Tomcat配置优化

作者：ky

减配优化

场景一：假设当前REST应用（微服务）

- 分析：它不需要静态资源，Tomcat容器静态和动态
  - 静态处理：DefaultServlet
    - 优化方案：通过移除conf/web.xml中org.apache.catalina.servlet.DefaultServlet
  - 动态处理：JspServlet
    - 优化方案：通过移除conf/web.xml中org.apache.jasper.servlet.JspServlet

    > DispatcherServlet:Spring Web MVC应用Servlet

    > JspServlet：编译并且执行Jsp页面 DefaultServlet：Tomcat处理静态资源的Servlet

  - 移除welcome-file-list

    ```
    <welcome-file-list>
      <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    ```

  - 如果程序是REST JSON Content或者MIME Type：application/json
  - 移除Session设置

    对于微服务/REST应用，不需要Session，因为不需要状态。

    Spring Security OAuth 2.0、 JWT

    Session通过jsessionId进行用户跟踪，HTTP无状态，需要一个ID与当前用户会话联系。
    Spring Session HttpSession jsessionId作为Redis，实现多个机器登录，用户会话不丢失。

    存储方法：Cookie、URL重写、SSL

  - 移除Value

    `Value`类似于`Filter`

    移除`AccessLogValue`,可以通过Nginx的AccessLog替代，`Value`实现都需要消耗Java应用的计算时间。

场景二：需要JSP的情况

- 分析：`JspServlet`无法移除，了解`JspServlet`处理原理

  > Servlet周期
  >
  > - 实例化：Servlet和Filter实现类必须包含默认构造器。反射的方式进行实例化

> - 初始化：Servlet容器调用Servlet或Filter init() 方法
> - 销毁：Servlet容器关闭时，Servlet或者Filter destroy() 方法被调用

Servlet或者Filter在一个容器中，是一般情况在一个Web App中是一个单例，不排除应用定义多个。

- JspServlet相关的优化`ServletConfig`参数：
  - 需要编译：
    - compiler
    - modificationTestInterval
  - 不需要编译：
    - development设置false

      development=false，那么这些jsp要如何编译：预编译。优化方案：

      - Ant Task执行JSP编译
      - Maven插件：org.codehaus.mojo:jspc-maven-plugin

      ```
      <dependency>
          <groupId>org.apache.sling</groupId>
          <artifactId>jspc-maven-plugin</artifactId>
          <version>2.1.0</version>
      </dependency>
      ```

      JSP->翻译.jsp或者.jspx文件成.java->编译.class

  总结，`conf/web.xml`作为Servlet应用的默认`web.xml`，实际上，应用程序存在两份`web.xml`，其中包括应用的`web.xml`，最终将两者合并。

JspServlet如果development参数为true，它会自定检查文件是否修改，如果修改重新翻译，再 编译（加载和执行）。言外之意，JspServlet开发模式可能会导致内存溢出，原因：卸载Class不及时所导致Perm区域不够。

```
>如何卸载class：
>
>ParentClassLoader 加载 1.class 2.class 3.class
>
>ChildClassLoader 加载 4.class 5.class
>
>1.class需要卸载，需要将ParentClassLoader设置为null,当ClassLoader被GC后，1-3class全部会被卸载。
```

配置调整

关闭自动重载

`context.xml`

```
<Context docBase="" reloadable="false"></Context>
```

修改连接线程池数量

`server.xml`

```
<Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
maxThreads="150" minSpareThreads="4"/>
<Connector executor="tomcatThreadPool" port="8080" protocol="HTTP/1.1"
connectionTimeout="20000" redirectPort="8443"/>
```

> 通过程序来理解，`<Executor>`实际的Tomcat接口：
>
> - `org.apache.catelina.Executor`
> - 扩展：J.U.C标准接口`java.util.concurrent.Executor`
> - 实现：`org.apache.catalina.core.StandardThreadExecutor`
>
> ```
> /**
>  * max number of threads,默认最大数量
>  */
> protected int maxThreads = 200;
>
> /*
>  * min number of threads
>  */
> protected int minSpaceThreads = 25;
> ```
>
> ```
> * 线程池：
> ```
>
> `org.apache.tomcat.util.threads.ThreadPoolExecutor`(`java.util.concurrent.ThreadPoolExecutor`)
>
> 总结：Tomcat IO连接器使用的线程池实际是标准的Java线程池的扩展，最大线程数量和最小线程数量实际上分别是MaximumPoolSize 和 CorePoolSize。