HOW TO DEVELOP API APPS

COUNTERSOFT

How to Develop Api Apps

Contents

1. Introduction

Gemini has lots of features built in, which you can explore in our docs. However there are times when you need something to work very specifically to your requirements in which case you can create your own apps using the Gemini App Framework.

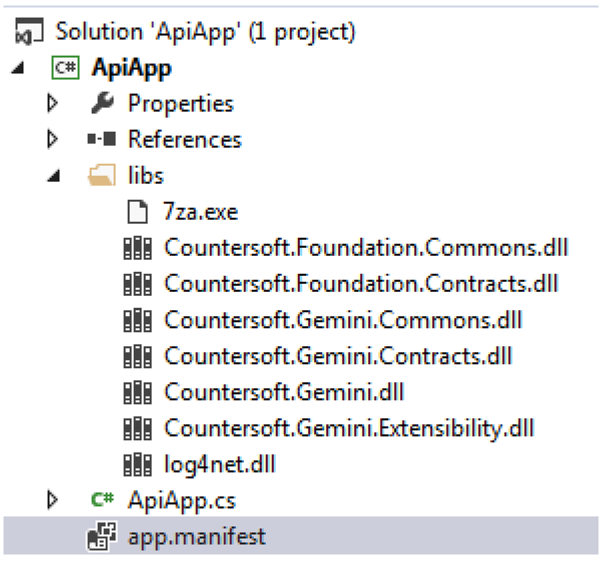
This guide will show you how to create and deploy a Gemini API app. API apps allow you to extend the Gemini core API with your own custom routing and controllers to execute your code.

2. Create Project

Open Visual Studio and download our project template *Gemini API App Project Template* from Tools -> Extensions and Updates -> Online. Close and open Visual Studio again in order to select the new template from File -> New -> Project -> Visual C#. Enter a *Name* and *Solution name*.

3. Structure

The new template will create a project with the following structure:



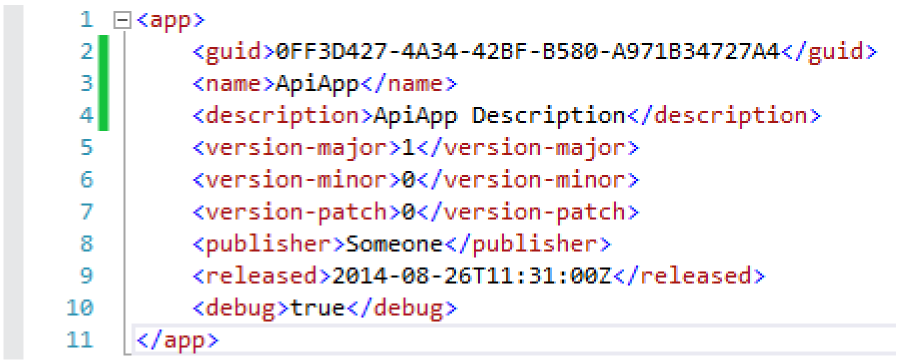
```
Solution 'ApiApp' (1 project)
  ApiApp
    Properties
    References
    libs
      7za.exe
      Countersoft.Foundation.Commons.dll
      Countersoft.Foundation.Contracts.dll
      Countersoft.Gemini.Commons.dll
      Countersoft.Gemini.Contracts.dll
      Countersoft.Gemini.dll
      Countersoft.Gemini.Extensibility.dll
      log4net.dll
    ApiApp.cs
    app.manifest
```

Libs
Contains all relevant DLL’s the app uses.

Manifest
The app.manifest file describes your app.

4. App.Manifest

The manifest file describes your app and is used during deployment.



```
<app>
    <guid>0FF3D427-4A34-42BF-B580-A971B34727A4</guid>
    <name>ApiApp</name>
    <description>ApiApp Description</description>
    <version-major>1</version-major>
    <version-minor>0</version-minor>
    <version-patch>0</version-patch>
    <publisher>Someone</publisher>
    <released>2014-08-26T11:31:00Z</released>
    <debug>true</debug>
</app>
```

Option	**Description**
GUID	Globally unique identifier (GUID) for your app
Name	Application title displayed in screen setup for Administrators to identify your app
Description	Brief description of the application
Version Major	Semantic versioning for your app
Version Minor	
Version Patch	
Publisher	Name of the organization/individual who owns the app
Released	Date the app was released
Debug	Can be *true* or *false*. When *true* app is not cached and can be changed on disk (useful during development)

5. Attributes

Set attributes to describe your app.

```
[AppType(AppTypeEnum.API),
AppGuid("0FF3D427-4A34-42BF-B580-A971B34727A4"),
AppAuthor("Countersoft"),
AppName("ApiApp"),
AppDescription("ApiApp sample description")]
[OutputCache(Duration = 0, NoStore = true, Location = System.Web.UI.OutputCacheLocation.None)]
public class ApiAppController : BaseController
{
```

Attributes

Option	Description
AppType	The type of app - Widget, Event, Timer etc.
AppGuid	Globally unique identifier (GUID) for your app
AppAuthor	Your name
AppName	Application title displayed in Screen setup for Administrators to identify your app
AppDescription	Brief description of the application

6. Project Setup

a) Replace all Countersoft DLL’s in the libs folder with the latest Gemini DLL’s from your %Gemini%/bin installation folder.

b) Open ApiApp.cs and change the namespace, class name and all other references from “ApiApp” to something unique. Use a name relevant to its function.

c) Replace the AppGuid with a new GUID.

Open the app.manifest and make sure the Guid is changed to the AppGuid. You can change the name and description as well. Leave the debug property as “true” for as long as you are developing your app. Change it to “false” when the app is ready to go live.

7. Authentication

Add the *AuthorizeApi* attribute to force authentication when accessing your methods.

```
public class ApiAppController : BaseApiController
{
    [AuthorizeApi]
    [System.Web.Mvc.HttpGet]
    public Issue GetItem(int issueid)
    {
```

There are two ways to authenticate a user.

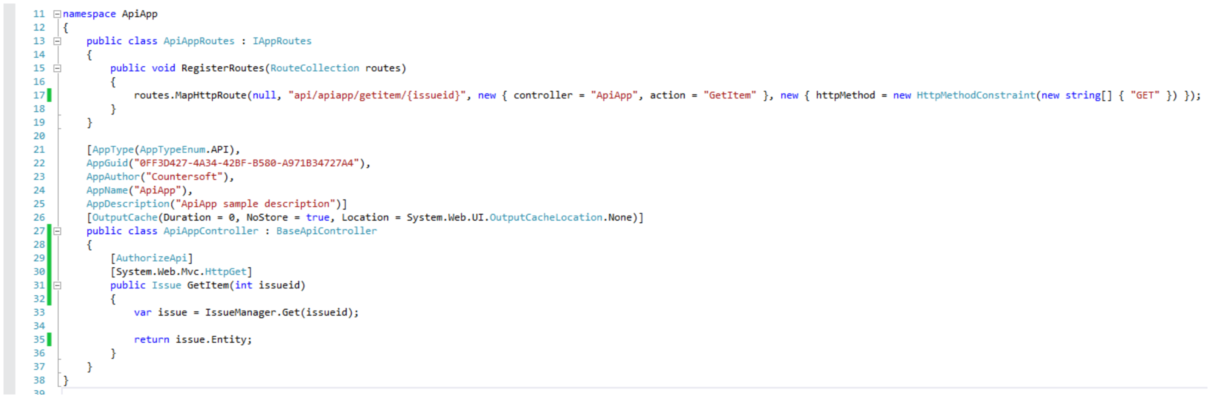
1. Using basic authentication

2. Passing the authentication credentials as a base64 encoded string in the URL e.g. http://{geminiUrl}/api/custom/getitem/{issueid}?auth=base64EncodedCredentials

You can use your username:apikey or username:password to authenticate a user. If you're using the username:password combination then the password must be additionally **md5 encrypted**.

8. Routes and Controller

The example below shows you how to create the routes and controller in an API app.



```
namespace ApiApp
{
    public class ApiAppRoutes : IAppRoutes
    {
        public void RegisterRoutes(RouteCollection routes)
        {
            routes.MapHttpRoute(null, "api/apiapp/getitem/{issueid}", new { controller = "ApiApp", action = "GetItem" }, new { httpMethod = new HttpMethodConstraint(new string[] { "GET" }) });
        }
    }

    [AppType(AppTypeEnum.API),
    AppGuid("0FF3D427-4A34-42BF-B580-A971B34727A4"),
    AppAuthor("Countersoft"),
    AppName("ApiApp"),
    AppDescription("ApiApp sample description")]
    [OutputCache(Duration = 0, NoStore = true, Location = System.Web.UI.OutputCacheLocation.None)]
    public class ApiAppController : BaseApiController
    {
        [AuthorizeApi]
        [System.Web.Mvc.HttpGet]
        public Issue GetItem(int issueid)
        {
            var issue = IssueManager.Get(issueid);

            return issue.Entity;
        }
    }
}
```

The "GetItem" method returns the full Issue. There are two ways to query this method.

1. You can query the method using the URL http://{geminiUrl}/api/custom/getitem/{issueid} as a GET request**.** Don't forget to include the authentication header.

2. You can query the method using the Gemini ServiceManager

```
ServiceManager serviceManager = new ServiceManager("http://localhost/gemini", "manager", "manager", string.Empty);
var issue = serviceManager.GenericService.Get<Issue>("/apiapp/getitem/237716");
```

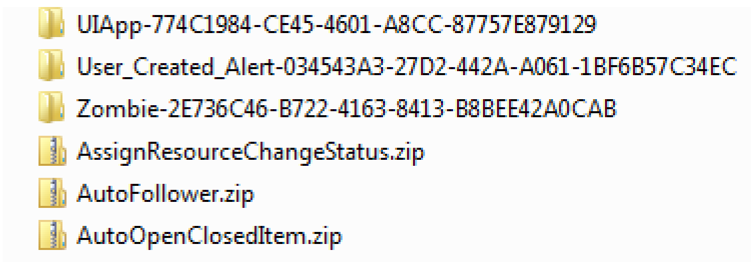
The route "api/apiapp/getitem/{issueid}" can be anything you want. Just be aware that Gemini is using routes as well and ensure that your route does not collide with Gemini, otherwise your methods might not be called. You can see all Gemini routes in the file Global.asax which you can find in the root folder of the Gemini installation directory.

List of important services, which offer methods to retrieve, create, update and delete items.

Name	**Description**
ProjectTemplateManager	Provides access to project templates
ProjectManager	Provides access to projects
IssueManager	Provides access to issues
UserManager	Provides access to users
CustomFieldManager	Provides access to custom fields
NavigationCardsManager	Provides access to workspaces
MetaManager	Provides access to meta data (status, priority etc.)

9. Deployment

Apps are deployed to the App_data/Apps folder which is located where you installed Gemini on your web server.



```
UIApp-774C1984-CE45-4601-A8CC-87757E879129
User_Created_Alert-034543A3-27D2-442A-A061-1BF6B57C34EC
Zombie-2E736C46-B722-4163-8413-B8BEE42A0CAB
AssignResourceChangeStatus.zip
AutoFollower.zip
AutoOpenClosedItem.zip
```

ZIP files are the packaged apps and the folders are deployed apps.

a) We use Post-Build Events in the project to generate the ZIP file. Build your solution and you should find your app's zip file in the bin/target directory.

b) Stop your Gemini application pool and copy the ZIP into %Gemini%/App_Data /apps

c) Start your Gemini application pool and this should extract the content of the ZIP into a folder

The app should now be available in Gemini. Make sure all relevant permissions are assigned for the app in Customize -> Apps -> AppName and also enable the app in Customize -> Template -> Process screens.

When you re-deploy the app then make sure the app's ZIP file and the folder are deleted from %Gemini%/App_Data/apps before you copy in the new ZIP file.

10. References Examples

Several Gemini app examples with source code http://countersoft.github.io/

Docs http://docs.countersoft.com/developing-custom-apps/