ABP提供了一个基础设施，它可以自动记录所有和应用程序交互的日志。

记录的信息有：相关的**租户Id**，调用者的**用户Id**，调用的 **服务名称**（调用方法的类名），调用的**方法名** ， 执行**方法名**（序列化成Json） ， **执行时间** ， 执行**时长**（单位毫秒），客户端**IP地址** ，客户端**计算机名称**和**异常**（抛出异常的情况下）。

审计系统使用[IAbpSession](http://www.cnblogs.com/farb/p/ABPSession.html" \o "IAbpSession)来获得当前的UserId和TenantId。

审计系统使用了IAuditingStore来存储信息。如果你没有实现该接口，那么默认会使用SimpleLogAuditingStore，并且它会将审计信息记录到日志中（log）。

### **配置**

要配置审计，可以在模块的PreInitialize方法中使用**Configuration.Auditing**属性。审计是**默认开启的**。可以像下面那样关闭它：

public class MyModule : AbpModule

{

public override void PreInitialize()

{

Configuration.Auditing.IsEnabled = false;

}

//...

}

这儿是审计配置属性的列表：

**IsEnabled**: 用于完全开启或关闭审计系统。默认为true。

**IsEnabledForAnonymousUsers**: 如果此值为true,那么没有登录到系统的用户的审计日志也会保存。默认为false。

**MvcControllers**: 用于为ASP.NET MVC控制器配置审计。

**IsEnabled**: 用于为MVC控制器开启或关闭审计。默认为true

**IsEnabledForChildActions**: 用于为子MVC action开启或关闭审计。默认为false。

**Selectors**: 用于选择其他的类保存审计日志。

**Selectors**是选择其他的类型来保存审计日志的谓词列表。如：

Configuration.Auditing.Selectors.Add(

new NamedTypeSelector(

"Abp.ApplicationServices",

type => typeof (IApplicationService).IsAssignableFrom(type)

)

);

你可以将你的选择器添加到模块的PreInitialize方法中。而且，如果你不喜欢为应用服务保存审计日志，那么你可以通过名称移除选择器。

### **通过特性开启/关闭**

虽然可以通过配置选择审计类，但是也可以为一个单独的类或方法使用**Audited**和 **DisableAuditing**特性。举个例子：

[Audited]

public class MyClass

{

public void MyMethod1(int a)

{

//...

}

[DisableAuditing]

public void MyMethod2(string b)

{

//...

}

public void MyMethod3(int a, int b)

{

//...

}

}

除了MyMethod2方法之外，MyClass的所有方法都参与审计了，因为MyMethod2显式关闭了。审计特性可以用于想要参与审计的方法保存审计日志。

### **注意**

为了保存审计日志，方法必须是**public**的。私有的和受保护的方法会被忽略。

如果方法是通过类的引用调用的，那么该方法必须是**virtual**的。

### **我项目中的例子**

下面的代码是点击登录按钮提交表单数据的action方法：

[HttpPost]

[UnitOfWork]

[DisableAuditing]

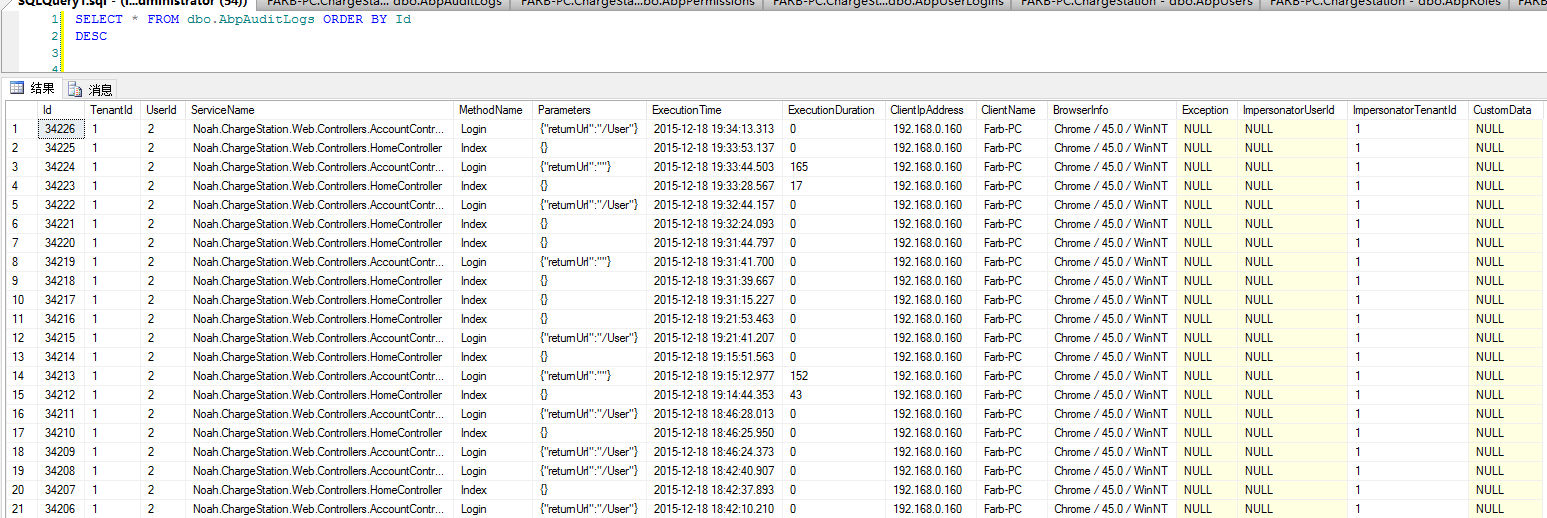
public virtual async Task<JsonResult> Login(LoginViewModel loginModel, string returnUrl = "")

{

...

}

下面的截图是数据库中的审计日志信息，并没有Login方法的审计日志



下面我将上面的代码的特性**[DisableAuditing]**去掉，再次登录系统，可以看到有了审计记录如下：

