业务和记录一些

1.

权限过滤器：

/// <summary>

/// 权限过滤器 根据一角色一过滤器 主要在个人中心或是后台管理的时候用上

/// </summary>

[AttributeUsage(AttributeTargets.Method | AttributeTargets.Class, Inherited = true, AllowMultiple = true)]

public class AdminAuthorizeAttribute : FilterAttribute, IAuthorizationFilter

{

/// <summary>

/// 关于授权

/// </summary>

/// <param name="filterContext"></param>

public void OnAuthorization(AuthorizationContext filterContext)

{

if (\_dontValidate)

return;

if (filterContext == null)

throw new ArgumentNullException("filterContext");

if (OutputCacheAttribute.IsChildActionCacheActive(filterContext))

throw new InvalidOperationException("您无法使用[AdminAuthorize]属性，当一个子动作缓存处于活动状态");

if (IsAdminPageRequested(filterContext))

{

这里调用SSO单点登陆

SSOHandle(filterContext);

}

}

/// <summary>

/// SSO单点登录

/// </summary>

/// <param name="filterContext"></param>

public void SSOHandle(AuthorizationContext filterContext)

{

var workContext = EngineContext.Current.Resolve<IWorkContext>();

var currentCustomer = workContext.CurrentCustomer;

//未登录

if (!IsLogin(currentCustomer))

{

var queryToken = filterContext.HttpContext.Request["token"];

if (queryToken != null)

{

if (queryToken == "$Token$")

{

filterContext.Result = new RedirectResult(getTokenURL(filterContext));

return;

}

//持有令牌

//调用WebService获取主站凭证

string webServiceUrl = string.Format("{0}/TokenService.asmx?wsdl", FormsAuthentication.LoginUrl);

WebServiceProxy proxy = new WebServiceProxy(webServiceUrl);

object[] para = new object[1] { queryToken };

var o = proxy.ExecuteQuery("TokenGetCredence", para);

if (o != null)

{

var \_authenticationService = EngineContext.Current.Resolve<IAuthenticationService>();

\_authenticationService.SignIn((string)o, false);

//令牌正确

string sourceUrl = filterContext.HttpContext.Request.Url.AbsoluteUri;

sourceUrl = Regex.Replace(sourceUrl, @"\?Token=[^&]\*&?", string.Empty);

sourceUrl = Regex.Replace(sourceUrl, @"&Token=[^&]\*&?", string.Empty);

filterContext.Result = new RedirectResult(sourceUrl);

return;

}

else

{

//令牌错误

filterContext.Result = new RedirectResult(GetLoginURL(filterContext));

return;

}

}

else

{

filterContext.Result = new RedirectResult(getTokenURL(filterContext));

return;

}

}

//登录后检验权限

var permissionService = EngineContext.Current.Resolve<IPermissionService>();

bool result = permissionService.Authorize(StandardPermissionProvider.AdministratorAccessPanel);

if (!result)

{

filterContext.Result = new RedirectResult("~/NoPermission");

return;

}

}

关于过滤器中OnAuthorization()方法的查找：

mvc中OnActionExecuting和 OnAuthorization：

执行顺序，OnAuthorization在AuthorizeAttribute特性种，

先执行OnAuthorization，然后执行AuthorizeCore函数；

OnActionExecuting方法在ActionFilterAttribute特性中，

同时存在OnAuthorization和OnActionExecuting，在授权检查完毕之后执行OnActionExecuting，也就是说OnActionExecuting后执行

最后该过滤器用的地方：

namespace eWorld.Web.Controllers

{

[NopHttpsRequirement(SslRequirement.Yes)]

[AdminValidateIpAddress]

[AdminAuthorize]

public abstract partial class BaseAdminController : BaseController

{

原来是Web上面，并不是Web API

2.

3.

4.

5.

6.

7.

8.