Interchange.Core.Interfaces
{
    public interface IContentApp
    {
        List<CustomSettingsField> UserSettings { get; set; }
        List<CustomNavigationItem> Navigation { get; }
        IContentAppService Service { get; set; }
        string Title { get; set; }
        string ListName { get; set; }
        string AdminGroup { get; set; }
        string SharepointURL { get; set; }
        string ListId { get; set; }
        string Layout { get; set; }
    }
}
```

Sample Use

```
namespace YourContentApp
{
    [Export(typeof(IContentApp))]
    public class YourAppTypeName : IContentApp
    {
        public string AdminGroup
        {
            get;set;
        }
        public string Layout
        {
            get;set;
        }
        public string ListId
        {
            get;set;
        }
        public string ListName
        {
            get;set;
        }
        public List<CustomNavigationItem> Navigation
        {
            get;
        }
        public IContentAppService Service
        {
            get;set;
        }
        public string SharepointURL
        {
            get;set;
        }
        public string Title
        {
            get;set;
        }
        public List<CustomSettingsField> UserSettings
        {
            get;set;
        }
    }
}
```

UserSettings

UserSettings are collected from the InterChange admin when choosing a list to use with your Content App. These settings will be made available to you under this property. You can hard code temporary settings for use in your application. They will be filled with actual settings from the user once added to InterChange.

Example of using Settings in your Custom Content App and how it shows up in InterChange

```
UserSettings = new List<CustomSettingsField>();
UserSettings.Add(new CustomSettingsField() { Name = "SDK FIELD 1", HelpText = "SDK HELP TEXT 1", Value = "SDK HARNESS 1" });
UserSettings.Add(new CustomSettingsField() { Name = "SDK FIELD 2", HelpText = "SDK HELP TEXT 2", Value = "SDK HARNESS 2" });
UserSettings.Add(new CustomSettingsField() { Name = "SDK FIELD 3", HelpText = "SDK HELP TEXT 3", Value = "SDK HARNESS 3" });
```

Custom Settings that show in InterChange

Navigation

Navigation is a Generic List of Links that will show in the Left hand navigation pane. This exposes a way for you to provide navigation around your application. InterChange and the Harness will automatically bind this data.

Service (IContentAppService)

Service is your mini Data Layer for talking to SharePoint. This should help speed up your development and provide basic CRUD operations when working with Lists and Items. You can certainly call the SharePoint API's directly, but this layer should be much easier to use.

```
namespace Akumina.Interchange.Core.Interfaces
{
    public interface IContentAppService
    {
        List<ActionResultMessage> Publish(Item item);
        List<ActionResultMessage> Update(Item item);
        List<ActionResultMessage> Add(Item item);
        Item Get(string itemId);
        CamlQueryResponse<List<Item>> GetList(Paging paging);
        MappingUser GetCurrentUser();
    }
}
```

Title

Title is a placeholder for the name of the Content App when the user adds it to InterChange. You can simply hardcode this to something like “My Content App” so you can see where the actual data will reside once in InterChange.

ListName / ListId

These properties should be used when interfacing with any Sharepoint calls, or Service calls. By default, the Service layer uses these properties when retrieving data. The user in InterChange will replace them with the chosen list. This is your way to interface with the eventually chosen ListName/ListId once deployed to InterChange.

AdminGroup

All Content Apps in InterChange require the user to define the Admin Group when interfacing with a List. The developer can set this property, as all Service calls will honor this permission when accessing Data. This will be overwritten when deployed to InterChange. If your user is not a part of this group when working with the Harness, the data will not be available.

SharepointURL

This property is required to properly authenticate using the Harness. Once your Content App is deployed to InterChange, this SharepointURL will be filled from the SiteAddress in the Global Settings.

Layout

This property is an Injection point used by the Harness and Interchange in terms of the Layouts that will be used by your Views. You donot need to set this, it will be injected by the Framework.

IContentAppController Interface

The controller interface is used in order to serve up routing from within InterChange. Without it, the routing will not work outside of the Harness itself. This interface also contains the IContentApp interface so that you can access the properties configured as a part of your Content App. It also will allow you to make CRUD calls into Sharepoint with very little effort.

```
namespace Akumina.Interchange.Core.Interfaces
{
    public interface IContentAppController
    {
        IContentApp ContentApp { get; set; }
    }
}
```

Sample Use

```
namespace YourContentApp
{
    [Export(typeof(IContentAppController))]
    public class HomeController : Controller, IContentAppController
    {
        public IContentApp ContentApp
        {
            get; set;
        }

        public ActionResult Index()
        {
            CamlQueryResponse<List<Item>> response = ContentApp.Service.GetList(new Paging()
```

```
{ PageIndex = 1, PageSize = 20 });

            ViewBag.Settings = ContentApp.UserSettings;
            ViewBag.Items = new
ConverterService().ConvertListingItems(response.RelevantResults);
            ViewBag.MappingUser = ContentApp.Service.GetCurrentUser();

            return View();
        }
    }
}
```

Important Details

References

These are not all of the references; we just called out the Akumina and Sharepoint specific dlls. All dlls required are found in the \packages folder. Check the SimpleContentApp project to see all references.

Akumina References

```
    <Reference Include="Akumina.Caching">
      <HintPath>..\packages\Akumina\Akumina.Caching.dll</HintPath>
    </Reference>
    <Reference Include="Akumina.Common">
      <HintPath>..\packages\Akumina\Akumina.Common.dll</HintPath>
    </Reference>
    <Reference Include="Akumina.Infrastructure">
      <HintPath>..\packages\Akumina\Akumina.Infrastructure.dll</HintPath>
    </Reference>
    <Reference Include="Akumina.Interchange.Core">
      <HintPath>..\packages\Akumina\Akumina.Interchange.Core.dll</HintPath>
    </Reference>
    <Reference Include="Akumina.InterChange.SDK.ContentAppHarness.Web">
      <HintPath>..\packages\Akumina\Akumina.InterChange.SDK.ContentAppHarness.Web.dll</HintPath>
    </Reference>
    <Reference Include="Akumina.Interchange.Services">
      <HintPath>..\packages\Akumina\Akumina.Interchange.Services.dll</HintPath>
    </Reference>
    <Reference Include="Akumina.Logging">
      <HintPath>..\packages\Akumina\Akumina.Logging.dll</HintPath>
    </Reference>
```

Sharepoint References

```
    <Reference Include="Microsoft.SharePoint.Client, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>..\packages\Akumina\Microsoft.SharePoint.Client.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.SharePoint.Client.DocumentManagement, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>..\packages\Akumina\Microsoft.SharePoint.Client.DocumentManagement.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.SharePoint.Client.Publishing, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>..\packages\Akumina\Microsoft.SharePoint.Client.Publishing.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.SharePoint.Client.Runtime, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
```

```
      <HintPath>..\packages\Akumina\Microsoft.SharePoint.Client.Runtime.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.SharePoint.Client.Search, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>..\packages\Akumina\Microsoft.SharePoint.Client.Search.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.SharePoint.Client.Search.Applications, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>..\packages\Akumina\Microsoft.SharePoint.Client.Search.Applications.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.SharePoint.Client.Taxonomy, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>..\packages\Akumina\Microsoft.SharePoint.Client.Taxonomy.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.SharePoint.Client.UserProfiles, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>..\packages\Akumina\Microsoft.SharePoint.Client.UserProfiles.dll</HintPath>
    </Reference>
    <Reference Include="Microsoft.SharePoint.Client.WorkflowServices, Version=16.0.0.0, Culture=neutral,
PublicKeyToken=71e9bce111e9429c, processorArchitecture=MSIL">
      <SpecificVersion>False</SpecificVersion>
      <HintPath>..\packages\Akumina\Microsoft.SharePoint.Client.WorkflowServices.dll</HintPath>
    </Reference>
```

Dependency Injection with Unity

The following services are injected by the Harness – the implementations reside in the Akumina.InterChange.Services.dll

```
IUnityContainer container = new UnityContainer();
container.RegisterType<IUserService, UserService>();
container.RegisterType<IContentTypeService, ContentTypeService>();
container.RegisterType<ISecurityService, SecurityService>();
container.RegisterType<IAppService, AppService>();
container.RegisterType<ICollaborationService, CollaborationService>();
container.RegisterType<IAppRepository, AppSpRepository>();
container.RegisterType<IWorkflowService, Workflow4Service>();
container.RegisterType<IConfigurationService, HarnessConfigurationService>();
container.RegisterType<ILoginService, LoginService>();
container.RegisterType<IContentAppService, ContentAppService>();
container.RegisterType<IDecisionService, DecisionService>();
container.RegisterType<IActionService, ActionService>();
ObjectFactory.Container = container;
```

Global.asax

The harness includes a default Global.asax implementation that handles routing and dependency injection. When creating your new app, or using the existing ‘Starter Apps’ you can inherit from the ‘GlobalApplication’ class from the Harness.

```
using Akumina.InterChange.SDK.ContentAppHarness.Web;

namespace YourContentApp
{
    public class MvcApplication : GlobalApplication
    {
        protected override void Application_Start()
        {
            base.Application_Start();
        }

        protected override void Application_Error()
        {
            Exception exception = Server.GetLastError();

            base.Application_Error();
        }


    }
}
```

Routing

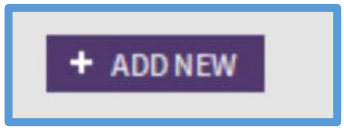
This is the out of the box routing provided by the Harness, you can add your own in the Global.asax Application_Start event, be sure to use the DataTokens and register the routes under your Area. The GetCurrentAppConfiguration() call will read the name of your class that inherits 'IContentApp' – see samples above.

```
AppConfiguration currentApp =
ObjectFactory.Get<IConfigurationService>().GetCurrentAppConfiguration();
if (currentApp != null)
{
RouteTable.Routes.MapRoute("AddSave", currentApp.Type + "/{id}/Add/Save",
new { controller = "Add", action = "Save" }).DataTokens.Add("area", currentApp.Type);

RouteTable.Routes.MapRoute("View", currentApp.Type + "/{id}/{controller}/{itemid}",
new { controller = "View", action = "Index" }).DataTokens.Add("area", currentApp.Type);

RouteTable.Routes.MapRoute("Add", currentApp.Type + "/{id}/Add",
new { controller = "Add", action = "Index" }).DataTokens.Add("area", currentApp.Type);

RouteTable.Routes.MapRoute("Edit", currentApp.Type + "/{id}/{controller}/{itemid}/{action}",
new { controller = "Edit", action = "Index" }).DataTokens.Add("area", currentApp.Type);

RouteTable.Routes.MapRoute("Default", currentApp.Type + "/{id}",
new { controller = "Home", action = "Index" }).DataTokens.Add("area", currentApp.Type);
}
```

ViewBag

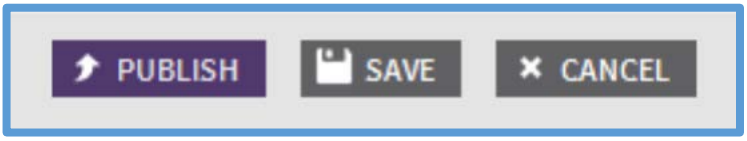
```
ViewBag.ShowListingHeaderOptions = true;
```

Enables the following button



```
ViewBag.ShowItemHeaderOptions = true;
```

Enables the following buttons



You can bind to the Add New button using the following

```
$(document).ready(function () {

    $(".ak-btn-save").click(function (event) {
        var action = $(event.target).text();
        $("#action").val(action);
        $("#formSave").submit();
        event.preventDefault();

    });

    $(".ak-btn-publish").click(function (event) {
        var action = $(event.target).text();
        $("#action").val(action);
        $("#formSave").submit();
        event.preventDefault();

}
```

Sample Service Code (using the IContentAppService)

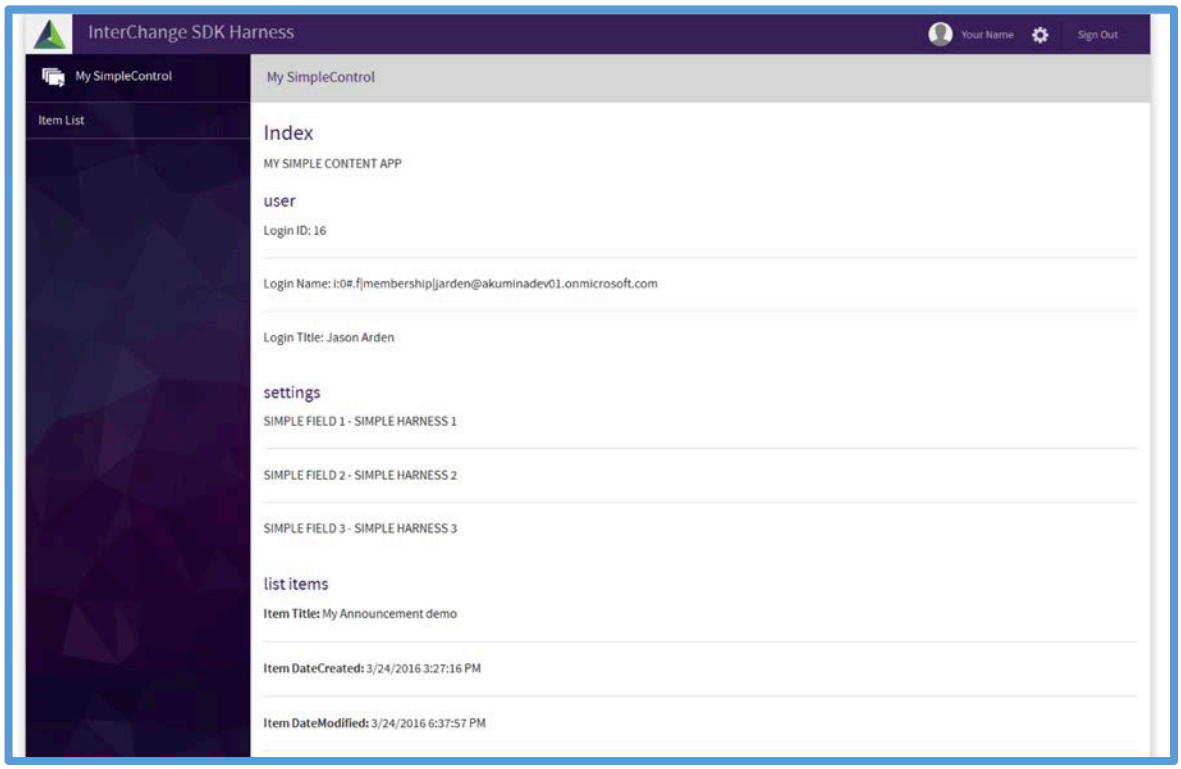
Here is an example of using ContentApp property when Inherting from IContentAppController

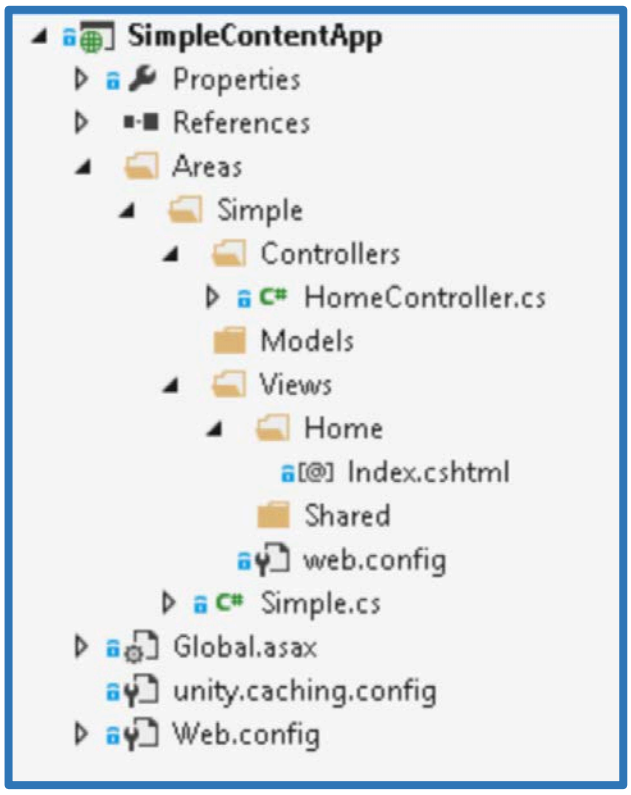
```
//get 1st page of items, 20 items per page
CamlQueryResponse<List<Item>> response = ContentApp.Service.GetList(new Paging() { PageIndex
= 1, PageSize = 20 });
//get current user
MappingUser map = ContentApp.Service.GetCurrentUser();
//convert response using ConverterSrervice
ViewBag.Items = new ConverterService().ConvertListingItems(response.RelevantResults);
```

Starter Content Apps

Simple Content App

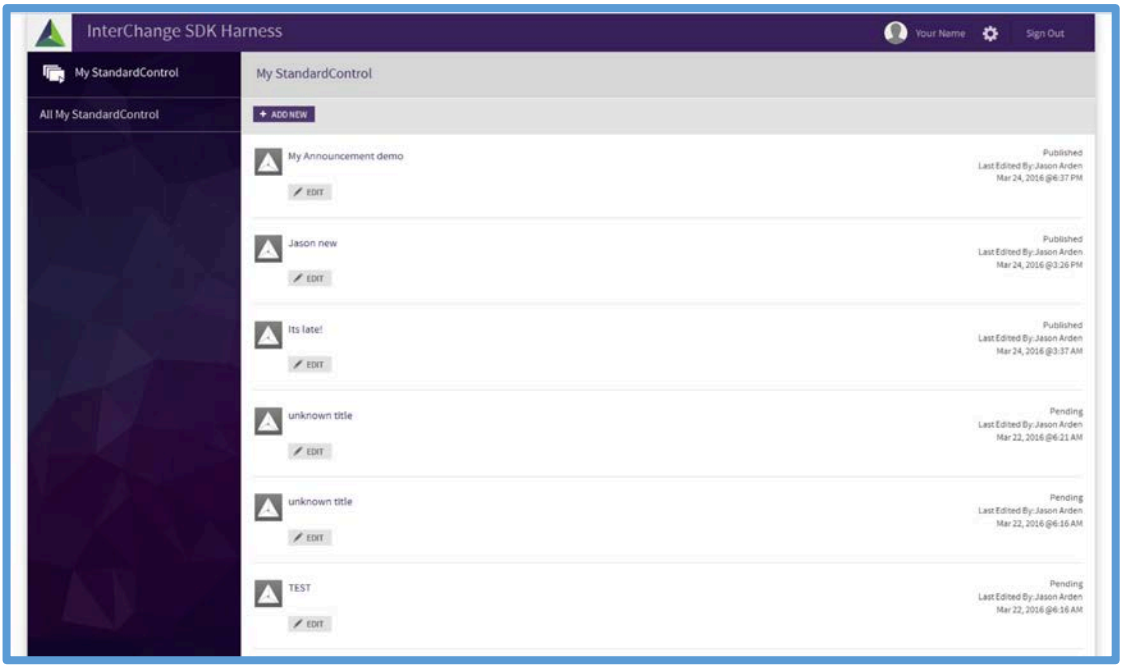
The Simple Content App is a very basic 1-Controller, 1-View ‘Hello World’ style app that shows you the basics for creating your first Content App.

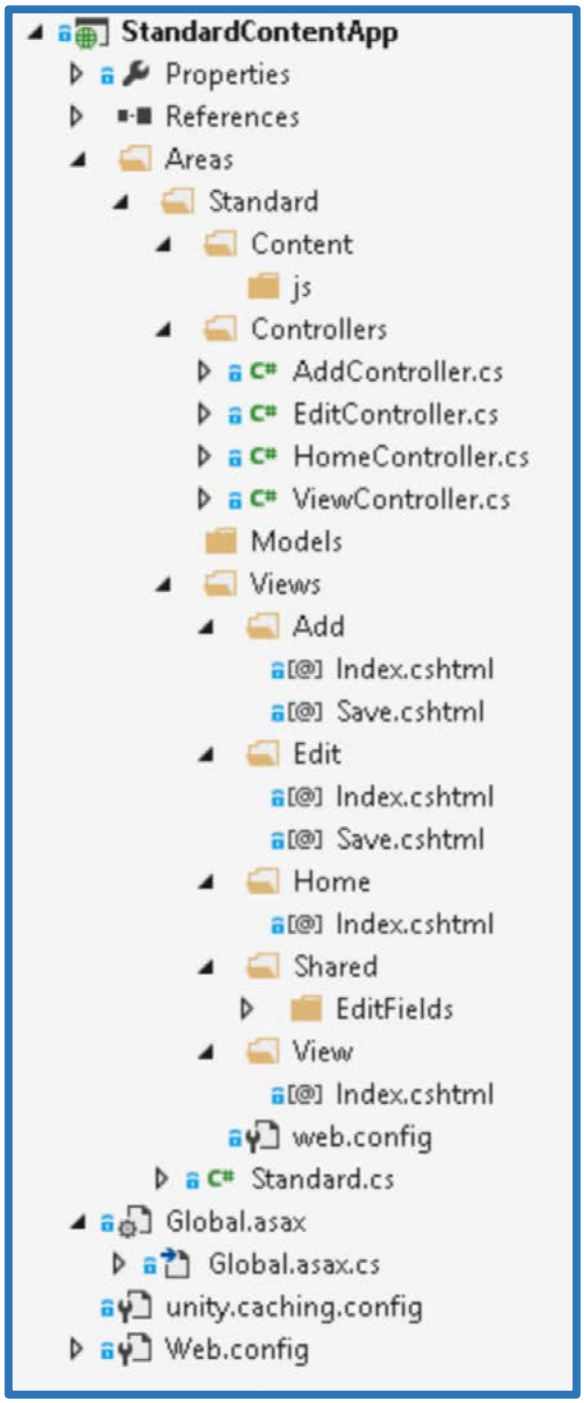




Standard Content App

The Standard Content App is most likely the default use for many of your Content Apps. It provides Listing, Editing, Adding and Updating List Items. It allows you to completely change the UI around editing a List / List Item.


InterChange SDK Harness
Your Name
Sign Out
My StandardControl
All My StandardControl
My StandardControl
+ ADD NEW
My Announcement demo
EDIT
Published
Last Edited By: Jason Arden
Mar 24, 2016 @6:37 PM
Jason new
EDIT
Published
Last Edited By: Jason Arden
Mar 24, 2016 @3:26 PM
Its late!
EDIT
Published
Last Edited By: Jason Arden
Mar 24, 2016 @3:37 AM
unknown title
EDIT
Pending
Last Edited By: Jason Arden
Mar 22, 2016 @6:21 AM
unknown title
EDIT
Pending
Last Edited By: Jason Arden
Mar 22, 2016 @6:16 AM
TEST
EDIT
Pending
Last Edited By: Jason Arden
Mar 22, 2016 @6:16 AM


StandardContentApp
Properties
References
Areas
Standard
Content
js
Controllers
AddController.cs
EditController.cs
HomeController.cs
ViewController.cs
Models
Views
Add
Index.cshtml
Save.cshtml
Edit
Index.cshtml
Save.cshtml
Home
Index.cshtml
Shared
EditFields
View
Index.cshtml
web.config
Standard.cs
Global.asax
Global.asax.cs
unity.caching.config
Web.config

Adding Your Content App to Interchange

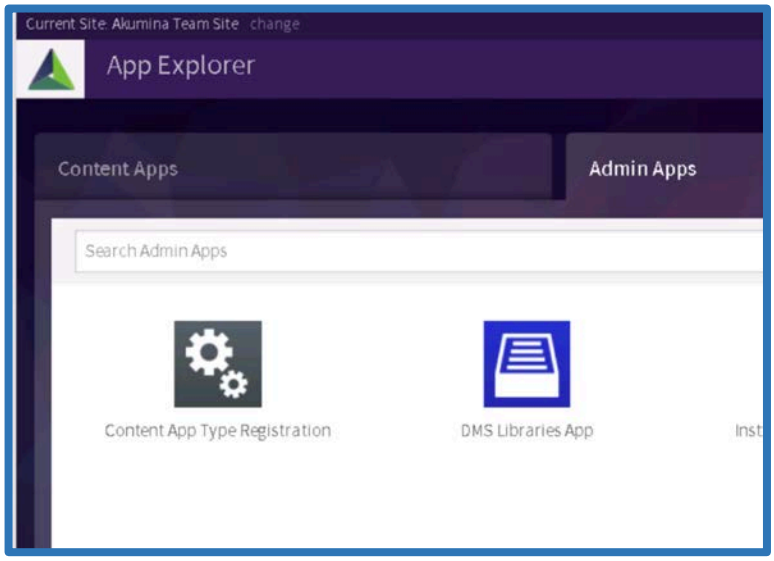
Content App Packaging

Namespace of class inheriting from IContentApp should match your Controllers

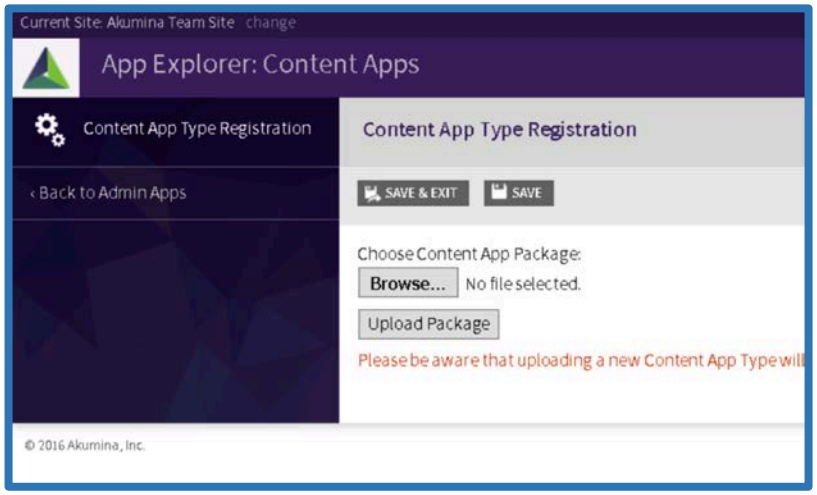
Uploading Your Content App Package

a. Log into InterChange
b. Click on Admin Tab
c. Choose ‘Content App Type Registration’
d. Click on ‘Upload new Content App package’
e. Browse to your zip file created in the ‘Content App Packaging’ step above and click Upload
f. Please be aware uploading a content app package will recycle the app pool
g. Verify the progress output below the Upload button to validate your Content App was recognized and that the ‘Area’ files get extracted properly
h. If you have server access, verify your Area folder shows up under webroot\Areas
i. Verify that your DLL has been extracted into webroot\target directory

Content App Type Registration



Content App Type Registration Upload



Status after clicking Upload Package

- Uploading Content App Package...
- Uploading Content App Package...DONE
- Extracting Content App Package...
- Extracting Content App Package...DONE, Found **Standard**
- Confirming Content App Package and Deploying 'binaries' to InterChange...
- Confirming Content App Package and Deploying 'binaries' to InterChange...DONE
- Extracting Content App Package and Deploying 'Area' to InterChange...
- Deploying File: \Standard\Views\web.config
- Deploying File: \Standard\Views\Add\Index.cshtml
- Deploying File: \Standard\Views\Add\Save.cshtml
- Deploying File: \Standard\Views\Edit\Index.cshtml
- Deploying File: \Standard\Views\Edit\Save.cshtml
- Deploying File: \Standard\Views\Home\Index.cshtml
- Deploying File: \Standard\Views\Shared\EditFields_CheckBox.cshtml
- Deploying File: \Standard\Views\Shared\EditFields_Date.cshtml
- Deploying File: \Standard\Views\Shared\EditFields_LabelHeader.cshtml
- Deploying File: \Standard\Views\Shared\EditFields_RichText.cshtml
- Deploying File: \Standard\Views\Shared\EditFields_Text.cshtml
- Deploying File: \Standard\Views\View\Index.cshtml
- Extracting Content App Package and Deploying 'Area' to InterChange...DONE
- Bouncing App Pool, you will be required to login again...
- Bouncing App Pool, you will be required to login again...DONE

Extraction Details

Temp path configuration

In the file interchange.settings.config

Default Value: C:\temp

<add key="akumina:tempuploadpath " value="C:\adifferentpathhere " />

Directories affected during Extraction

1. Temp path directory (“C:\temp”)
2. Webroot\Areas
3. Webroot\target
    a. Webroot\target\temp (this is the shadow directory for \target, files will get deleted and re-copied here during app startup)

Enabling/Disabling your Content App

b. Log into InterChange
c. Click on Admin Tab
d. Choose ‘Content App Type Registration’
e. You should see your new Content App Type in the list with a checkbox next to it
f. Simply check the box to enable the app and click on ‘Save’ or ‘Save & Exit’

Enable / Disable Content App Types