J2EE 期末复习参考

往年考题解析

1. JSP out的隐含对象

隐含对象	类型
out	JspWriter
config	ServletConfig
page	JSP.this
pageContext	PageContext
exception	throwable
request	HttpServletRequest
response	HttpServletResponse
application	ServletContext
session	HttpSession

2. EL（表达式语言）

   记住，表达式语言中的隐含对象除了 pageContext 的类型是 PageContext外，其余类型皆是 Map。

3. PageContext类：

   PageContext 扩展了 JspContext，提供有关何时在 Servlet 环境中使用 JSP 技术的有用上下文信息。

   PageContext 实例提供对与某个 JSP 页面关联的所有名称空间的访问，隐式对象是自动添加到 pageContext 的。

   在PageContext类中的get对象基本都是其隐含对象，而RequestDispatcher必然不会被包含其中。
   RequestDispatcher应该由Servlet对象来创建和取得。

All Methods | Instance Methods | Abstract Methods | Concrete Methods

Modifier and Type	Method and Description
abstract void	**forward**(String relativeUrlPath) This method is used to re-direct, or "forward" the current ServletRequest and ServletResponse to another active component in the application.
ErrorData	**getErrorData**() Provides convenient access to error information.
abstract Exception	**getException**() The current value of the exception object (an Exception).
abstract Object	**getPage**() The current value of the page object (In a Servlet environment, this is an instance of javax.servlet.Servlet).
abstract ServletRequest	**getRequest**() The current value of the request object (a ServletRequest).
abstract ServletResponse	**getResponse**() The current value of the response object (a ServletResponse).
abstract ServletConfig	**getServletConfig**() The ServletConfig instance.
abstract ServletContext	**getServletContext**() The ServletContext instance.
abstract HttpSession	**getSession**() The current value of the session object (an HttpSession).
abstract void	**handlePageException**(Exception e) This method is intended to process an unhandled 'page' level exception by forwarding the exception to the specified error page for this JSP.
abstract void	**handlePageException**(Throwable t) This method is intended to process an unhandled 'page' level exception by forwarding the exception to the specified error page for this JSP.
abstract void	**include**(String relativeUrlPath) Causes the resource specified to be processed as part of the current ServletRequest and ServletResponse being processed by the calling Thread.
abstract void	**include**(String relativeUrlPath, boolean flush) Causes the resource specified to be processed as part of the current ServletRequest and ServletResponse being processed by the calling Thread.
abstract void	**initialize**(Servlet servlet, ServletRequest request, ServletResponse response, String errorPageURL, boolean needsSession, int bufferSize, boolean autoFlush) The initialize method is called to initialize an uninitialized PageContext so that it may be used by a JSP Implementation class to service an incoming request and response within it's _jspService() method.
BodyContent	**pushBody**() Return a new BodyContent object, save the current "out" JspWriter, and update the value of the "out" attribute in the page scope attribute namespace of the PageContext.
abstract void	**release**() This method shall "reset" the internal state of a PageContext, releasing all internal references, and preparing the PageContext for potential reuse by a later invocation of initialize().

4. ServletConfig类：

   这是servlet的配置类，负责在 servlet 容器初始化时向 servlet 传递信息。

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
String	**getInitParameter**(String name) Gets the value of the initialization parameter with the given name.
Enumeration<String>	**getInitParameterNames**() Returns the names of the servlet's initialization parameters as an Enumeration of String objects, or an empty Enumeration if the servlet has no initialization parameters.
ServletContext	**getServletContext**() Returns a reference to the **ServletContext** in which the caller is executing.
String	**getServletName**() Returns the name of this servlet instance.

> 有一个小技巧，就是servletconfig类只能获取初始化参数（InitParameter）而不能获取参数（parameter），这是由于servletconfig只是负责初始化servlet而并不会对servlet内的参数。参数这种东西只会在请求中出现。初始化尚未完成何来请求？

5. servlet类：

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type	Method and Description
void	**destroy**() Called by the servlet container to indicate to a servlet that the servlet is being taken out of service.
ServletConfig	**getServletConfig**() Returns a **ServletConfig** object, which contains initialization and startup parameters for this servlet.
String	**getServletInfo**() Returns information about the servlet, such as author, version, and copyright.
void	**init**(ServletConfig config) Called by the servlet container to indicate to a servlet that the servlet is being placed into service.
void	**service**(ServletRequest req, ServletResponse res) Called by the servlet container to allow the servlet to respond to a request.

servlet中的Service函数可以抛出 Servlet Exception 当 servlet 出现异常时；由于这个服务可能会涉及到输入输出，所以IOException是必然会有的，用来处理输入输出时可能遇到的问题。

6. WebServlet类：

   本注释用于声明一个Servlet。本注释会在部署的时候由容器处理，这个注释对应的Servlet可以通过注释声明的URL地址进行访问。

Optional Element Summary

Optional Elements

Modifier and Type	Optional Element and Description
boolean	**asyncSupported** 声明这个servlet是否支持异步操作。
String	**description** 这个Servlet的描述。
String	**displayName** 这个Servlet向外展示的名字。
WebInitParam[]	**initParams** 这个Servlet的初始化参数。
String	**largeIcon** 这个Servlet的large-icon。
int	**loadOnStartup** The load-on-startup order of the servlet
String	**name** 这个Servlet的名字。
String	**smallIcon** 这个Servlet的small-icon。
String[]	**urlPatterns** 这个Servlet的所有URL地址。
String[]	**value** 这个Servlet的所有URL地址。

> 你不看上图也能推测到，URLPattern必然是一个String类型的，它的复数形式就是String数组。

7. Annotation：

你需要了解的Annotation只有这四个：

1. WebFilter是用来这个注释用来声明一个Servlet过滤器，会在部署的时候被容器处理，和本注释对应的过滤器会被应用到一个特定的URL地址和Servlet，也可以通过注释指定过滤器的派发类型。
2. WebInitParam用在Servlet和过滤器的实现类上，用来声明它们的初始化参数。
3. WebListener用来声明一个WebListener。
4. 最后一个第六题已经讲了。

8. HttpServletResponse类：

拓展 ServletResponse 接又来提供 HTTP 在发送响应时的功能。它可以通过特定的方法来访问 HTTP 头部和 cookies。

它的方法无非是操作HTTP协议报文：

1. 访问cookie
2. 访问数据报头（和它相关的操作）
3. 访问报头（和它相关的操作）
4. 设置状态码

9. HttpServletRequest类：

这个类是用来给 HTTP 提供请求信息的，也是对ServletRequest的拓展。因为这个类封装了请求信息，所以几乎就没有set方法，可以通过这个类获取这些请求信息，如URI上下文（URL和查询字符串）、Cookies、HTTP报头相关、HTTP方法（DOGHPPT），以及请求所在会话的相关信息（session）。

Modifier and Type	Method and Description
boolean	**authenticate(HttpServletResponse response)** 使用 ServletContext的容器登录机制来认证发送请求的用户。
String	**changeSessionId()** 改变与本请求相关的会话的id，返回新的id号。
String	**getAuthType()** 返回搭建本 servlet 的认证方案类型.
String	**getContextPath()** 返回请求 URI 中指明请求上下文的那部分。
Cookie[]	**getCookies()** 返回随本请求发送的包含所有 Cookie 对象的数组。
long	**getDateHeader(String name)** 返回一个Date对象的long类型的值，该值对应一个指定名字的请求头部。
String	**getHeader(String name)** 返回 String类型的指定请求头部的值。
Enumeration<String>	**getHeaderNames()** 返回这个请求包含的所有报头名字的集合。
Enumeration<String>	**getHeaders(String name)** 返回一个指定的请求报头对应的所有值的集合（所有值都是String 类型）
int	**getIntHeader(String name)** 返回一个int类型的请求报头的值。
String	**getMethod()** 返回生成该请求用到的 HTTP 方法名字，例如 GET, POST, 或者 PUT.
String	**getQueryString()** 返回查询字符串（在该请求 URL 路径部分的后面）。
String	**getRequestedSessionId()** 返回由客户端所确定的的会话ID。
String	**getRequestURI()** 返回请求 URL 中从协议名字到查询字符串的部分（这部分出现在 HTTP 请求的第一行）。
StringBuffer	**getRequestURL()** 重构客户端发送请求的 URL。
String	**getServletPath()** 返回请求 URL 中调用本 servlet 的那一部分。
HttpSession	**getSession()** 返回和本请求关联的目前正在使用的会话，如果该请求没有创建会话，那么就帮它创建一个。
HttpSession	**getSession(boolean create)** 返回和本请求相关联的 HttpSession 对象，如果目前还没有创建会话并且 create为 true, 返回一个新的会话。
boolean	**isRequestedSessionIdFromCookie()** 检查被请求的会话ID是否被服务器以 HTTP cookie 的形式传送。
boolean	**isRequestedSessionIdFromURL()** 检查被请求的会话ID是否被服务器作为请求 URL 的一部分。
boolean	**isRequestedSessionIdValid()** 检查被请求的会话ID是否仍然有效。

那为什么请求URL要想得到需要返回的是Stringbuffer类型而不是String类型的？因为实际上并没有完整现成的URL，而是需要通过调用该函数根据已有的信息如URI和查询字符串合成一个URL，这就需要缓冲区来存储可能需要的数据。

0. JSP的三大元素：

   众所周知，JSP只需要理解它的三大元素就可以了：

元素	内容	表示
脚本元素	声明(用于定义在其它脚本元素中可以使用的变量、方法或类)	<%! ... %>
	脚本段(在一个脚本段定义的变量为局部变量)	<% Java代码块 %>
	表达式(java 语言中完整的表达式)	<%= Java 表达式 %>
指令元素	Page指令（用于设置 JSP页面的属性）	<% @ page 属性1="属性1" ... %>
	Include指令（用于在 jsp 页面中静态包含一个文件）	<%@ include file = "文件名" %>
	taglib 指令（将标签库描述符文件引入到该 JSP页面中，利用前缀名去引用）	<%@ taglib uri = "标签库表述符文件" prefix = "前缀" %>
动作元素	>	用于实例化 JavaBean，或者定位一个已经存在的 JavaBean实例，并把实例的引用赋给一个变量
	<jsp:setProperty>	使用 Bean 中的 setXXX() 方法设置 JavaBean的简单属性和索 引属性。
	<jsp:forward>	允许在运行时将当前的请求转发给一个静态的资源。
	<jsp:include>	用于在当前页面中包含静态和动态的资源 ,一旦被包含的页面执行完毕,请求被转向到页面中继续执行。
	<jsp:getProperty>	用来访问一个 Bean 的属性，并把属性的值转化成一个 String，然后发送到输出流中。

1. HttpSessionEvent

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`HttpSession`	`getSession()` Return the session that changed.

这个类其实很简单，就是根据一个给定的事件源创建一个事件。事件是会话的事件，在事件结束后会使用上面的这个方法返回那个被事件更新了的会话。这个类除了构造函数外就只有这一个方法。

2. HttpSessionAttributeListener

Method Summary

All Methods | Instance Methods | Default Methods

Modifier and Type	Method and Description
default void	**attributeAdded**(**HttpSessionBindingEvent** event) Receives notification that an attribute has been added to a session.
default void	**attributeRemoved**(**HttpSessionBindingEvent** event) Receives notification that an attribute has been removed from a session.
default void	**attributeReplaced**(**HttpSessionBindingEvent** event) Receives notification that an attribute has been replaced in a session.

这个类是用来监听Http会话属性的改变的。一个HttpSession的事件会造成HttpSession属性的改变，Listener就会监听这种改变。其中，对属性的操作就对应着该类的三个方法：增添属性、删除属性、替换属性。

3. Spring Controller

controller处理函数的返回数据：

1. ModelAndView
2. Model
3. ModelMap
4. Map
5. View
6. String
7. void
8. @ResponseBody Object

唯独是没有List类型。

4. <jsp: include> 与 <%@ include file = ”文件名” %> 的区别：

前者可以用于访问当前页面中的动态资源和静态资源；后者只能包含静态资源。那么，什么是网页中的动态资源和静态资源呢？这里的静态与动态都是指J2EE 组件的静态与动态，只有动态组件才是web层的组件：

类型	组件
静态组件	JSP页面、Servlet
动态组件	Applet、HTML页面

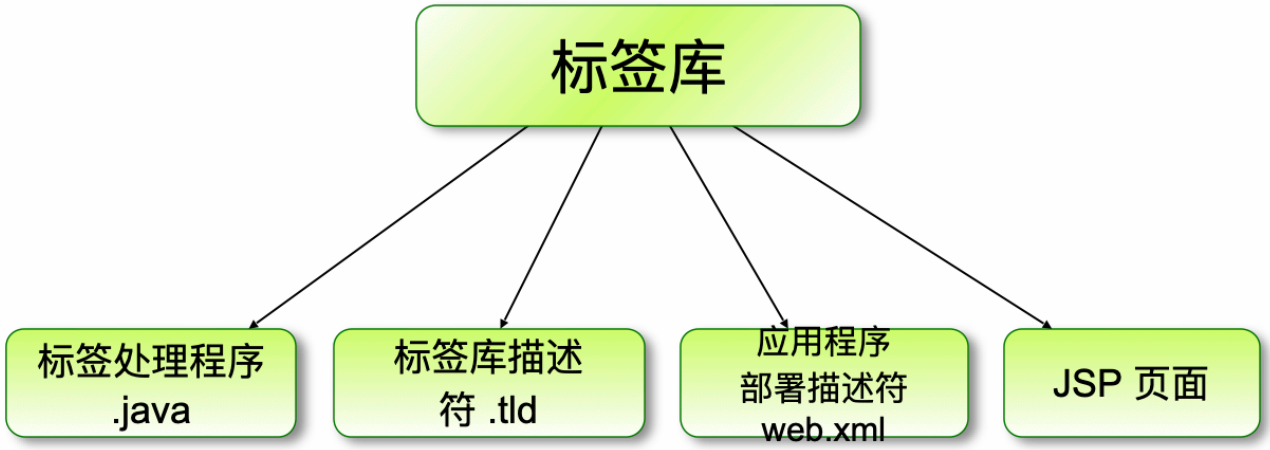
所以说如果想要包含 servlet 的话必须只能是用<jsp: include> 。它是一个 jsp 动作，动作是动态的动作；而 <%@ include file = ”文件名” %> 只是一个指令元素，它是静态的指令。

<%! ... %> 与 <% ... %> 的区别：

前者是脚本的声明，用于定义在其它脚本元素中可以使用的变量、方法或类，换句话讲就是一个使用它可以定义一个全局变量；而后者是脚本段，用于在一个脚本段定义的变量为局部变量。所以第二次调用的时候 a 已经累加了两次，而 b 仍是从0开始累加一次，因为局部变量在每一次调用JSP时都会清零。

5. 标记库描述符文件

1. 标签处理程序的组件及其关系：



2. 什么是自定义标签?

> - 自定义标签在 JSP 页面中嵌入少量的 Java 代码.
> - 自定义标签遵循标准 HTML 标签格式。
> - 可以通过创建 Java 类来实现自定义标签
> - 需要标签处理程序和标签库描述符文件

3. 如何自定义一个标签?

步骤	内容	意义
1	编写 *.java 文件	标签处理程序文件
2	编写 *.tlb 文件	标签库描述符文件
3	编写 web.xml 文件	应用程序部署描述符文件，指定对标签的引用

所以，对这道题而言，标签库描述符文件是一个以.tlb 结尾的文件；只有自定义的标签才需要在该文件中声明；只有自定义的标签需要标记处理类的声明；该文件必须要部署在/WEB-INF 目录下：

```
<%@ taglib uri='WEB-INF/welcome.tld' prefix='w' %> // 通过 prefix = w 来引用这个标签
<html>
  <body>
    <w:HelloWorldTag />
  </body>
</html>
```

6. ServletContext类

是一个Servlet的环境对象，用来创建一个Servlet的环境信息。

它定义的方法是为了能让servlet 能和创建它的 servlet 容器建立沟通，比如派发一个请求、写入一个日志文件等。

每个web 应用都只能有一个 Context 对象（web 应用程序是一个 servlet 的集合）。

servletContext类里面的方法主要有这么几大类：

1. 对Filter的操作
2. 对Listener的操作
3. 对Servlet的操作
4. 对Attribute的操作

5. 对Context的操作
6. 对InitParameter的操作
7. 对log的操作
8. 对Path的操作

方法摘要	
java.lang.Object	**getAttribute**(java.lang.String name) 返回Servlet环境对象中指定的属性对象。
java.util.Enumeration	**getAttributeNames**() 返回一个Servlet环境对象中可用的属性名的列表。
ServletContext	**getContext**(java.lang.String uripath) 返回一个Servlet环境对象，这个对象包括了特定URI路径的Servlets和资源，如果该路径不存在，则返回一个空值。
int	**getMajorVersion**() 返回Servlet引擎支持的Servlet API的主版本号。
java.lang.String	**getMimeType**(java.lang.String file) 返回指定文件的MIME类型，如果这种MIME类型未知，则返回一个空值。
int	**getMinorVersion**() 返回Servlet引擎支持的Servlet API的次版本号。
java.lang.String	**getRealPath**(java.lang.String path) 一个符合URL路径格式的指定的虚拟路径的格式是：/dir/dir/filename.ext。
RequestDispatcher	**getRequestDispatcher**(java.lang.String uripath) 如果这个指定的路径下能够找到活动的资源(例如一个Servlet，JSP页面，CGI等等)就返回一个 特定URL的RequestDispatcher对象，否则，就返回一个空值。Servlet引擎负责用一个 request dispatcher对象封装目标路径。
java.net.URL	**getResource**(java.lang.String uripath) 返回一个URL对象，该对象反映位于给定的URL地址（格式：/dir/dir/filename.ext） 的Servlet环境对象已知的资源。
java.io.InputStream	**getResourceAsStream**(java.lang.String uripath) 返回一个InputStream对象，该对象引用指定的URL的Servlet环境对象的内容。
java.lang.String	**getServerInfo**() 返回一个String对象，该对象至少包括Servlet引擎的名字和版本号。
Servlet	**getServlet**(java.lang.String name) 最初用来返回一个指定名称的Servlet，如果没找到就返回一个空值。
java.util.Enumeration	**getServletNames**() 最初用来返回一个String对象的列表，该列表表示了在这个Servlet环境下所有已知的Servlet对象名。
java.util.Enumeration	**getServlets**() 最初用来返回在这个Serveletl环境下所有已知的Servlet对象的列表。
void	**log**(java.lang.Exception exception, java.lang.String msg) （这种用法将被取消）写指定的信息到一个Servlet环境对象的log文件中。
void	**log**(java.lang.String msg) 写指定的信息到一个Servlet环境对象的log文件中。
void	**log**(java.lang.String msg, java.lang.Throwable t) 写指定的信息到一个Servlet环境对象的log文件中。
void	**removeAttribute**(java.lang.String name) 从指定的Servlet环境对象中删除一个属性。
void	**setAttribute**(java.lang.String name, java.lang.Object o) 给予Servlet环境对象中你所指定的对象一个名称。

7. Cookie类：

   一个cookie有一个名字、一个值、可选的属性：comment（描述）、path（路径）、domain（作用域）、age（寿命）、version（版本）。

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
Object	**clone**() 覆盖掉标准的 java.lang.Object.clone 返回这个 cookie 的拷贝。
String	**getComment**() 返回一个描述来解释这个 cookie 的用意，或者返回 null 如果这个 cookie 没有comment。
String	**getDomain**() 返回这个 cookie 的作用域名字。
int	**getMaxAge**() 以秒为单位返回这个 cookie 的最大寿命。
String	**getName**() 返回这个 cookie 的名字。
String	**getPath**() 返回一个路径。这个路径是浏览器将这个cookie发送到的那个服务器的路径。
boolean	**getSecure**() 返回 true 如果浏览器只是用安全协议来发送这个 cookie。或者返回 false ，如果浏览器可以使用任何协议来发送这个 cookie。
String	**getValue**() 得到这个 cookie 目前的值。
int	**getVersion**() 返回编译这个 cookie 用到的协议的版本。
void	**setComment**(String purpose) 指明用来描述这个 cookie 的用意的comment。
void	**setDomain**(String domain) 指明这个 cookie 的作用域。
void	**setMaxAge**(int expiry) 设置这个 cookie 的最大寿命（以秒为单位）。
void	**setPath**(String uri) 设置一个路径，客户端将这个 cookie 返回到这个路径。
void	**setSecure**(boolean flag) 指示服务器这个cookie是否应该通过只安全协议来发送。
void	**setValue**(String newValue) 将 cookie 赋上一个新值。
void	**setVersion**(int v) 设置编译这个 cookie 的协议的版本号。

   所以，getAttribute绝对不会是Cookie类的成员。应该是在ServletContext里。

8. 定义方法的标签是声明标签：<%! ... %>

   <% ... %> 是脚本段标签，用于声明局部变量；

   <%= ... %> 是Java表达式标签，用于包含一个直接转换成Java代码的Java表达式；

<%@ ... %> 是指令元素标签，用来包含一个Page\Taglib\Include 指令

9. 记住，任何和protocol、value、name相关的返回值，如果它没有在函数签名中特殊说明需要转化成其他类型（如getIntValue)返回值默认都是String。所以getProtocol的返回值一定是String类型的。

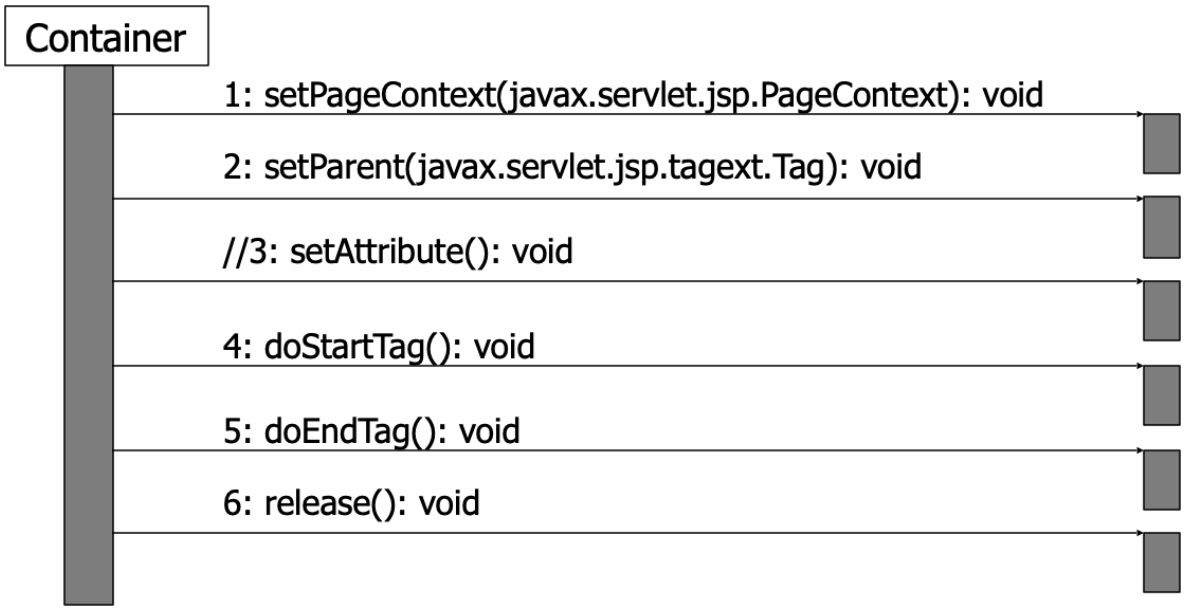
10. tag和simpleTag接口：

tag接口一般不会去直接实现，而是实现一个tagSupport来继承tag。它的方法主要是这样的：

```
public int doStartTag();  // 标签的开始标签内可以执行的动作
public int doEndTag();  // 标签的结束标签内可以执行的动作，它定义了四个结束标签后即将执行的动作：
/*
  Tag.EVAL_BODY_INCLUDE 包含主体内容
  Tag.SKIP_BODY      不包含主体内容
  Tag.EVAL_PAGE       包含后面的页面内容
  Tag.SKIP_PAGE      不包含主体的内容
*/

/* 我们知道，标签的开始和结束一定是配对的。 */
```

一个标签的生命周期是这样的：



填空代码解析