

[百度一下](#)

您查询的关键词是：**web config** 设置 缓存 以下是该网页在北京时间 2016年12月15日 22:51:14 的快照：

如果打开速度慢，可以尝试[快速版](#)；如果想更新或删除快照，可以[投诉快照](#)。

百度和网页 <http://www.myexception.cn/web/1600809.html> 的作者无关，不对其内容负责。百度快照谨为网络故障时之索引，不代表被搜索网站的即时页面。

当前位置:我的异常网» **Web**前端 » WebConfig的那些事情

## WebConfig的那些事情

www.MyException.Cn 网友分享于：2015-08-26 浏览：26次

### WebConfig的那些事儿

配置文件的文件后缀一般是**.config**，在asp.net中配置文件名一般默认是**web.config**。每个**web.config**文件都是基于XML的文本文件，并且可以保存到**Web**应用程序中的任何目录中。在发布**Web**应用程序时**web.config**文件并不编译进dll文件中。如果将来客户端发生了变化，仅仅需要用记事本打开**web.config**文件编辑相关设置就可以重新正常使用，非常方便。现在来说配置文件的查找优先级、配置文件节点说明

#### 配置文件的查找优先级

在.net提供了一个针对当前机器的配置文件，这个文件是**machine.config**，它位于一般位于C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config文件下，**machine.config**文件的内容：在这个文件夹下还有一个**web.config**文件，这个文件包含了asp.net网站的常用配置。下面是这个**web.config**文件的内容：

asp.net网站IIS启动的时候会加载配置文件中的配置信息，然后缓存这些信息，这样就不必每次去读取配置文件信息。在运行过程中asp.net应用程序会监视配置文件的变化情况，一旦编辑了这些配置信息，就会重新读取这些配置信息并缓存。当我们要读取某个节点或者节点组信息时，是按照如下方式搜索的：

(1) 如果在当前页面所在目录下存在**web.config**文件，查看是否存在所要查找的结点名称，如果存在返回结果并停止查找。

(2) 如果当前页面所在目录下不存在**web.config**文件或者**web.config**文件中不存在该结点名，则查找它的上级目录，直到网站的根目录。

(3) 如果网站根目录下不存在**web.config**文件或者**web.config**文件中不存在该节点名则

在C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\**web.config**文件中查找。

(4) 如果

在C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\**web.config**文件中不存在相应结点，则

在C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Config\**machine.config**文件中查找。

(5) 如果仍然没有找到则返回null。

所以如果我们对某个网站或者某个文件夹有特定要求的配置，可以在相应的文件夹下创建一个**web.config**文件，覆盖掉上级文件夹中的**web.config**文件中的同名配置即可。这些配置信息的寻找只查找一次，以后便被缓存起来供后来的调用。在asp.net应用程序运行过程中，如果**web.config**文件发生更改就会导致相应的应用程序重新启动，这时存储在服务器内存中的用户会话信息就会丢失（如存储在内存中的Session）。一些软件（如杀毒软件）每次完成对**web.config**的访问时就会修改**web.config**的访问时间属性，也会导致asp.net应用程序的重启。

#### 配置文件节点说明

**web.config**文件是一个XML文件，它的根结点是<configuration>，在<configuration>节点下的常见子节点有：<configSections>、<appSettings>、

<connectionStrings>和<system.web>。其中<appSettings>节点主要用于配置一些网站的应用配置信息，而<connectionStrings>节点主要用于配置网站的数据数据库连接字符串信息。

<system.web>节点主要是网站运行时的一些配置，它的常见节点有如下：

<appSettings>节点

<appSettings>节点主要用来存储asp.net应用程序的一些配置信息，比如上传文件的保存路径等，以下是一个例子：

```
<appSettings>
  <!--允许上传的图片格式类型-->
  <addkey="ImageType" value=".jpg;.bmp;.gif;.png;.jpeg"/>
  <!--允许上传的文件类型-->
  <addkey="FileType" value=".jpg;.bmp;.gif;.png;.jpeg;.pdf;.zip;.rar;.xls;.doc"/>
</appSettings>
```

对于<appSettings>节点中的值可以按照key来进行访问，以下就是一个读取key值为“FileType”节点值的例子：

```
string fileType=ConfigurationManager.AppSettings["FileType"];
```

2.2 <connectionStrings>节点

<connectionStrings>节点主要用于配置数据库连接的。我们可以<connectionStrings>节点中增加任意个节点来保存数据库连接字符串，将来在代码中通过代码的方式动态获取节点的值来实例化数据库连接对象，这样一旦部署的时候数据库连接信息发生变化我们仅需要更改此处的配置即可，而不必因为数据库连接信息的变化而需要改动程序代码和重新部署。

以下就是一个<connectionStrings>节点配置的例子：

```
<connectionStrings>
  <!--SQL Server数据库配置-->
  <addname="BasicDataSystem" connectionString="DataSource=(local);Initial Catalog=AspNetStudy;User ID=sa;Password=123456"/>
</connectionStrings>
```

在代码中我们可以这么实例化数据库连接对象：

```
//读取web.config节点配置
string connectionString = ConfigurationManager.ConnectionStrings["AspNetStudyConnectionString1"].ConnectionString;
//实例化SqlConnection对象
SqlConnection connection = new SqlConnection(connectionString);
```

这样做的好处是一旦开发时所用的数据库和部署时的数据库不一致，仅仅需要用记事本之类的文本编辑工具编辑connectionString属性的值就行了。

<compilation>节点

<compilation>节点配置 ASP.NET 使用的所有编译设置。默认的debug属性为“true”，即允许调试，在这种情况下会影响网站的性能，所以在程序编译完成交付使用之后应将其设为“false”。

<authentication>节点

设置asp.net身份验证模式，有四种身份验证模式，它们的值分别如下：

Mode	说明
Windows	使用Windows身份验证，适用于域用户或者局域网用户。
Forms	使用表单验证，依靠网站开发人员进行身份验证。
Passport	使用微软提供的身份验证服务进行身份验证。
None	不进行任何身份验证。

<authentication>节点控制用户对网站、目录或者单独页的访问，必须配合<authentication>节点一起使用。

<customErrors>节点

<customErrors>节点用于定义一些自定义错误信息的信息。此节点有Mode和defaultRedirect两个属性，其中defaultRedirect属性是一个可选属性，表示应用

程序发生错误时重定向到的默认URL，如果没有指定该属性则显示一般性错误。  
**Mode**属性是一个必选属性，它有三个可能值，它们所代表的意义分别如下：

Mode	说明
On	表示在本地和远程用户都会看到自定义错误信息。
Off	禁用自定义错误信息，本地和远程用户都会看到详细的错误信息。
RemoteOnly	表示本地用户将看到详细错误信息，而远程用户将会看到自定义错误信息。

这里有必要说明一下本地用户和远程用户的概念。当我们访问asp.net应用程序时所使用的机器和发布asp.net应用程序所使用的机器为同一台机器时成为本地用户，反之则称之为远程用户。在开发调试阶段为了便于查找错误**Mode**属性建议设置为**Off**，而在部署阶段应将**Mode**属性设置为**On**或者**RemoteOnly**，以避免这些详细的错误信息暴露了程序代码细节从而引来黑客的入侵。

下面我们添加一个页面CustomErrorsDemo.aspx，在它的Page\_Load事件里抛出一个异常，代码如下：

```
public partial class CustomErrorsDemo : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        throw new Exception("故意抛出的异常。");
    }
}
```

配置<customErrors>如下：

```
<customErrorsmode="RemoteOnly">
    <errorstatusCode="403"redirect="NoAccess.htm" />
    <errorstatusCode="404"redirect="FileNotFound.htm" />
</customErrors>
```

这时本地运行CustomErrorsDemo.aspx的效果如下：

*故意抛出的异常。*

说明: 执行当前 Web 请求期间，出现未经处理的异常。请检查堆栈跟踪信息，以了解有关该错误以及代码中导致错误的出处的详细信息。


异常详细信息: System.Exception: 故意抛出的异常。


源错误:

http://blog.csdn.net/dandanzmc

```
行 13:      {
行 14:          {
行 15:              throw new Exception("故意抛出的异常。");
行 16:          }
行 17:      }
```

远程访问看到结果：

如果我们将customErrors的Mode属性设置为“On”本地运行和远程访问都会看到如下效果：

如果将customErrors的Mode属性设置为“Off”本地运行和远程访问都会看到如下效果：

### 故意抛出的异常。

说明: 执行当前 Web 请求期间, 出现未经处理的异常。请检查堆栈跟踪信息, 以了解有关该错误以及代码中导致错误的出处的详细信息。

异常详细信息: System.Exception 故意抛出的异常。

源错误:

```

http://blog.csdn.net/dandanzmc
行 13:      {
行 14:          {
行 15:              throw new Exception("故意抛出的异常。");
行 16:          }
行 17:      }

```

### <error>子节点

在<customErrors>节点下还包含有<error>子节点, 这个节点主要是根据服务器的HTTP错误状态代码而重定向到我们自定义的错误页面, 注意要使<error>子节点下的配置生效, 必须将<customErrors>节点节点的Mode属性设置为"On"。下面是一个例子:

```

<customErrorsmode="On"defaultRedirect="GenericErrorPage.htm">
  <errorstatusCode="403"redirect="403.htm" />
  <errorstatusCode="404"redirect="404.htm" />
</customErrors>

```

在上面的配置中如果用户访问的页面不存在就会跳转到404.htm页面, 如果用户没有权限访问请求的页面则会跳转到403.htm页面, 403.htm和404.htm页面都是我们自己添加的页面, 我们可以在页面中给出友好的错误提示。

### <httpHandlers>节点

<httpHandlers>节点用于根据用户请求的URL和HTTP谓词将用户的请求交给相应的处理程序。可以在配置级别的任何层次配置此节点, 也就是说可以针对某个特定目录下指定的特殊文件进行特殊处理。

下面我们以一个例子来说明<httpHandlers>节点的法, 在我们的asp.net应用程序中建立一个IPData目录, 在IPData目录中创建一个IPData.txt文件, 然后在Web.config中添加以下配置:

```

<httpHandlers>
  <add path="IPData/*.txt" verb="*" type="System.Web.HttpForbiddenHandler"/>
</httpHandlers>

```

上面的代码的作用是禁止访问IPData目录下的任何txt文件。

然后新建一个页面, 在页面中添加一个超级链接, 链接到该目录下IPData.txt文件, 代码如下:

```

<%@PageLanguage="C#"AutoEventWireup="true"CodeFile="HttpHandlersDemo.aspx.cs" Inherits="HttpHandlersDemo"%>
<!DOCTYPEhtmlPUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1"runat="server">
  <title>httpHandlers节点的例子</title>
</head>
<body>
  <form id="form1"runat="server">
    <div>
      <a href="IPData/IPData.txt"title="打开IPData/IPData.txt">打开IPData/IPData.txt</a>
    </div>
  </form>
</body>
</html>

```

运行这个页面的效果如下:



当前web.config文件的<customErrors>节点配置如下:

```
<customErrorsmode="On" defaultRedirect="GenericErrorPage.htm">
  <errorstatusCode="403" redirect="403.htm" />
  <errorstatusCode="404" redirect="404.htm" />
</customErrors>
```

如果存在403.htm和404.htm页面，点击超级链接之后会出现如下效果：



我们从上图中可以看到当<customErrors>节点的Mode属性为“On”时，因为被禁止访问IPData文件夹下的所有txt文件，所以会跳转到自定义的没有权限提示页面，即403.htm。

**<httpRuntime> 节点**

<httpRuntime>节点用于对 ASP.NET HTTP 运行库设置。该节可以在计算机、站点、应用程序和子目录级别声明。

例如下面的配置控制用户最大能上传的文件为40M（40\*1024K），最大超时时间为60秒，最大并发请求为100个。

```
<httpRuntimemaxLength="40960" executionTimeout="60" appRequestQueueLimit="100"/>
```

**<pages> 节点**

<pages>节点用于表示对特定页设置，主要有三个属性，分别如下：

属性名	说明
buffer	是否启用了 HTTP 响应缓冲。
enableViewStateMac	是否应该对页的视图状态运行计算机身份验证检查 (MAC)，以放置用户篡改，默认为false，如果设置为true将会引起性能的降低。
validateRequest	是否验证用户输入中有跨站点脚本攻击和SQL注入式漏洞攻击，默认为true，如果出现匹配情况就会发HttpRequestValidationException 异常。对于包含有在线文本编辑器页面一般自行验证用户输入而将此属性设为false。

下面就是一个配置节点的例子：

```
<pagesbuffer="true" enableViewStateMac="true" validateRequest="false"/>
```

**<sessionState> 节点**

<sessionState>节点用于配置当前asp.net应用程序的会话状态配置。以下就是一个常见配置：

```
<sessionStatecookieless="false" mode="InProc" timeout="30" />
```

上面的节点配置是设置在asp.net应用程序中启用Cookie，并且指定会话状态模式为在进程中保存会话状态，同时还指定了会话超时为30分钟。

<sessionState>节点的Mode属性可以是以下几种值之一：

属性名	说明
Custom	使用自定义数据来存储会话状态数据。
InProc	默认值。由asp.net辅助进程来存储会话状态数据。
Off	禁用会话状态。
SQLServer	使用进程外SQL Server数据库保存会话状态数据。
StateServer	使用进程外 ASP.NET 状态服务存储状态信息。

一般默认情况下使用InProc模式来存储会话状态数据，这种模式的好处是存取速度快，缺点是比较占用内存，所以不宜在这种模式下存储大型的用户会话数据。

**<globalization> 节点**

用于配置应用程序的全球化设置。此节点有几个比较重要的属性，分别如下：

属性名	说明
fileEncoding	可选属性。设置.aspx、.asmx 和 .asax 文件的存储编码。
requestEncoding	可选属性。设置客户端请求的编码，默认为UTF-8.
responseEncoding	可选属性。设置服务器端响应的编码，默认为UTF-8.

以下就是asp.net应用程序中的默认配置：

```
<globalizationfileEncoding="utf-8"requestEncoding="utf-8"responseEncoding="utf-8"/>
```

**web.config**是asp.net应用程序中一个很重要的配置文件，通过**web.config**文件可以方便我们进行开发和部署asp.net应用程序。此外还能对程序进行一些灵活的控制。原来并不知道**web.config**还有这么大的学问，通过 前几天遭遇的意见事情，让我不得不重新拾起来，人在江湖混久了，迟早都会还的。

文章评论