目录

[BS架构 1](#_Toc500907209)

[URL 2](#_Toc500907210)

[TCP， IP和HTTP 2](#_Toc500907211)

[HTTP协议 2](#_Toc500907212)

[HTTP数据包 2](#_Toc500907213)

[HTTP状态码 3](#_Toc500907214)

[组件、容器 3](#_Toc500907215)

[Servlet和JSP 3](#_Toc500907216)

[Servlet如何处理Http协议 3](#_Toc500907217)

[GET与POST请求方式的区别 3](#_Toc500907218)

[Servlet运行步骤 4](#_Toc500907219)

[重定向、转发 4](#_Toc500907220)

[绝对路径、相对路径 4](#_Toc500907221)

[Servlet的生命周期 4](#_Toc500907222)

[Servlet核心接口和抽象类 5](#_Toc500907223)

[Servlet线程安全问题 5](#_Toc500907224)

[Scriptlet 5](#_Toc500907225)

[JSP转换为Servlet过程中的编码 5](#_Toc500907226)

[JSP指令 5](#_Toc500907227)

[JSP九大内置对象（隐含对象） 6](#_Toc500907228)

[Cookie和Session 6](#_Toc500907229)

[web.xml 6](#_Toc500907230)

[Servlet过滤器 7](#_Toc500907231)

[Servlet监听器 7](#_Toc500907232)

[EL表达式 8](#_Toc500907233)

[JSTL 8](#_Toc500907234)

[自定义JSP标签 8](#_Toc500907235)

# HTTP数据包

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 请求数据包 | | | | |  |
| 请求行(request line) | 请求方式 | 请求路径 | 协议类型/版本 | |
| GET | …/sayHi | HTTP/1.1 | |
| 消息头（若干）(header) | 一些键值对，一般由w3c定义，大部分自动生成 | | | |
| 实体内容(body) | 只有当请求方式为post时，才有实体内容? | | | |
| 响应数据包 | | | | |  |
| 状态行(status line) | 协议类型/版本 | 状态码 | | 状态描述 |
| HTTP/1.1 | 200 | | OK |
| 消息头（若干）(header) | Web服务器返回消息 | | | |
| 实体内容 | 程序处理的结果 | | | |

# web.xml

|  |  |  |  |
| --- | --- | --- | --- |
| 部署描述符文件 | | | |
| web.xml加载过程 | | 启动一个WEB项目的时候，WEB容器(JBoss, Tomcat等)会去读取它的配置文件web.xml，读取<listener>和<context-param>两个节点 | |
| 紧接着，容器创建一个ServletContext(application)，这个web项目的所有部分都将共享这个上下文 | |
| 容器将<context-param>转换为键值对，存入servletContext | |
| 容器创建<listener>中的类实例，根据配置的class类路径<listener-class>来创建监听 | |
| 容器会读取<filter>，根据指定的类路径来实例化过滤器 | |
| 加载顺序 | | ServletContext -> context-param -> listener -> filter -> servlet | |
| 标签 | <web-app> | | 根元素 |
| <icon> | | 指出IDE和GUI工具用来表示Web应用的大图标和小图标；  <large-icon> <small-icon> |
| <display-name> | | 定义web应用的名称 |
| <disciption> | | Web应用描述，给出于此相关的说明性文本 |
| <context-param> | | 声明应用范围内的初始化参数。它用于向ServletContext提供键值对，我们的listener, filter等在初始化时会用到这些上下文中的信息。在servlet里面可以通过getServletContext().getInitParameter("param-name")得到。  <param-name> <param-value> |
| <filter>  <filter-mapping> | | 过滤器，将一个名字与一个实现javaxs.servlet.Filter接口的类相关联  <filter-name> <filter-class> <init-param> (<param-name> <param-value>)  <filter-name> <url-pattern> |
| <listener> | | 监听器 <listerner-class> |
| <servlet>  <servlet-mapping> | | <servlet-name> <servlet-class> <init-param> (<param-name> <param-value>)  <load-on-startup> <jsp-file>  <servlet-name> <url-pattern> |
| <session-config> | | 会话超时配置，单位分钟 <session-timeout> |
| <mime-mapping> | | [mime-mapping元素将mime类型映射到扩展名，用于规定下载格式](http://baike.baidu.com/view/1002818.htm)  <extension> <mime-type> |
| <welcome-file-list> | | 设置欢迎文件页 <welcome-file> |
| <error-page> | | 错误页面 <exception-type> <location> |
| <jsp-config> | | 设置jsp |

# Servlet过滤器

|  |  |
| --- | --- |
| 定义 | 是一个小型的web组件，它们通过拦截请求和响应，以便查看、提取或以某种方式操作客户端和服务器之间交换的数据，实现过滤的功能。 拦截响应??? |
| 编写步骤 | 1. 写一个java类，实现javax.servlet.Filter接口 或 直接新建Filter文件 |
| 1. 在doFilter()里编写处理逻辑   如有需要，可以在init()设置一个FilterConfig实例接收FilterConfig参数  FilterChain:过滤器链，如调用了该对象的doFilter()，则容器会继续向后调用过滤器或处理器 |
| 1. 在web.xml配置Filter，如有多个过滤器，容器会依据<filter-mapping>配置的先后顺序来执行 |
| 优点 | 1. 实现代码的"可插拔性"，即增加或减少某个功能模块，不会影响程序的正常执行 2. 可以将多个相同处理逻辑的模块集中写在过滤器里面，方便代码的维护 |

# 

# Servlet监听器

|  |  |
| --- | --- |
| 定义 | Servlet规范中定义的一种特殊的组件，用来监听Servlet容器产生的事件并进行相应的处理 |
| 生命周期相关 | ServletRequestListener |
| HttpSessionListener: 统计网站在线人数 |
| ServletContextListener: 统计网站历史访问次数，系统启动时初始化信息 |
| 绑定数据相关 | ServletRequestAttributeListener |
| HttpSessionAttributeListener |
| ServletContextAttributeListener |
| 编写步骤 | 写一个java类，实现XxxListener接口 或 直接新建Listener文件 |
| 在监听器接口方法中编写处理逻辑 |
| 在web.xml配置Listener |