一种截取过滤器实现方案的应用研究

陈 刚
（内江职业技术学院 信息电子工程系，四川 内江 641100）

摘 要：ASP.NET MVC 具有优秀的灵活性和可扩展性，但其内置特征并没有实现 Web 应用中一些常见功能。要较好地解决这个问题，需遵循软件工程的基本原则并配合使用一定的设计模式。为此，提出一种基于 ASP.NET MVC3 的可配置全局动态截取过滤器实现方案，并结合实例给出该方案在优化 ASP.NET MVC 架构安全性能上的应用。

关键词：ASP.NET MVC；截取过滤器模式；安全性能
中图分类号：TP311.1 文献标识码：A 文章编号：1674-7720(2012)19-0005-03

The application research about implementation scheme of intercepting filter pattern

Chen Gang
(Department of Information & Electronic Engineering, Neijiang Vocational & Technical College, Neijiang 641100, China)

Abstract: ASP.NET MVC has good flexibility and extensibility, but it's built-in features don't realize some usual functions in the web application. To solve the problems, should follow the basic principles of software engineering and cooperate with using a design pattern. For this reason, this paper puts forward a kind of implementation scheme of intercepting filter pattern based on ASP.NET MVC3. Then, combining with concrete example, demonstrates the application of this implementation scheme in optimizing the safety performance of ASP.NET MVC framework.

Key words: ASP.NET MVC; intercepting filter pattern; safety performance

基于 Web 程序提供的产品和服务，使信息产业已渗透到人们日常生活、学习和工作的方方面面，随之而来的是 Web 程序也日益成为恶意攻击的目标。当前 Web 应用的开发都是基于某一软件体系架构，与别的开发架构一样，ASP.NET MVC 的内置特征并不能对常见的网络安全问题给予足够的支持。为了更好地解决这个问题，同时遵循软件工程的一些基本原则，(如分隔原则、构件原则以及复用原则等)，需要配合一定的设计模式，以提高软件产品的质量。

1 ASP.NET MVC 架构

1.1 ASP.NET MVC 架构原理

微软自 2009 年推出 ASP.NET MVC1 以来，到目前为止已推出第 3 个正式版本，即 ASP.NET MVC3。ASP.NET MVC 将 Web 应用程序的输入逻辑、业务逻辑和用户界面逻辑 3 个不同的方面进行隔离，从而将一个 MVC 程序分为 3 个部分：模型(Model)、视图(View)和控制器(Controller)。每一部分都定义良好且是自包含的，即：与数据操作相关的逻辑仅包含在 Model 之中；与数据显示相关的逻辑仅包含在 View 中；处理用户请求和输入仅包含在 Controller 中。Model、View 和 Controller 描述如下：

(1)Model：是对应用领域的定义，它可以是简单的 view model，也可以是 domain model。view models 只是描述在 View 和 Controller 之间传递的数据；而 domain model 则包含业务逻辑和数据操作规则。Model 的设计是 MVC 的核心。

(2)View：代表应用程序的用户界面，显示数据。

(3)Controller：是 View 和 Model 之间的胶合剂，它处理请求，执行 Model 上定义的操作，选择 View 显示给用户。

在处理 HTTP 请求时，Controller、Model 和 View 之间的相互作用如图 1[1]所示。

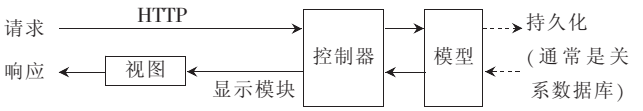

图 1 控制器、模型和视图之间的相互作用

1.2 ASP.NET MVC 架构存在的问题

ASP.NET MVC3 具有非常优秀的灵活性和可扩展性，但其内置特性对基于 MVC 架构的应用只提供了基础的支持，许多在实际应用中需要的功能 ASP.NET MVC3 并没有实现，只是留下一些接口供开发者根据具体的应用环境加以实施。这其中比较重要的一点就是 Web 程序的安全性能。因为随着信息技术的迅猛发展，人们日常的学习、生活和工作越来越依赖于 Web 程序所提供的服务，与此同时 Web 程序也日益成为恶意攻击的目标。所以，如何在基于 MVC 架构的应用中解决 Web 程序的安全风险就成为了系统开发人员必须面对和解决的问题。据 OWASP(Open Web Application Security Project)提供的数据显示，目前排在前两位的 Web 程序安全风险是注入和 CSS(Cross-Site Scripting)[2]，要较为完善地解决这个问题需要遵循软件工程的一些基本原则，并配合使用一定的设计模式，才能开发出质量较高的应用系统。

2 基于 ASP.NET MVC3 的截取过滤器

2.1 截取过滤器模式概述

截取过滤器模式是 Web 应用程序专用的，表示层的设计模式表达的是表示层的请求处理机制，它工作在应用程序级别上，对行为和信息流而非对象进行处理。截取过滤器模式的核心思想是用一个简单的机制实现不同过滤动作的处理组件的添加和删除，通过创建可插拔的过滤器，以标准的方式提供各种服务，而不需要修改核心的请求处理代码。定义在这些过滤器中的方法被 ASP.NET MVC 的内置代码解析出来挂接到 MVC 请求处理管道的恰当的位置上，并在各自的环节上截取 Web 请求，对请求信息进行分析、修改或重定向，从而为 Web 请求的处理提供诸如安全、登录、加密等服务。

截取过滤器有如下的优点[3]：

(1)关注分离：由于过滤器中的逻辑和应用程序的逻辑是去耦合的，因此当底层特性改变时，应用程序的代码不受影响。

(2)灵活性：由于过滤器之间是相互独立的，因此可以任意地组合这些过滤器而不需要改变过滤器的代码。

(3)中心化配置：由于过滤器的可组合性，可以使用单一的配置文件来加载过滤器链，也可以修改这个配置文件来决定处理请求的过滤器链的组成。

(4)可复用性：因为过滤器不依赖于它们操作的环境，因此这些过滤器能够在别的 Web 程序中被重用。

(5)部署时的兼容性：由于过滤器链能够在运行时基于配置文件构建出来，因此可以在部署期间改变过滤器的顺序而不用修改代码。

2.2 截取过滤器实现方案

考虑到截取过滤器在 ASP.NET WebForm 架构中的实现，为保持一致性和连贯性，本文针对 ASP.NET MVC3 提出一种基于配置文件的全局动态截取过滤器实现方案。该方案中的任何一个全局动态过滤器是否应用于某个 Controller 或 Action 取决于该过滤器在配置文件中的配置情况，并使用库类 FilterProviders 注册过滤器。该方案简要描述如下：

各过滤器独立编码，并编译为独立的程序集，并在配置文件中配置各过滤器要应用到哪些 Controller 或 Action；然后，使用类 FiltersConfiguration 的 GetAllFilters()方法读取配置文件中各个过滤器的配置信息，并用这些配置信息构建一个泛型列表 List<FilterNode>，接着使用类 FilterProvider 的 GetFilters()对泛型列表 List<FilterNode>中的 FilterNode 对象进行实例化；最后在 Global.cs 文件的 RegisterGlobalFilters 方法中使用 FilterProviders 注册过滤器。

该方案中各个部分描述如下：

(1)过滤器：过滤器继承抽象类 FilterAttribute 并实现接口 IActionFilter、IExceptionFilter、IResultFilter、IAuthorizationFilter 中的一个或多个。过滤器及其使用情况在配置文件中进行配置，配置结构如下：

```
<GlobalFilter>
  <Allfilters>
    <filter name="NameSpace.ClassName" assembly="
                                        assemblyName">
      <element controller="ControllerName" action="
                                          actionName"/>
    </filter>
  </Allfilters>
</GlobalFilter>
```

其中，节点 filter 用于配置一个过滤器；节点 element 用于设置该过滤器需要应用的 Controller 和 Action，若要应用到所有的 Controller 和 Action，则需将该节点属性 controller 和 action 都设置为字符“*”；节点 Allfilters 可以包含 0 个或多个 filter 节点，每个 filter 节点可以包含 1 个或多个 element 节点。

(2)读取配置信息：使用自定义类 FilterNode、Element、FiltersConfiguration 来读取配置信息。FilterNode 对应于 filter 节点，用于存放配置文件中每个过滤器的配置信息，相应地该类的成员有：属性 Name、Assembly 以及 Elements。Name 对应于 filter 节点的 Name 属性，Assembly 对应于 filter 节点的 assembly 属性，Elements 包含 filter 节点所有 element 子节点的配置信息。Element 类对应于 element 节点，包含属性 Controller 和 Action，分别对应于 element 节点的 controller 和 action 属性。FiltersConfiguration 类读取配置文件中过滤器的配置信息，其方法是用 GetAllFilters()实现该功能，并返回一个由过滤器配置信息

构建的一个泛型列表 List<FilterNode>。

(3)过滤器提供器:自定义类 FilterProvider,该类实现 IFilterProvider 接口,用于实例化过滤器,包含有:属性 Assembly 和 Name、字段 actions 以及方法 Add 和 GetFilters。Assembly 和 Name 分别对应于配置文件中各过滤器的程序集名称和类名;字段 actions 包含当前要实例化的过滤器在配置文件中 element 节点的信息,是一个 ControllerAction 实例的泛型列表。自定义类 ControllerAction,包含属性 Controller 和 Action,对应于 element 节点的配置信息;Add 方法用于向 actions 字段添加 element 节点的信息,构建泛型列表 List<ControllerAction>;GetFilters 方法用于实例化过滤器。

(4)注册过滤器:在 Global.cs 文件的RegisterGlobalFilters 方法中调用 FiltersConfiguration 的 GetAllFilters 方法读取各过滤器的配置信息。针对每一个过滤器,使用一个 FilterProvider 实例来处理,并将其添加到 MVC FilterProvider collection 中。

采用了本截取过滤器实现方案的系统的应用模型如图 2 所示。

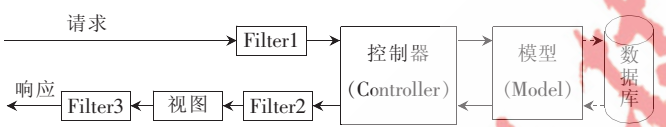

图 2 本截取过滤器实现方案的应用模型

说明如下:

(1)Filter1 可以是 Authorization filter 或 Action filter,Filter2 可以是 Action filter 或 Result filter,Filter3 可以是 Result filter 或 Exception filter。

(2)Filter1、Filter2、Filter3 中任何一个都可以代表 0 个或多个过滤器。

3 截取过滤器实现方案的应用

本实例以某一在线购书网站来说明如何利用本文提出的基于 ASP.NET MVC3 的截取过滤器实现方案消除 Web 程序的注入风险(包括 SQL 注入和 JavaScript 注入)。该系统提供常见的在线购物功能,其中的一些动态内容存在注入风险,如:注册、会员登录、库存搜索、服务质量评价和书籍缺货预订等,由于这些风险跨越多个 Controller,因此可以考虑用本文提出的截取过滤器实现方案来解决这个问题,即:针对 SQL 注入风险定义过滤器类 SqlInjectionFilterAttribute,针对 JavaScript 注入风险定义过滤器类 JavaScriptFilterAttribute。这两个类均继承自 FilterAttribute 并分别实现 IActionFilter.OnActionExecuting 方法和 IResultFilter.OnResultExecuted 方法。限于篇幅,具体的编码不能详述,这里仅对这两个方法的实现算法做一扼要说明。

(1)SqlInjectionFilterAttribute.OnActionExecuting 方法

本方法对用户输入的内容进行过滤,以有效减少 SQL 注入的发生,对输入数据中有潜在威胁的字符,如单引号、连接符、SQL 关键字等分别进行处理:将所有单独出现的单引号改成两个单引号;删除连字符;对含有诸如 exec、insert、select、delete、from、update 等 SQL 关键字的输入内容,不予响应。

(2)JavaScriptFilterAttribute.OnResultExecuted 方法[4]

在本方法中对用户输入字符串中包含的一些特殊字符进行编码,如:字符“<”被转换成 <,字符“>”转换成 >,字符“&”被转换成 &等,从而消除 JavaScript 注入风险。

这两个过滤器在 Web.config 中的配置如下(限于篇幅做了部分省略):

```
<GlobalFilter>
 <Allfilters>
  <filter name="BookStore.Filters.
   SqlInjectionFilterAttribute" assembly="assemblyName">
    <element controller="AccountController"action="*"/>
     <element ……/>      //多个其他存在注入风险的
                         //Controller 和 Action,省略
  </filter>
   <filter name="BookStore.Filters.JavaScriptFilterAttribute"
assembly="assemblyName">
    <element controller="AccountController" action="*"/>
     <element ……/>      //多个其他存在注入风险的
                         //Controller 和 Action,省略
   </filter>
  </Allfilters>
</GlobalFilter>
```

在使用 ASP.NET MVC 架构进行应用系统开发中有效地使用截取过滤器模式,可以使在遵循软件工程的一些基本原则的情况下,较好地解决一些 Web 应用中常见的问题。本文针对 ASP.NET MVC3 提出一种可配的全局动态截取过滤器实现方案,并应用该方案来优化 ASP.NET MVC 架构的安全性能,体现了软件工程中分隔原则、构件原则以及复用原则等一些基本准则,提高了代码的复用性、增强了系统的灵活性,并使得程序的扩展和维护变得更加容易。

参考文献

[1] FREEMAN A, SANDERSON S. Pro ASP.NET MVC3 framework[M]. Apress press,2011.

[2] OWASP. Category: OWASP top ten project[EB/OL]. (2012-01-18). https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project.

[3] Microsoft. Intercepting filter [EB/OL].[2012-05-01]. http://msdn.microsoft.com/en-us/library/ff647251.aspx.

[4] SALEM A. Intercepting filter approach to injection flaws[J]. Journal of Information Processing Systems.,2010,12(4):563-574.

(收稿日期:2012-05-01)

作者简介:

陈刚,男,1969 年生,硕士,工程师,主要研究方向:.NET 及 XML 数据库。