HOW TO DEVELOP TIMER APPS

COUNTERSOFT

How to Develop Timer Apps

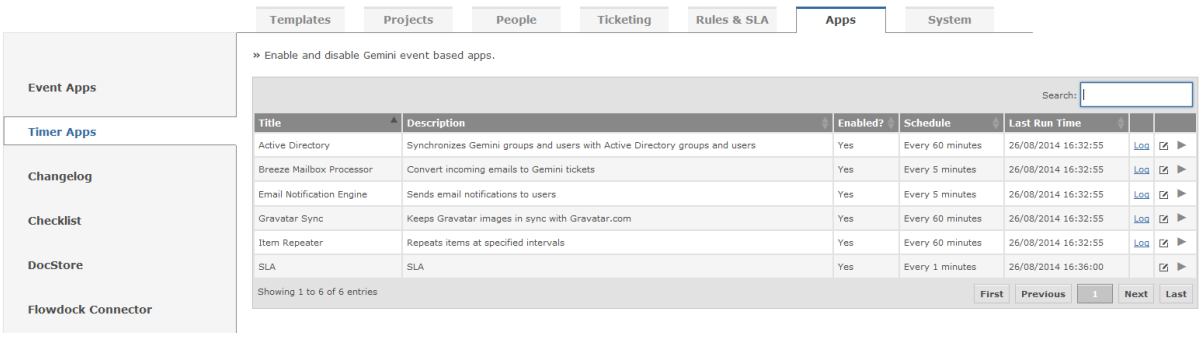
Contents

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 timer app. A timer app allows you to invoke custom logic at set intervals. For example, you may wish to check for new comments on items every 15 minutes.

Gemini Administrators can enable or disable timer apps and set their schedule.

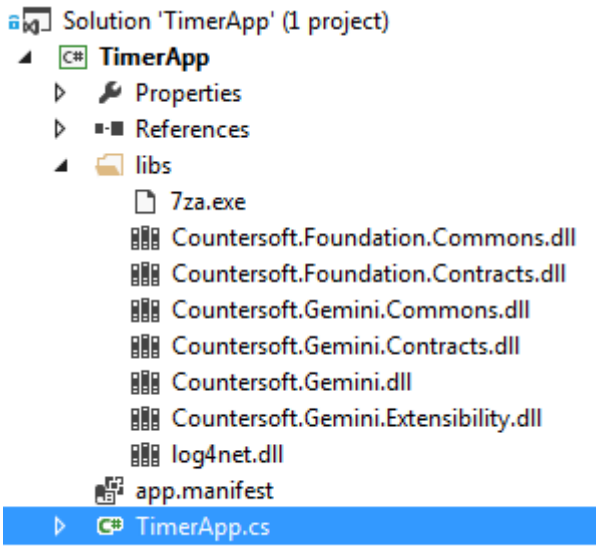


2. Create Project

Open Visual Studio and download our project template *Gemini Timer 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.



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>7FAC56D6-C8FA-49B7-8585-B65327436802</guid>
    <name>TimerApp</name>
    <description>TimerApp 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.

```
21  namespace TimerApp
22  {
23      [AppType(AppTypeEnum.Timer),
24      AppGuid("7FAC56D6-C8FA-49B7-8585-B65327436802"),
25      AppAuthor("Countersoft"),
26      AppName("TimerApp"),
27      AppDescription("TimerApp sample description")]
```

Attributes

Option	Description
AppType	The type of app i.e. 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 TimerApp.cs and change the namespace, class name and all other references from "TimerApp" 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. Methods and Services

These are the available methods you can override.

Name	**Description**
Run(IssueManager)	Called on every time interval and requires an *IssueManager* parameter. Add your app logic in here.
GetInterval(IGlobalConfigurationWidgetStore)	Gemini calls this method to retrieve the saved interval value. Requires an IGlobalConfigurationWidgetStore parameter.
SetInterval(IGlobalConfigurationWidgetStore dataStore, TimerJobSchedule schedule)	Optional method. Allows you to add your own logic for setting the interval.

The *Run* method provides direct access to the *IssueManager*, but you can access any other manager by passing in the *issueManager*.

```
public override bool Run(IssueManager issueManager)
{
    //Add your logic here
    UserManager userManager = new UserManager(issueManager);

    return true;
}
```

You can override the GetInterval method to add your custom logic. When Gemini requests the interval for the app you can set the method to send back a default value if the user hasn't set an interval yet.

```
public override TimerJobSchedule GetInterval(IGlobalConfigurationWidgetStore dataStore)
{
    var data = dataStore.Get<TimerJobSchedule>(AppGuid);

    if (data == null || data.Value == null || (data.Value.Cron.IsEmpty()
        && data.Value.IntervalInHours.GetValueOrDefault() == 0
        && data.Value.IntervalInMinutes.GetValueOrDefault() == 0))
    {
        // Interval 60 minutes is default
        return new TimerJobSchedule(60);
    }

    return data.Value;
}
```

You can override the SetInterval method to add your custom logic.

```
public override void SetInterval(IGlobalConfigurationWidgetStore dataStore, TimerJobSchedule schedule)
{
    //Custom Logic

    base.SetInterval(dataStore, schedule);
}
```

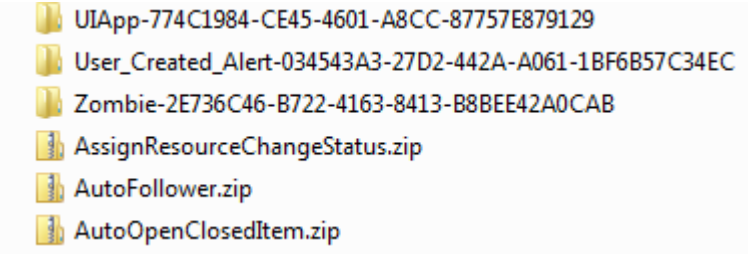
Only one instance of the app can run at a time. That means if you set the interval to 1 minute and it takes 5 minutes for the app to complete, then the timer app will actually run every 5 minutes.

List of some 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.)

8. Deployment

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



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.

9. References Examples

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

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