**配置文件 config**

awdawd

**配置文件的作用和注意事项**

1. **主要作用：**

-> 配置文件都是基于XML的文本文件，主要作用于存储一些应用程序在功能实现上所需要的一些信息或是设置应用程序在运行时所需要的一些配置等等

-> 配置文件的作用域于当前所运行应用程序的目录及虚拟子目录下

-> 当我们想要读取配置文件中某个节点或节点组的信息的时候，他是按以下步骤来进行查找的

1. 如果当前页面所在目录下存在config文件，查看是否存在所查找的节点名称，如果存在则返回结果并停止查找
2. 如果当前页面所在目录下不存在config文件或config文件中不存在所需要查找的节点，则会在当前页面的上级目录进行查找，直至网站的根目录
3. 如果网站根目录下不存在config文件或者config文件中不存在该节点名的节点，则在文件中查找
4. **注意事项：**

*-> App.config [Windows Appliction]*

1. 在Windows应用程序开发当中，项目所默认添加的配置文件名则为App.config
2. App.config对于Windows应用程序开发主要作用于数据库连接字符串的获取和配置、存储应用程序需要记录且用到的一些信息等，主要使用到的功能并不是很多

*-> Web.config [Web Application]*

1. 在Web应用程序开发当中，项目所默认添加的配置文件名则为Web.config
2. Web应用程序在启动的时候，会自动缓存下来配置文件上可正确读取到的信息，其主要目的是为了防止后期在应用程序运行的过程当中对配置文件频繁重复的进行读取，但是一些重要的信息（数据库连接字符串、身份验证信息、错误信息等）只会在开始启动Web应用程序的时候才会被读取缓存起来，所以，为了Web应用程序在发布后的一些后续使用中能够正确读取且应用到我们手动去配置的一些节点信息，Web应用程序在运行的过程当中会时刻监听着当前所运行的应用程序所使用的Web.config，一旦里面的信息发生了改变的话，Web应用程序的工作进程(IIS 6.0 : w3wp.exe / IIS 5.0 : aspnet\_wp.exe)则会自动的重启一次，以便能够正确的读取且应用到我们刚刚所配置的一些节点信息。**需要注意的是**，一些软件（杀毒软件）每完成一次对Web.config的访问则会修改一次Web.config的访问时间，这样也会造成Web应用程序的重启，如果在此之前Web应用程序缓存了一些用户的临时数据，则会造成丢失，但是我们可以通过设置Web.config上指定节点的 [configSource] 属性来让当前节点的信息指向我们所指定的另一个配置文件来作为维护，通过这样的操作后，当我们修改了所指向的配置文件上的一些信息也不会造成Web应用程序的重启了（所修改的配置文件并不是当前运行Web应用程序所使用的配置文件），尽量的避免Web应用程序频繁的重启，具体使用详情可以参考以下网址：

<https://dotblogs.com.tw/puma/2010/01/25/aspnet-connectionstring-appsetting-file-configsource>

**配置文件的具体使用**

1. **<connectionString>节点**

-> 该节点主要用于配置数据库连接的，我们可以在当前节点中增加任意个数的子节点<add>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

-> 当我们在配置文件中配置好了连接字符串后，需要手动的引入名为 [System.Configuration] 的程序集，并加载相应的命名空间后才可以在c#代码中获取刚刚配置好的数据库连接字符串

-> c#代码：

1. 获取数据库连接字符串

微信截图_20190326225327

-> 子节点：

1. <add>：添加信息操作
2. <add **name** = "value" />：该属性用于设置获取数据库连接字符串的键值
3. <add **connectionString** = "sqlconnectionString" />：该属性用于设置数据库连接的字符串

-> 示例代码：



1. **<appSettings>节点**

-> 该节点主要用于存储一些在应用程序中有关于业务上可以用到的一些配置信息，比如：打印机名称、权限、……，节点上的信息以 key & value 的方式来进行存储和查找

-> 当我们在配置文件中配置好了应用程序配置信息后，需要手动的引入名为 [System.Configuration] 的程序集，并加载相应的命名空间后才可以在c#代码中通过指定的key来获取刚刚配置好的应用程序配置信息

-> c#代码：

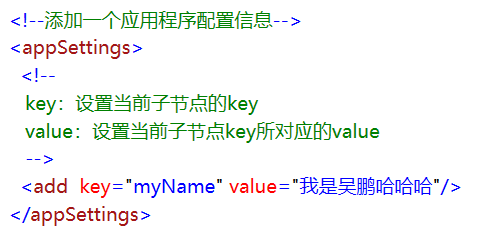
1. 获取应用程序配置信息

微信截图_20190326230907

-> 子节点：

1. <add>：添加信息操作
2. <add **key** = "keyName" />：该属性用于设置当前子节点的key
3. <add **value** = "values" />：该属性用于设置当前子节点key所对应的value

-> 示例代码：



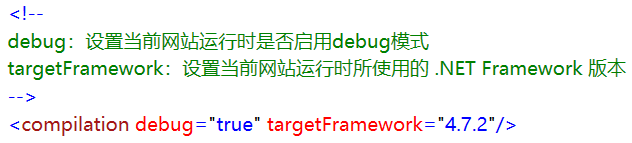
1. **<system.web>节点**

-> 该节点主要用于设置ASP.NET网站在运行时所需要的一些必要的、全局的配置

-> 子节点：

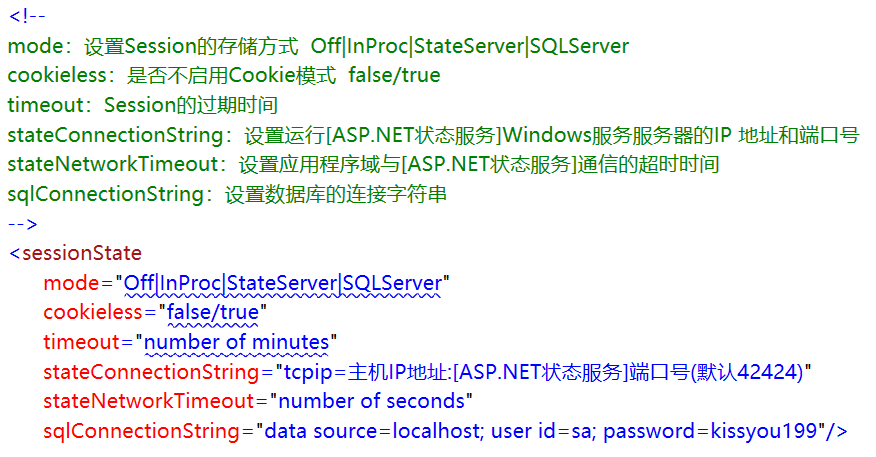
1. <compilation>：控制网站的一些全局性的配置
2. <compilation **debug** = "boolen" />：该属性用于设置ASP.NET网站在运行的时候是否启用debug模式，需要我们 注意的是，当我们把网站上线部署至生产环境的时候，该属性的值一定要设置为 [false]，因为在debug模式下去访问网 站是一件非常消耗性能和资源的事情，网站打开速度可能因此变得很慢
3. <compilation **targetFramework** = "version" />：该属性用于设置ASP.NET网站所使用的 .NET Framework 版本

-> 示例代码：



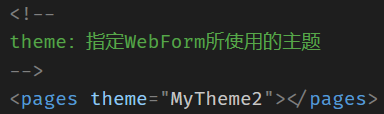
1. <sessionState>：控制Session的存储方式
2. <sessionState **mode** = "Off|InProc|StateServer|SQLServer" />：该属性用于设置Session的存储方式，需要注意的 是ASP.NET默认的Session存储方式为InProc
3. <sessionState **cookieless** = "true/false" />：该属性用于设置是否不启用Cookie模式进行传输 [SessionID] ，当 我们不启用该模式后，[SessionID] 则通过URL传达给客户端，反正则是通过Cookie的方式把 [SessionID] 传给客户端
4. <sessionState **timeout** = "number of minutes" />：该属性用于设置Session存储在服务器上的过期时间，当然这 个设置是针对当前应用程序域全局Session的，我们也可以通过C#代码设置指定某个Session的过期时间
5. <sessionState **stateConnectionString** = "tcpip=主机IP地址:[ASP.NET状态服务]端口号(默认42424)" />：属性为 使用 [StateServer] 模式下的必写属性，用于设置运行[ASP.NET状态服务]Windows服务服务器的IP 地址和端口号
6. <sessionState **stateNetworkTimeout** = "number of seconds" />：该属性为使用 [StateServer] 模式下的可选属 性，用于控制服务器在通过 [ASP.NET状态 服务] 进行Session存储的时候，最大操作时间，默认为10s
7. <sessionState **sqlConnectionString** = "connectionString" />：该属性为使用 [SQLServer] 模式下的必写属 性， 用于设置SqlServer的连接字符串
8. 关于更多的属性可以参考这篇文章：<https://www.cnblogs.com/SkySoot/archive/2012/07/13/2590117.html>

-> 示例代码：



1. <pages>：控制WebForm Page中的一些相关属性设置
2. <pages theme= "themeName" />：该属性用于指定WebForm所使用的主题

-> 示例代码

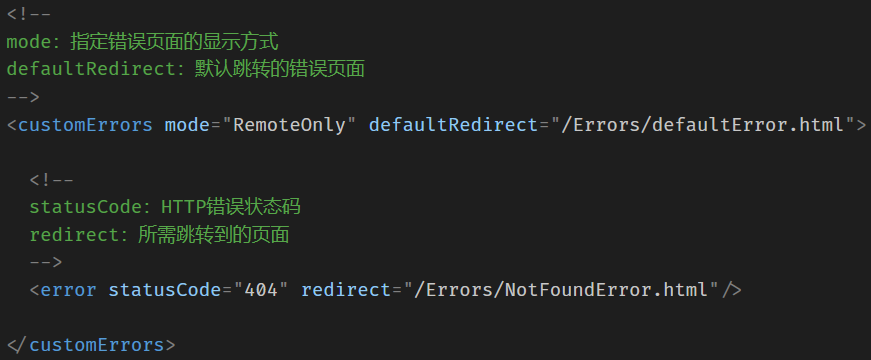


1. <customErrors>：设置客户端请求服务器资源的时候出现错误信息的显示方式
2. <customErrors mode= "On|Off|RemoteOnly" />：该属性用于指定错误页面的显示方式，On 总是根据自定义错误 页面配置进行跳转、Off 不启用自定义定制错误界面，显示回默认的堆栈报错信息、RemoteOnly 只有服务器主机访问该 主机的时候显示回默认的堆栈报错信息，远程主机则显示为该节点所自定义的错误页面
3. <customErrors defaultRedirect= "redirectUrl" />：当客户端请求服务器资源的时候所返回的对应HTTP错误状态码 并没有在子节点【error】中进行配置的话，则跳转到该默认错误页面

-> 子节点

1. <error>：设置特定HTTP错误状态码所需跳转的页面
2. <error statusCode= "errorHttpCode" />：所指定的特定HTTP错误状态码
3. <error redirect= "url" />：在特定的HTTP错误状态码下需要跳转到的页面

-> 示例代码



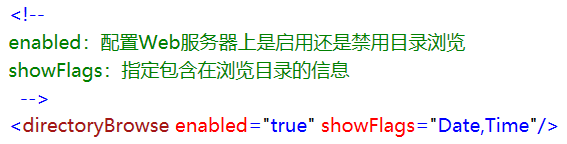
1. **<system.webServer>节点**

-> 该节点可以用于设置IIS7的站点级和应用程序级的一些配置，其中还包含了Web服务器引擎和模块使用的一些配置

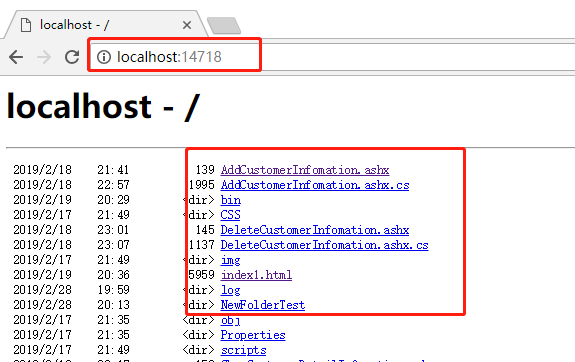
-> 子节点：

1. <directoryBrowse>：控制服务器端文件浏览方式
2. <directoryBrowse **enabled** = "boolen" />：该属性用于设置请求一个没有首页的域名的时候是否显示该项目文档中 的所有文件。一般我们再开发的时候，如果不想设置一个首页，那么我们就可以通过把该属性的值设置为[true]，这样当我 们在请求一个没有首页的域名的时候，IIS就会罗列出该项目文档中的所有文件，如果我们想查看哪个文件，点击一下即可 浏览，这样就省去了在浏览器的地址栏中输入请求文件名去发起请求的繁琐过程。
3. <directoryBrowse **showFlags** = "flags" />：该属性用于设置当启用了服务器目录浏览模式后，能够显示在浏览目录 上的信息，我们还可以通过使用逗号分隔来显示多种不同的信息在浏览目录上

-> 示例代码：



-> 效果演示：

s

1. <modules>：用于添加自定义的HttpModule

-> 子节点：

1. <add>：添加自定义HttpModule
2. <add **name**= "moduleName" />：该属性用于设置当前HttpModule的Name
3. <add **type**= "nameSpace" />：该属性用于设置当前HttpModule的命名空间

-> 示例代码：



1. <handlers>：用于添加自定义的HttpHandler

-> 子节点：

1. <add>：添加自定义HttpHandler
2. <add **name**= "handlerName" />：该属性用于设置当前HttpHandler的Name
3. <add **path**= "\*.path" />：该属性用于设置当前HttpHandler所处理的Resource Request
4. <add **verb**= "handlerVerb" />：该属性用于指定当前HttpHandler所支持的HTTP方法类型
5. <add **type**= "nameSpace" />：该属性用于设置当前HttpHandler的命名空间

-> 示例代码：

