21

Organization with Areas

This chapter covers:

* Organizing large applications with areas
* Using T4MVC to help manage links and URLs
* Applying security to separate areas

As ASP.NET MVC websites become larger and more complex, the number of controllers inevitably grows. With a large number of controllers, we start to notice many controllers might logically belong together as a group. We might have administration sections of our application, product catalog sections, customer care sections, shopping cart and ordering sections and so on. Each of these application areas will likely share nothing more than perhaps a common logon widget or a master page. However, each application area probably has quite a lot of common functionality with other controllers and views within that area. To help tame large applications, ASP.NET MVC 2 introduces the concept of areas. Areas allow us to segregate controllers, models and views into different physical locations. In this chapter, we will examine using areas to separate our application's different concerns. Next, we will use the T4MVC templates to help us generate our URLs and links between areas. Finally, we will look at securing content separated into areas.

21.1 Creating a basic Area

To create our first area, we can start by right-clicking the project in the Solution Explorer and selecting Add > Area..., as shown in figure 21.1.

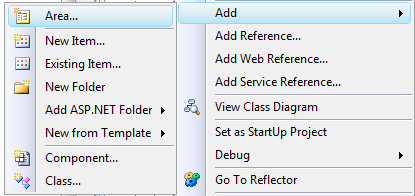


Figure 21.1 The Add Area menu option

Selecting Add Area brings up the Add Area dialog box, where we need to enter an Area name, shown in figure 21.2.

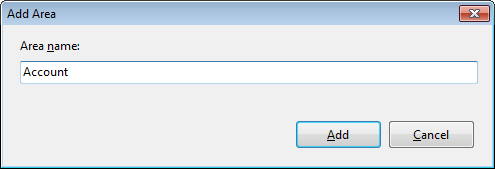


Figure 21.2 The Add Area dialog box

When the first area is created, a new top-level Areas folder is added to the MVC project. Inside this Areas folder, each individual area resides in its own folder. In each Area folder, you will find folders for controllers, models and views specific to that area. Finally, the Add Area wizard adds an area registration class. The project shown in figure 21.3 includes three areas for administration, product catalog and account information.

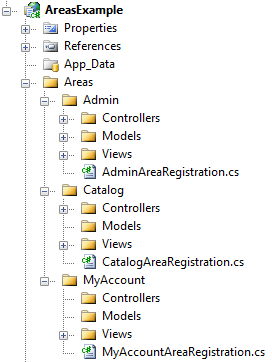


Figure 21.3 A project with three separate areas

The Add Area wizard is included with the ASP.NET MVC 2 installer, but we are not forced to use the wizard. The wizard creates the correct folder structure and area registration class. If the tooling were not available to us for some reason, we simply need to follow the same folder structure conventions. Besides the folder structure, the wizard creates an important area registration class. This class contains information describing the name and routing information pertinent to our area, and allows us to modify the default area registration information. If we used the wizard, our area registration class would similar to listing 21.1.

Listing 21.1 The default area registration class

public class AdminAreaRegistration : AreaRegistration

{

public override string AreaName

{

get

{

return "Admin";

}

}

public override void RegisterArea(AreaRegistrationContext context)

{

context.MapRoute(

"Admin\_default",

"Admin/{controller}/{action}/{id}",

new { controller = "Profile", action = "Index", id = "" }

);

}

}

Our AdminAreaRegistration class contains area registration information, and inherits from the MVC class AreaRegistration. AreaRegistration is an abstract class with one abstract property, AreaName, and one abstract method, RegisterArea. The AreaName property is used later for routing purposes. The RegisterArea method accepts a single AreaRegistrationContext object. The AreaRegistrationContext contains properties and methods which we can use to describe our area. In general, we can simply use the MapRoute method to describe the routes our area should use. In the above example, all route URLs starting with "Admin" will be directed to controllers in the Admin area.

Including route information, the AreaRegistrationContext also allows us to configure our area's type namespace. By default, the Namespaces property will contain

21.2 Managing links and URLs with T4MVC

21.3 Securing Areas