1、什么是JavaEE

javaee 是企业级应用开发。

2、软件的分类

- C/S 结构是Client/Server（客户机/服务器）的简称，
  - 桌面应用程序采用的多是这种结构；
- B/S 结构是Browser/Server（浏览器/服务器）的简称，
  - 特点是客户端无需安装特定的软件，只需要安装一个浏览器就可以与系统进行交互。

3、Web应用服务器

Web应用服务器能够运行服务器上的应用程序，并将结果返回给客户端浏览器；

例如，**Tomcat**就是一种Web应用服务器；通常情况下，Web应用服务器兼具HTTP服务器的部分功能；

4、Servlet 的概念及功能

- Servlet是JavaEE规范中的Web开发组件；
- Servlet运行在服务器端，需要Servlet容器的支持，例如Tomcat；
- 可以通过浏览器访问Servlet，Servlet可以生成动态页面返回给浏览器；
- Servlet也是一段代码，是一个Java类，这个Java类需要遵守一定的编写规范，例如，必须继承于javax.servlet.http.HttpServlet类；

5、客户端访问服务器端Servlet的方式

- 直接从地址栏输入URL访问； --- get请求
- 在网页中点击超级链接访问；--- get请求
- 在网页中通过表单提交访问；--- 根据method属性决定请求方式
- AJAX

6、传递请求参数的方式

- 超链接 --- <a href="MyServlet?uname='tom'&upass=123">跳转到servlet中
- 表单 --- name 属性 <input type="text" name="uname"/>

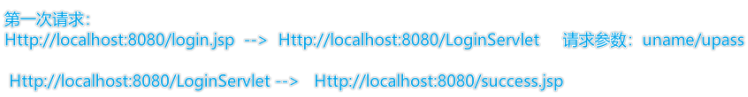
6.1 获取请求参数

方法声明	方法描述
java.lang.String **getParameter**(java.lang.String name)	返回某个指定名字的请求参数的值，值为String类型；
java.lang.String[] getParameterValues(java.lang.String name)	返回指定名字的请求参数的值，值为String[]类型，一般用于一个名字对应多个值情况；

```
// 解决响应的乱码问题
response.setContentType("text/html;charset=utf-8");
// 解决请求的乱码问题
request.setCharacterEncoding("utf-8");
```

7、Servlet跳转

- 响应重定向
  - response.sendRedirect("LoginFailServlet");
  - 发起了两次请求，地址栏改变，不能传递请求参数

  - 

- 请求转发
  - request.getRequestDispatcher("LoginSuccessServlet").forward(request, response);
  - 一次请求，地址栏不变，可以传递请求参数

8、常用的会话跟踪技术有四种

- URL方式：需要保存的信息直接追加到URL后，
  - 例如：http://127.0.0.1:8080/chapter03/viewList?pageNo=12
- 隐藏域方式：可以使用表单中的隐藏域保存相关信息，
  - 例如：<input type="hidden" name=“status" value=“true">
- **Cookie方式**：将状态信息保存到客户端，服务器能够获得相关信息进行分析，从而生成对客户端的响应；例如简化登录功能就可以使用Cookie实现；
- **Session方式**：将状态信息保存到服务器的会话对象中，通过唯一标记的ID值与客户端进行绑定使用；例如访问控制功能就可以使用Session实现；

8.1 什么是Cookie

Cookie是Web服务器**保存在客户端**的一系列**文本信息**

保存的位置分两种：

- Cookie可能保存在**客户端浏览器的所占内存中**，关闭浏览器后，Cookie就不再存在
- Cookie也可能保存在客户**PC机的硬盘上**，设置有效时间，超过有效时间后失效

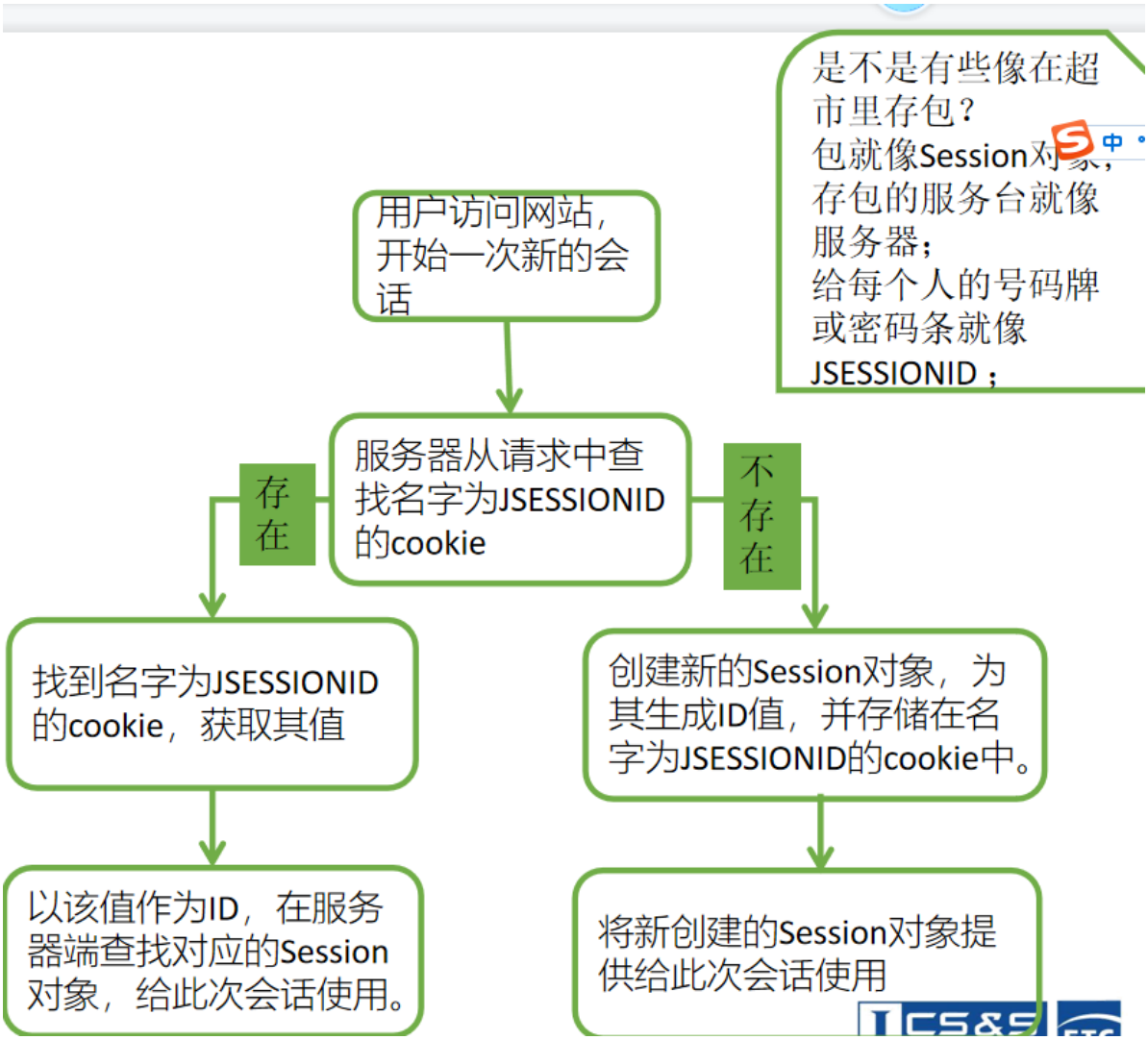
方法声明	方法描述
Cookie(java.lang.String name, java.lang.String value)	创建Cookie对象，指定名字和对应的值；
void setMaxAge(int expiry)	设置最大生命时间（秒），如果不设置，当前浏览器关闭，cookie即失效；
void setValue(java.lang.String newValue)	设置Cookie的值；
setDomain(java.lang.String domain)	设置cookie的域名；

8.2 什么是Session

客户端向服务器端发送请求，服务器端接受请求并生成响应返回给客户端，客户端对服务器端这样一次连续的调用过程，被称为会话（session）。

Session是存储在服务器上的对象，该对象由服务器创建并维护；

8.2.1 session 的执行流程



8.2.2 获取session 对象

方法声明	方法描述
HttpSession getSession()	获取与当前请求相关的Session对象，如果不存在，创建一个新的；
HttpSession getSession(boolean create)	如果create为true，则与getSession()方法相同；如果create是false，则如果不存在，返回null;

8.2.3 session 的常用方法

方法声明	方法描述
void setAttribute(java.lang.String name, java.lang.Object o)	将任意类型对象设置为会话的属性，指定一个名字；
java.lang.Object getAttribute(java.lang.String name)	通过属性的名字，获取属性的值；
void removeAttribute(java.lang.String name)	通过属性的名字，删除属性；

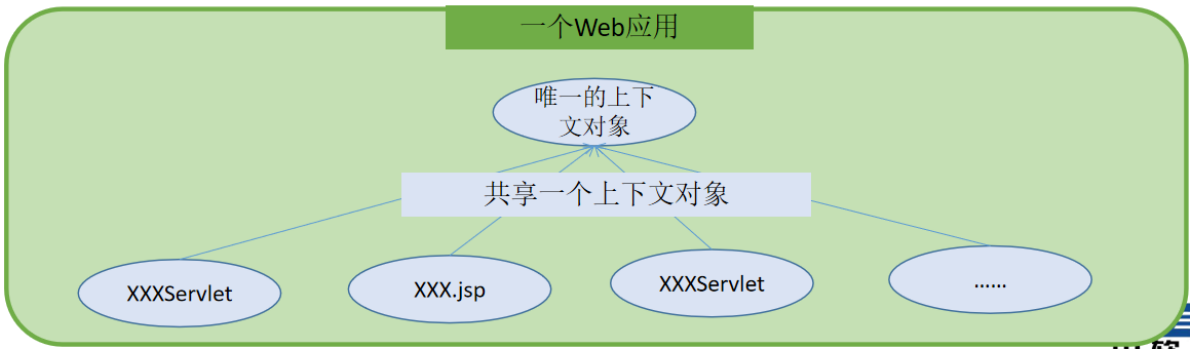
8.2.4 session 的有效时间

- 服务器都有默认的会话失效时间，Tomcat默认是30**分钟**；
- 可以在web.xml中配置失效时间，例如：配置失效时间是50**分钟**；
  - ```
    <session-config>
      <session-timeout>50</session-timeout>
    </session-config>
    ```
- 调用HttpSession接口中的两个方法，可以对指定的会话对象进行销毁；
方法声明	**方法描述**
void setMaxInactiveInterval(int interval)	单位是秒 为特定的会话对象设定不活动时间，超过这个时间内没有被访问使用，容器自动销毁该会话对象；
void invalidate()	立刻销毁调用该方法的会话对象，并把所有绑定到该会话的对象解除绑定；

> 优先级：
>
> setMaxInactiveInterval(int interval)
>
> 在本项目中配置的有效时间
>
> 最后采用的是服务器的默认时间

9、上下文

- 上下文对象是用来存储全局范围信息的对象；换句话说，一个Web应用只有唯一 一个上下文对象；
- 当服务器启动的时候，就会为每一个应用创建一个上下文对象；
- 当服务器关闭的时候，上下文对象就销毁；



9.1 获取上下文对象

- servlet 中
  - ```
    // 获取上下文对象
    ServletContext ctxt = getServletContext();
    ```
- jsp 中

- ```
  <%--  获取上下文对象: application --%>
  <%=application.hashCode()%>
  ```

> **servlet 的作用范围（作用域）：（从小到大）**
>
> 请求范围 request ---- 在同一个请求中
>
> 会话范围 session ---- 在同一个会话中（一次连续的调用过程，没有关闭浏览器）
>
> 上下文范围 ServletContext ----- 没有关闭服务器

9.2 上下文参数

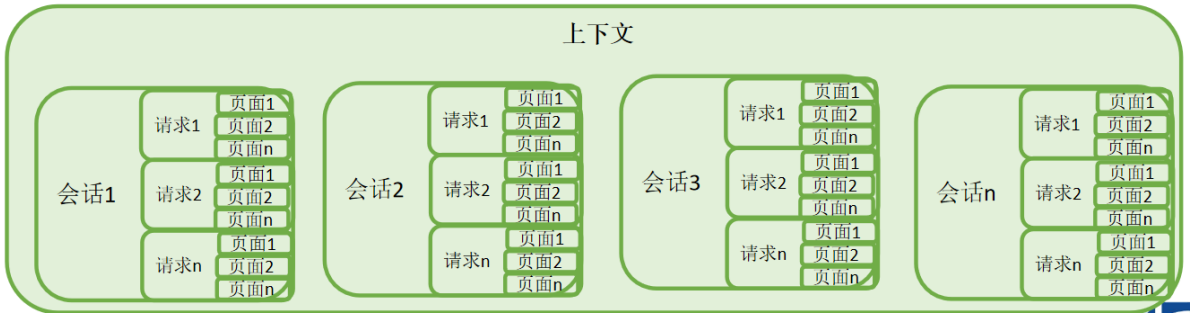
- 在**web.xml**中可以配置上下文参数，使用ServletContext中的**getInitParameter**方法可以获取该参数；
- ```
  <context-param>
          <param-name>version</param-name>
          <param-value>2.0</param-value>
  </context-param>
  ```
- 获取上下文参数：
  - ```
    返回ServletContext对象
            ServletContext ctxt=this.getServletContext();
    //      获取上下文参数
            String version=ctxt.getInitParameter("version");
            System.out.println("上下文参数version的值："+version);
    ```

9.3 上下文范围

方法声明	方法描述
void setAttribute(java.lang.String name, java.lang.Object o)	将任意类型对象设置为上下文属性，指定一个名字；
java.lang.Object getAttribute(java.lang.String name)	通过属性的名字，获取属性的值；
void removeAttribute(java.lang.String name)	通过属性的名字，删除属性；

9.4 四个作用范围

- 在Web应用中，有四大作用域范围
  - 页面范围：一个Servlet或JSP文件；
  - 请求范围：一次请求中可以访问多个Servlet或JSP； 访问的Servlet或JSP能够包含其他资源；
  - 会话范围：一次会话中可以包含多个请求；
  - 上下文范围：上下文包含所有会话；
    ```



10、监听器 Listener接口

- 监听器是为了监听事件并处理的，所以要理解监听器的概念，首先要理解事件的概念。
- 某些操作总会触发一种事件发生，如启动或关闭容器、创建或销毁会话等。当发生了某种事件，容器将创建对应的事件类对象。也就是说，API中已经定义好了事件的类型，容器进行了实现，当某些特定操作发生时，会自动触发相应的事件。

- Servlet API中定义了8种监听器接口，用来监听不同的事件类型
- 上下文相关的监听器
  - ServletContextListener：上下文监听器，监听ServletContextEvent事件。
  - ServletContextAttributeListener：上下文属性监听器，用来监听ServletContextAttribute事件。
- 请求相关的监听器
  - ServletRequestListener：请求监听器，监听ServletRequestEvent事件。
  - ServletRequestAttributeListener：请求属性监听器，用来监听ServletRequestAttributeEvent事件。
- 会话相关的监听器
  - HttpSessionListener：会话监听器，监听HttpSessionEvent。
  - HttpSessionActivationListener：会话活化监听器，监听HttpSessionEvent事件。
  - HttpSessionAttributeListener：会话属性监听器，监听HttpSessionAttributeEvent事件。
  - HttpSessionBindingListener：会话绑定监听器，监听HttpSessionAttributeEvent事件。

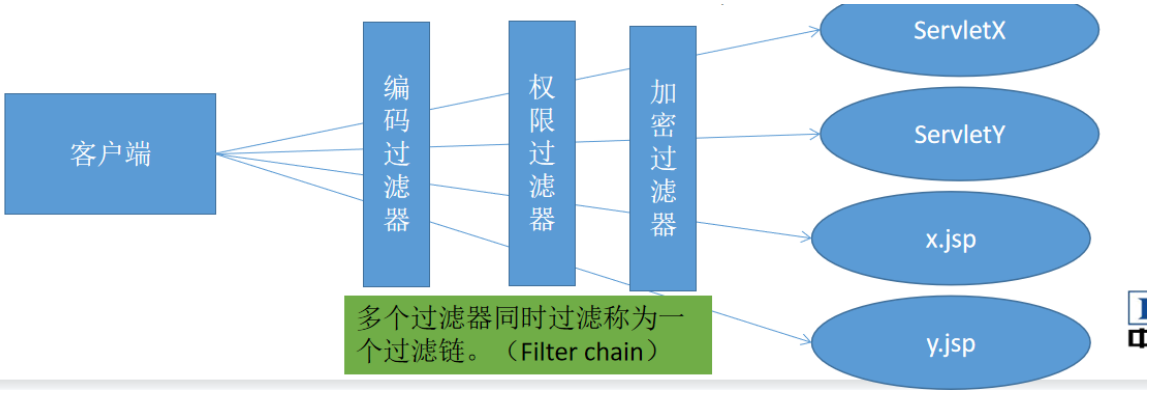
10.1 监听的步骤

- 编写监听器非常简单，只需要：
  - 写一个类实现相应的XXXLisntener接口；
  - 重写接口中的方法，实现监听的功能；

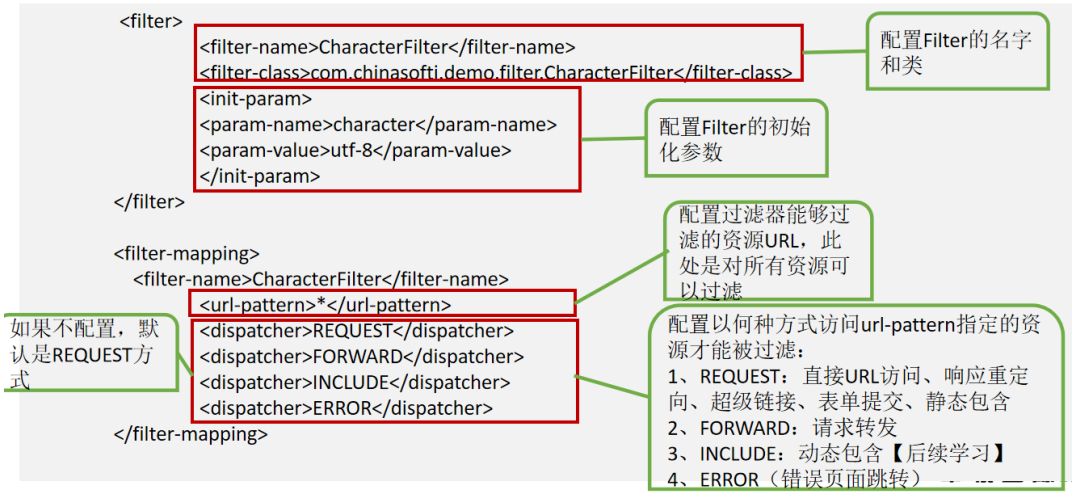
- 要想监听器生效，需要在web.xml中进行配置，例如：
  - ```
    <listener>
    <listener-class>com.chinasofti.demo.listener.VisitCountsListener</listener-class>
    </listener>
    ```
  - ```
    @WebListener
    ```

11、过滤器 Filter 接口

- 在Web应用中，如果对服务器端的多个资源（Servlet/JSP）有“通用”的处理，可以在每个资源中写相同的代码，而这样做显然过于冗余，修改时就需要逐一修改，效率低下；
- 过滤器可以解决这样的问题：把通用的、相同的处理代码用过滤器实现，然后在web.xml中将过滤器配置给相关的资源使用即可；



- 过滤器的开发非常简单：
  - 自定义类实现Filter接口；
  - 实现接口中的方法，重点是doFilter方法：
- Filter接口中有三个方法，如下；
方法声明	方法描述
void init(FilterConfig filterConfig)	容器初始化过滤器对象后调用该方法，其中参数可以获取过滤器配置信息；
void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)	过滤器的服务方法，有三个参数，其中FilterChain中也定义了名字为doFilter方法，不过只有两个参数，可以把当前的请求和响应沿着过滤链进行传递；
void destroy()	容器销毁过滤器对象前进行调用；
- 过滤器的配置



12、AJax

Ajax：只刷新局部页面的技术,使用Ajax技术构建Web应用，能够实现异步提交请求，并可以避免刷新整个页面，提高用户体验。

- JavaScript：更新局部的网页
- XML：一般用于请求数据和响应数据的封装
- **XMLHttpRequest对象**：发送请求到服务器并获得返回结果
- CSS：美化页面样式
- 异步：发送请求后不等返回结果，由回调函数处理结果

- 同步与异步：
  - 同步：发送一个请求，需要等待返回结果才能发送下一个请求
  - 异步：发送一个请求，不需要等待返回结果就能发送下一个请求

12.1 JQ Ajax

- jquery对Ajax做了大量封装
- 三层封装：
  - 第一层$.ajax() 复杂

- 第二层.load() .get() .post()
- 第三层 $.getJson() 简单

- 使用ajax() 方法通过 HTTP 请求加载远程数据。
  - ajax()方法是 jQuery 底层 AJAX 实现，$.ajax()方法 返回其创建的 XMLHttpRequest 对象。
- 一、$. ajax的一般格式

- $. ajax({
- type: 'POST', --提交方式
- url: url , --提交地址
- data: data , --提交数据
- dataType: dataType, --响应数据的类型（根据提交类型来判定）
- success: success --成功操作

async ： false--同步请求，true--异步（默认）

- });

data主要方式有三种：
html拼接的，json数组，form表单经serialize()序列化的；
通过dataType指定，不指定智能判断。

get(url, [data], [callback])

url (String) 发送请求的URL地址.
data (Map)(可选参数) 要发送给服务器的数据，以 Key/value 的键值对形式表示，会做为QueryString附加到请求URL中
callback (Callback) (可选参数) 载入成功时回调函数(只有当Response的返回状态是success才是调用该方法)

简化的ajax请求：

如果是增删改 --- get 、 post 请求

如果是查询 --- getJSON请求