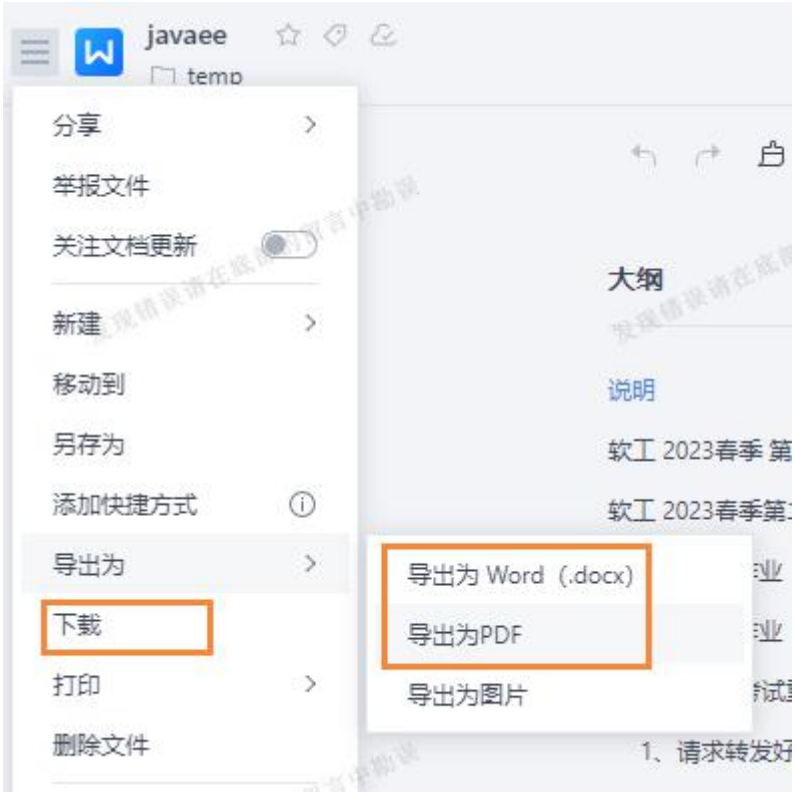
说明

点左上角三个杠可以下载 word 和 pdf。



建议打印成缩印

如果有发现错误可以在底部评论区勘误

软工 2023 春季 第一次作业

题目数量:40 道分值:100 分截止时间：2023-04-10 23:59:20

1、阅读下面 XML 代码片段：

<dateborn>1980-03-27</dateborn>

下面选项中能与之匹配的 Schema 是（ ） 2 分

A、<xs:element name="dateborn" type="xs: decimal"/>

B、<xs:element name="date" type="xs:date"/>

✓C、<xs:element name="dateborn" type="xs:date"/>

D、<xs:element name="dateborn" type="xs:time"/>

在 XML 代码片段中，<dateborn>1980-03-27</dateborn> 表示一个日期值，其格式为年-月-日。因此，与之匹配的 Schema 应该具有相应的类型定义。选项 C 中的类型定义为"xs:date"，表示一个日期类型，与给定的日期值相匹配。

2、下面关于客户端访问 Tomcat 服务器中的某个静态 HTML 文件时的说法中，正确的是（） 2 分

A、直接访问 HTML 等静态资源

✓B、先访问缺省 Servlet，由缺省 Servlet 再决定定位静态资源

C、先访问 HTML 静态资源，再访问缺省 Servlet

D、以上说法都不对

当客户端访问 Tomcat 服务器中的某个静态 HTML 文件时，Tomcat 会先尝试通过默认的 Servlet（通常是 DefaultServlet）来处理请求。默认的 Servlet 会根据配置和规则判断请求是否是针对静态资源，如果是，则会定位并返回相应的静态资源，否则将请求转发给适当的 Servlet 进行处理。

3、假设在 helloapp 应用中有一个 HelloServet.java 类，它位于 com.itheima 包中，那么这个类的.class 文件的存放路径应该是什么？（ ） 2 分

A、helloapp/HelloServlet.class

B、helloapp/WEB-INF/HelloServlet.class

C、helloapp/WEB-INF/classes/HelloServlet.class

✓D、helloapp/WEB-INF/classes/com/itheima/HelloServlet.class

根据 Java 的包命名规则，com.itheima 包中的 HelloServlet 类应该对应于 com/itheima/HelloServlet.java 文件的存放路径。当编译该类时，生成的.class 文件应该按照包的层次结构进行存放。在一个名为 helloapp 的应用中，通常会将.class 文件放置在 WEB-INF 目录下的 classes 目录中。由于 HelloServlet 位于 com.itheima 包中，因此它的.class 文件应该放置在 helloapp/WEB-INF/classes/com/itheima/HelloServlet.class 的路径下。

4、下列选项中，可以成功修改 Tomcat 端口号为 80 的是（） 2 分

A、<Connect port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />

B、<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />

✓C、<Connector port="80" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />

D、<Connect port="80" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />

在 Tomcat 的配置文件（通常是 server.xml）中，可以通过修改 Connector 元素的 port 属性来修改 Tomcat 的端口号。Connector 元素定义了 Tomcat 与客户端之间的连接，并指定了使用的协议、超时时间等参数。

5、开发中创建了一个 Servlet，该 Servlet 重写了其父类的 doGet()和 doPost()方法，那么其父类可能是（ ）

2 分

A、RequestDispatcher

B、HttpServletResponse

C、HttpServletRequest

✓D、HttpServlet

在 Java Web 开发中，Servlet 是通过继承 HttpServlet 类来创建的。 HttpServlet 类是 javax.servlet.http 包中的一个抽象类，它提供了处理 HTTP 请求的方法，包括 doGet()和 doPost()等。当开发者创建一个 Servlet 并重写 doGet()和 doPost()方法时，通常是通过继承 HttpServlet 类来实现的。因此，其父类可能是 HttpServlet，选项 D 是正确答案。

6、下面选项中，哪个头字段能够指出当前网页在客户端或代理服务器中缓存的有效时间？（） 2 分

A、Accept

B、Accept-Range

C、Accept-Age

✓D、Age

头字段是 HTTP 消息中的一部分，用于传递附加信息。在缓存机制中，可以使用 Age 头字段来表示资源在缓存中的存在时间。

7、一个 Servlet 可以被映射成虚拟路径的个数是 （） 2 分

A、1 个

B、2 个

C、0 个

✓D、多个

在 Java Web 开发中，Servlet 可以通过配置进行映射，使其能够响应特定的 URL 请求。Servlet 的映射可以通过多种方式进行设置，包括虚拟路径映射、扩展名映射、通配符映射等。当使用虚拟路径映射时，一个 Servlet 可以被映射成多个虚拟路径。这意味着可以使用不同的 URL 来访问同一个 Servlet，并由该 Servlet 处理相应的请求。

8、已知 web.xml 中存在如下配置：

```
<session-config>
   <session-timeout>2</session-timeout>
</session-config>
```

下面的说法，正确的是（）　2 分

A、在空闲状态下，2 秒后将导致 session 对象销毁

✓B、在空闲状态下，2 分钟后将导致 session 对象销毁

C、在空闲状态下，2 毫秒后将导致 session 对象销毁

D、在空闲状态下，2 小时后将导致 session 对象销毁

在给定的 web.xml 配置中，<session-timeout>2</session-timeout> 表示会话（session）的超时时间为 2 个单位。具体的单位取决于上下文，默认情况下，单位是分钟。因此，在空闲状态下，会话对象将在 2 分钟后销毁，而不是 2 秒、2 毫秒或 2 小时。

9、下列选项中，修改 Tomcat 端口号的文件是（）　2 分

A、conf.xml

B、context.xml

✓C、server.xml

D、service.xml

在 Tomcat 中，配置文件 server.xml 包含了 Tomcat 服务器的配置信息，包括连接器（Connector）的定义。连接器负责监听指定的端口并处理客户端的请求。在 server.xml 文件中，可以找到类似以下的配置：

```
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />
```

这个配置用于定义 Tomcat 监听的端口号。通过修改 port 属性的值，可以修改 Tomcat 的端口号。

10、已知在 web.xml 中配置的监听器如下：

```
<listener>
    <listener-class>cn.itcast.listener.MyListener</listener-class>
</listener>
```

则下面说法，正确的是（）　2 分

A、<listener>说明在 web.xml 中配置了一个监听器

✓B、< listener-class>是指监听器类所对应的包名及类名

C、一个 <listener>元素中可以出现多个< listener-class>子元素

D、<listener>元素中还可以添加<listener-name>子元素

11、下面选项中，在 web.xml 配置文件中定义 Servlet，包括 Servlet 的名称和 Servlet 的实现类的结点是（）
2 分

✓A、<servlet>

B、<servlet-mapping>

C、<servlet-config>

D、<wab-app>

在 web.xml 文件中，<servlet>元素用于定义 Servlet 的配置。它包含了 Servlet 的名称和 Servlet 的实现类的信息。

```
<servlet>
    <servlet-name>MyServlet</servlet-name>
    <servlet-class>com.example.MyServlet</servlet-class>
</servlet>
```

在上述示例中，<servlet-name>元素指定了 Servlet 的名称为"MyServlet"，<servlet-class>元素指定了 Servlet 的实现类为"com.example.MyServlet"。

12、下列选项中，关于配置 JAVA_HOME 环境变量的具体步骤，正确的是（） 2 分

A、打开环境变量，配置 path 参数

B、打开环境变量，配置 classpath 参数

✓C、打开环境变量，新建 JAVA_HOME 参数并配置值为 JDK 安装目录

D、以上说法都不对

13、下面选项中，用于在 web.xml 中配置监听器的元素是（） 2 分

A、<listener-url>

B、<url-listener>

✓C、<listener>

D、<listener-name>

在 web.xml 文件中，<listener> 元素用于配置监听器。它定义了一个监听器的相关配置信息，包括监听器的类名等。

```
<listener>
    <listener-class>com.example.MyListener</listener-class>
</listener>
```

在上述示例中，<listener> 元素用于配置一个监听器，其中 <listener-class> 子元素指定了监听器的类名为 "com.example.MyListener"。

14、下列关于 ServletConfig 对象的说法中，正确的是（ ）。 2 分

✓A、ServletConfig 对象可以用来获取 Servlet 的配置信息。

B、ServletConfig 对象可以实现 Servlet 信息的共享。

C、ServletConfig 对象保存的信息是通过 service()方法传递给 Servlet 的

D、ServletConfig 对象可以读取 web.xml 文件中所有的信息。

ServletConfig 对象是由 Web 容器在初始化 Servlet 时传递给 Servlet 的，用于提供 Servlet 的配置信息。通过 ServletConfig 对象，Servlet 可以获取在 web.xml 文件中配置的参数、初始化参数等。

15、下面选项中，用于根据虚拟路径得到文件的真实路径的方法是（） 2 分

✓A、String getRealPath(String path)

B、URL getResource(String path)

C、Set getResourcePaths(String path)

D、InputStream getResourceAsStream(String path)

在 Servlet 中，可以使用 getRealPath(String path)方法来获取指定虚拟路径的真实路径。该方法将虚拟路径转换为在服务器文件系统中的真实路径。

16、在浏览器中输入 http://localhost:8080/myWebApp/start/,就会调用 itcast 包中名为 Student 的 Servlet，那么在 web.xml 中配置正确的是（） 2 分

A、

```
<servlet>
    <servlet-name>/start/*</servlet-name>
    <servlet-value>itcast.Student</servlet-class>
</servlet>
```

B、

```
<servlet>
    <servlet-name>itcast.Student</servlet-name>
    <servlet-value>/start/*</servlet-class>
</servlet>
```

✓C、

```
<servlet>
    <servlet-name>student</servlet-name>
    <servlet-value>itcast.Student</servlet-class>
</servlet>
```

```
<servlet-mapping>
    <servlet-name>student</servlet-name>
    <url-pattern>/start/*</url-pattern>
</servlet-mapping>
```

D、

```
<servlet>
    <servlet-name>itcast.Student</servlet-name>
    <servlet-value>student</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>itcast.Student</servlet-name>
    <url-pattern>/start/*</url-pattern>
</servlet-mapping>
```

在 web.xml 中，需要配置一个<servlet>元素来定义 Servlet，并使用<servlet-mapping>元素将 Servlet 映射到 URL 模式上。根据给定的要求，我们需要将名为"Student"的 Servlet 映射到"/start/*"的 URL 模式上。
因此，选项 C、在 web.xml 中配置的内容正是满足要求的配置。其中，<servlet-name>为"student"，<servlet-class>为"itcast.Student"，并且使用<servlet-mapping>将 Servlet 名字为"student"的映射到"/start/*"的 URL 模式上。

17、下面关于 HTTP 请求头消息个数的说法中，正确的是（ ） 2 分

A、一个 HTTP 请求消息中只能允许有一个请求头消息

B、一个 HTTP 请求消息中只能允许有两个请求头消息

C、一个 HTTP 请求消息中只能允许有三个请求头消息

✓D、一个 HTTP 请求消息中允许有若干请求头消息

在 HTTP 请求消息中，可以包含多个请求头消息。请求头消息用于传递关于请求的附加信息，例如客户端信息、认证信息、缓存控制等。

18、 下面选项中，Tomcat 安装目录的子目录描述，错误的是（ ） 2 分

A、bin：用于存放 Tomcat 的可执行文件和脚本文件

B、conf：用于存放 Tomcat 的各种配置文件

✓C、lib：用于存放 Tomcat 服务器和 Web 应用程序需要访问的 DLL 文件

D、webapps：Web 应用程序的主要发布目录

选项 C、lib：用于存放 Tomcat 服务器和 Web 应用程序需要访问的 DLL 文件 是错误的描述。实际上，lib 目录用于存放 Tomcat 服务器和 Web 应用程序所需的 Java 库文件（JAR 文件），而不是 DLL 文件。

19、下列关于 ServletConfig 中 getServletName()方法的描述中，正确的是（ ） 2 分

A、获取 web.xml 中<param-name>元素的值

✓B、获取 web.xml 中<servlet-name>元素的值

C、获取 server.xml 中<servlet-name>元素的值

D、获取 server.xml 中< param-name >元素的值

ServletConfig 接口中的 getServletName()方法用于获取当前 Servlet 的名称。在 web.xml 中，使用<servlet-name>元素来定义 Servlet 的名称，而 getServletName()方法返回的就是该<servlet-name>元素的值。

20、下列选项中，哪个是 HTTP 请求行的各部分之间采用的分隔符？（ ） 2 分

✓A、空格

B、逗号

C、分号

D、叹号

HTTP 请求报文由 3 部分组成（请求行+请求头+请求体）

HTTP 请求行由三个部分组成：请求方法、请求目标和协议版本，它们之间使用空格作为分隔符。

示例 HTTP 请求行：

GET /index.html HTTP/1.1

21、下面选项中，说明浏览器允许接收图片的请求消息头有哪些？（）(多选) 5 分

A、Accept: text/html

✓B、Accept: image/gif

✓C、Accept: image/*

✓D、Accept: */*

B、Accept: image/gif 表示浏览器可以接收 GIF 格式的图片。

C、Accept: image/* 表示浏览器可以接收任意格式的图片。

D、Accept: / 表示浏览器可以接收任意类型的文件。

22、下列选项中，哪些是命名 XML 元素时应该遵守的规范（ ）（多选） 5 分

✓A、区分大小写，例如：<P>和<p>是两个不同的标记。

✓B、元素名称中，不能包含空格、冒号、分号、逗号和尖括号等，元素不能以数字开头。

✓C、建议名称不要以字符组合 xml（或 XML、或 Xml 等）开头。

✓D、建议名称的大小写尽量采用同一标准，要么全部大写，要么全部小写。

23、下面在 Servlet 映射的路径中使用通配符“*”的格式正确的是？（）　　5 分

✓A、*.扩展名

✓B、/*

C、/*.扩展名

D、/servlet/*.扩展名

24、阅读下面的代码：

```
<servlet>
  <servlet-name>TestServlet01</servlet-name>
<servlet-class>cn.itcast.chapter04.servlet.TestServlet01</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>  TestServlet01</servlet-name>
  <url-pattern>/TestServlet01</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>TestServlet01</servlet-name>
  <url-pattern>/Test01</url-pattern>
</servlet-mapping>
```

下面选项中，可以访问 chapter04 应用下该 Servlet 的是（）　5 分

✓A、http://localhost:8080/chapter04/TestServlet01

B、http://localhost:8080/chapter04/servlet/TestServlet01

✓C、http://localhost:8080/chapter04/Test01

D、http://localhost:8080/chapter04/servlet/Test01

25、将 web 应用发布到 tomcat 上 localhost 主机，以下哪几种方式可以完成（）（多选）　5 分

✓A、直接将 web 应用部署到 tomcat/webapps 下

✓B、将 web 应用通过 tomcat/conf/server.xml 进行配置

✓C、创建一个 xml 文件，并配置 web 应用信息，将 xml 文件放置 tomcat\conf\Catalina\localhost

D、tomcat/conf/context.xml 进行配置

A、直接将 web 应用部署到 Tomcat 的 webapps 目录下。这是最常见的方式，将 web 应用的 WAR 文件或解压后的文件夹直接放置在 Tomcat 的 webapps 目录中，Tomcat 会自动部署该应用。

B、修改 tomcat 的 server.xml （在 tomcat/conf/server.xml）文件，在<Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true">的下方加上一句：<Context path="/" docBase="C:\

\Users\\DYB\\Desktop\\testspring\\out\\artifacts\\testspring_war_exploded" debug="0" reloadable="false" />

其中，docBase 为编译输出所在目录，也就是你要设置为 tomcat 根目录的文件路径。这样 Host 标签中的 webapps 文件路径就自动失效，docBase 目录将自动生效。

C、创建一个 XML 文件，并配置 web 应用信息，将 XML 文件放置在 Tomcat 的 conf/Catalina/localhost 目录下。这种方式允许对单个应用进行个性化配置，可以在该 XML 文件中指定应用的上下文路径、部署路径等信息。

26、下面方法中，用于获取 ServletContext 对象的方法是（） 5 分

✓A、getServletConfig().getServletContext()

✓B、getServletContext()

C、getServlet().getServletContext()

D、response.getServletContext()

27、阅读下面配置 web 默认页面 index.html 的代码

```
<welcome-file-lists>
    <welcome-files>index.html</welcome-files>
</welcome-file-lists>
```

下面选项中，说法正确的是（） （多选） 5 分

✓A、第一行配置应改为<welcome-file-list>

B、第二行配置正确

C、第一行，第三行配置正确

✓D、应将第二行的开始与结束标签都改为 welcome-file

welcome-file-list 是一个配置在 web.xml 中的一个欢迎页，用于当用户在 url 中输入工程名称或者输入 web 容器 url（如 http://localhost:8080/）时直接跳转的页面.

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

28、下面关于 Servlet 的多重映射的说法中，正确的是 （） 5 分

✓A、可以配置多个<servlet-mapping>来实现

✓B、可以在一个<servlet-mapping>配置多个<url-pattern>来实现

C、可以在一个<url-pattern>配置多个<servlet-mapping>来实现

D、以上都不正确

29、以下关于请求方式 GET 和 POST 的描述中，哪些是正确的（ ）。 5 分

A、使用 GET 请求方式传入的参数没有数据大小限制

✓B、使用 POST 请求方式传入的参数没有数据大小限制

C、使用 GET 请求方式提交的数据在地址栏中不会显示

✓D、使用 POST 请求方式提交的数据在地址栏中不会显示

30、下面关于 XML Schema 约束文档中命名空间标准的描述，正确的是（ ）（多选） 5 分

✓A、一个 XML 中可以引入多个名称空间

✓B、可以使用 xmlns 来声明引用名称空间的前缀

✓C、引入的名称空间可以不指定前缀，即声明默认名称空间

✓D、不同的命名空间可以区分同名的元素

A、一个 XML 中可以引入多个命名空间。通过在 XML 文档中使用 xmlns 声明，可以引入多个命名空间，并将不同的元素分组到不同的命名空间中。

B、可以使用 xmlns 来声明引用命名空间的前缀。通过在 XML 元素中使用 xmlns:prefix 的形式，可以为命名空间定义一个前缀，以便在元素中使用该前缀来引用命名空间。

C、引入的命名空间可以不指定前缀，即声明默认命名空间。通过在 XML 元素中使用 xmlns，而不指定前缀，可以为元素声明默认的命名空间，使该元素及其子元素都位于该命名空间中。

D、不同的命名空间可以区分同名的元素。通过使用不同的命名空间，即使有相同的元素名称，它们也被视为不同的元素，因为命名空间可以提供唯一性。

✓31、XML 文档声明的语法格式为：<?xml 版本信息[编码信息][文档独立性信息]?> 1 分

文档声明以符号“<?”开头，以符号“?>”结束，中间可以声明版本信息，编码信息以及文档独立性信息。

XML 文档声明的语法格式为：

<?xml version="版本信息" encoding="编码信息" standalone="文档独立性信息"?>

其中，版本信息是必需的，用于指定 XML 版本号，例如 "1.0" 或 "1.1"。编码信息是可选的，用于指定 XML 文档的字符编码方式，例如 "UTF-8" 或 "ISO-8859-1"。文档独立性信息也是可选的，用于指示 XML 文档是否依赖外部文档，取值可以是 "yes" 或 "no"。

✗32、统计网站当前在线人数的计数器变量 count 变量，应该保存在 HttpSession 域对象中。 1 分

✓33、在 Servlet 程序中，只有属于同一个请求中的数据才可以通过 HttpServletRequest 对象传递。 1 分

✓34、在 HTTP 的请求消息中，最常用的就是 GET 和 POST 方式。　1 分

✓35、配置 JAVA_HOME 环境变量的具体步骤为，打开环境变量，新建 JAVA_HOME 参数并配置值为 JDK 安装目录的 bin 目录下。　1 分

✓36、一个完整的 Servlet 事件监听器包括 Listener 类和<listener>配置。　1 分

✓37、当某个 Web 应用没有缺省 Servlet 时，也会使用 Tomcat 已配好的 DefaultServlet 作为默认缺省的 Servlet。　1 分

✓38、自定义 xml 文件配置虚拟目录时，xml 文件名将作为应用名，所以访问时的应用名为 xml 的文件名。　1 分

✓39、在 XML 文档中可以使用名称空间和不使用名称空间两种方式引入 XML Schema 文档。　1 分

✗40、属性是对标记进一步的描述和说明，一个标记只能有一个属性。　1 分

软工 2023 春季第二次作业

题目数量:30 道分值:100 分截止时间：2023-04-24 20:00:00

1、下列关于<dispatcher>元素值 FORWARD 的作用，描述正确的是（ ） 3 分

A、表示用户直接访问页面时，Filter 将调用

B、目标资源通过 RequestDispatcher 的 include()方法访问时，Filter 将被调用

✓C、目标资源通过 RequestDispatcher 的 forward()方法访问时，Filter 将被调用

D、目标资源是通过声明式异常处理机制调用时， Filter 将被调用

当目标资源通过 RequestDispatcher 的 forward()方法访问时，<dispatcher>元素值为 FORWARD 的 Filter 将被调用。这意味着在 Servlet 中使用 forward()方法将请求转发到另一个资源时，与该资源相关联的过滤器将被调用。过滤器可以在转发过程中对请求进行处理或修改。

2、下列选项中，哪个是服务器向客户端发送 Cookie 的本质？（ ） 3 分

✓A、在 HTTP 响应头字段中增加 Set-Cookie 响应头字段

B、在 HTTP 响应头字段中增加 Cookie 响应头字段

C、在 HTTP 请求头字段中增加 Cookie 响应头字段

D、在 HTTP 请求头字段中增加 Set-Cookie 响应头字段

服务器向客户端发送 Cookie 的本质是通过在 HTTP 响应头字段中增加 Set-Cookie 响应头字段来实现。当服务器发送响应时，通过设置 Set-Cookie 头字段，将 Cookie 信息发送给客户端。客户端在接收到包含 Set-Cookie 头字段的响应后，会将该 Cookie 存储起来，并在后续的请求中将 Cookie 通过 Cookie 请求头字段发送回服务器。

3、下列选项中，哪个元素用于指定 Filter 拦截的资源被容器调用的方式（ ） 3 分

A、<filter-name>

B、<url-pattern>

C、<filter-class>

✓D、<dispatcher>

在<dispatcher>元素中，可以使用多个值来指定 Filter 拦截器在何种情况下被调用。常见的取值包括 REQUEST（默认值，表示通过 HTTP 请求调用）、FORWARD（表示通过 RequestDispatcher 的 forward()方法调用）、INCLUDE（表示通过 RequestDispatcher 的 include()方法调用）、ERROR（表示通过错误页面调用）等。这些值可以单独使用，也可以组合使用。

4、下面选项中，用于监听 ServletContext 对象中属性变更的接口是（） 3 分

A、HttpSessionAttributeListener

✓B、ServletContextAttributeListener

C、ServletRequestAttributeListener

D、ApplicationAttributeListener

当 ServletContext 对象中的属性被添加、修改或删除时，ServletContextAttributeListener 会触发相应的事件，并执行相应的操作。

5、下面选项中，哪个方法可以用于设置 Cookie 的有效域（） 3 分

A、String setPath(String pattern)

B、void setPath(String pattern)

✓C、void setDomain(String pattern)

D、String setDomain(String pattern)

选项 C、void setDomain(String pattern) 可以用于设置 Cookie 的有效域。该方法接受一个字符串参数，用于设置 Cookie 的有效域。通过设置有效域，可以控制 Cookie 在哪些域名下可见和访问。

6、下面选项中，能够实现将用户会话中的“counter”计数器的值增加 1 的选项是（） 3 分

A、HttpSession session = request.getSession(true); int ival = session.getAttribute(“counter”); if(ival == null){ ival = 1; }else{ Ival = ival + 1; } session.setAttribute (“counter”, ival);

B、HttpSession session = request.getSession (true); Integer ival = (Integer) session.getAttribute (“counter”); Session.setAttribute (“counter”, ival + 1);

✓C、HttpSession session = request.getSession (true); Integer ival = (Integer) session.getAttribute (“counter”); if(ival == null){ Ival = new Integer (1); } else { Ival = new Integer (ival.intValue () + 1); } session.setAttribute (“counter”, ival);

D、HttpSession session = request.getSession (); int ival = session.getAttribute(“counter”); if (ival == null){ ival = 1; } else { ival = ival + 1; }session.setAttribute (“counter”, new Integer (ival));

request.getSession (true)：与此请求关联的 HttpSession 或 null ，如果 create 是 false 并且请求没有有效会话返回 null

void setAttribute(String name, Object value)：第二个参数只能是对象

7、下列配置中，表示过滤器拦截所有用户请求访问的是（ ） 3 分

A、<url >/*</url>

✓B、<url-pattern>/*</url-pattern>

C、<url>/*</url>

D、<url-pattern>*</url-pattern>

8、下面选项中，关于获得 HttpSession 对象的说法正确的是（）　3 分

A、用 new 语句创建一个 HttpSession 对象

✓B、调用 ServletRequest 对象的 getSession()方法

C、调用 ServletConfig 对象的 getSession()方法

D、以上说法都不对

要获得 HttpSession 对象，通常可以使用 ServletRequest 对象的 getSession()方法。这个方法会检查当前请求是否已经有关联的 HttpSession 对象，如果有则返回该对象，如果没有则创建一个新的 HttpSession 对象并返回。通过这种方式，可以获取到当前请求的 HttpSession 对象，从而进行会话管理和数据存储操作。

9、下面选项中，关于统计网站当前在线人数的计数器 count 变量应该保存的域范围是（）　3 分

A、request

B、session

✓C、application

D、page

关于统计网站当前在线人数的计数器 count 变量应该保存在域范围为 C、application（应用程序域）中。计数器用于统计网站当前在线人数，这是一个全局的统计数据，应该在整个应用程序范围内共享。因此，应该将计数器变量保存在应用程序域中（也称为 ServletContext 域）。应用程序域是在应用程序启动时创建的，并在整个应用程序的生命周期中存在。

10、下列选项中，关于 session 保存数据的位置，说法正确的是（）　3 分

A、数据保存在客户端

✓B、数据保存在服务器端

C、数据保存在客户端与服务器端各一份

D、以上说法都不对

Session 是一种在服务器端保存用户状态信息的机制，它通过在服务器端存储数据来跟踪和维护用户会话状态。当用户与服务器建立会话时，服务器会为每个会话创建一个唯一的会话 ID，并将该会话 ID 发送到客户端（通常是通过 Cookie 或 URL 重写）。在会话期间，服务器使用会话 ID 来标识和检索与特定用户相关联的数据。这些数据存储在服务器的内存中或持久化到磁盘上，以便在用户的后续请求中访问和使用。

11、下面选项中，属于 Servlet 事件监听器的是（　）　5 分

✓A、用于监听域对象创建和销毁的事件监听器

✓B、用于监听域对象属性增加和删除的事件监听器

C、用于监听绑定到 ServletContext 域中某个对象状态的事件监听器

✓D、用于监听绑定到 HttpSession 域中某个对象状态的事件监听器

Servlet 规范中定义的一种特殊类，用于监听 web 应用程序中的 ServletContext、HttpSession 和 ServletRequest 等域对象的创建与销毁事件，以及监听这些域对象中的属性发生修改的事件

Servlet 监听器的分类：

按监听的事件类型 Servlet 监听器可以分为如下三种类型：

1、域对象（request、session、application）监听器：监听域对象自身的创建和销毁的事件监听器；原来域对象的创建和销毁我们无法参与，监听器的出现提供了参与的机会

2、属性监听器：监听域对象中的属性的增加和删除的事件监听器；

3、HttpSession 域内对象监听器：监听绑定到 HttpSession 域中的某个对象的事件监听器。

12、下面选项中，构成事件监听过程的是（ ） 5 分

✓A、事件

✓B、事件源

✓C、Listener

✓D、事件处理器

Servlet 事件监听器概述

在监听的过程中会涉及几个重要组成部分：

1）事件：用户的一个操作，如单击一个按钮、调用一个方法、创建一个对象。

2）事件源：产生事件的对象。

3）事件监听器：负责监听发生在事件源上的事件。

4）事件处理器：监听器的成员方法，当事件发生的时候会触发对应的处理器

事件监听器在进行工作时，可以分为几步：

1）将监听器绑定到事件源，也就是注册监听器。

2）事件发生时会触发监听器的成员方法，即事件处理器，传递事件对象。

3）事件处理器通过事件对象获得事件源，并对事件源进行处理。

根据监听事件不同可以将其分为三类

1）用于监听域对象创建和销毁的事件监听器(ServletContextListener 接口、HttpSessionListener 接口、ServletRequestListener 接口)。

2）用于监听域对象属性增加和删除的事件监听器(ServletContextAttributeListener 接口、HttpSessionBindingListener 接口、ServletRequestAttributeListener 接口)。

3）用于监听绑定到 HttpSession 域中某个对象状态的事件监听器(HttpSessionBindingListener 接口、HttpSessionActivationListener 接口)。

在 Servlet 规范中，这三类事件监听器都定义了相应的接口，在编写事件监听器程序时只需实现对应的接口就可以。Web 服务器会根据监听器所实现的接口，把它注册到被监听的对象上，当触发了某个对象的监听事件时，Web 容器会调用 Servlet 监听器与之相关的方法对事件进行处理。

13、下列选项中，属于 URL 重写的方法是（） （多选） 5 分

✓A、encodeURL(String url)

✓B、encodeRedirectURL(String url)

C、encodeForwardURL(String url)

D、encodeRedirect (String url)

从 Servlet3.0 规范看，这两个方法的功能类似，但略有差别，规范是这么描述的：

encodeRedirectURL(java.lang.String url)

Encodes the specified URL for use in the sendRedirect method or, if encoding is not needed, returns the URL unchanged.

对给定的 url 进行编码，以用于 sendRedirect 方法；如果不需要编码，则直接返回(未经修改的)url。

encodeURL(java.lang.String url)

Encodes the specified URL by including the session ID in it, or, if encoding is not needed, returns the URL unchanged.

对给定的 url，通过加上 session ID 的方式进行编码；如果不需要编码，则直接返回(未经修改的)url。

看了文档仍然弄不清两者的区别。但是 servlet3.0 只是一个规范，只有 API(interface)没有实现代码。查看 tomcat7（实现 serlvet3.0 规范）的 org.apache.catalina.connector.Response 的代码，有如下结论

共同点：

都对 url 附加上 jsessionid 参数进行了处理，如果需要，则在 url 的 path 后面附加上;jsessionid=xxx；如果不需要则直接返回传入的 url。

不同点：

encodeURL 在附加 jsessionid 之前还对 url 做了判断处理：如果 url 为空字符串(长度为 0 的字符串)，则将 url 转换为完整的 URL(http 或 https 开头的)；如果 url 是完整的 URL，但不含任何路径(即只包含协议、主机名、端口，例如 http://127.0.0.1)，则在末尾加上根路径符号/。

也就是 encodeURL 如果进行了编码，则返回的 URL 一定是完整 URL 而不是相对路径；而 encodeRedirectURL 则不对 URL 本身进行处理，只专注于添加 jsessionid 参数（如果需要）。

14、下面选项中，属于 javax.servlet.FilterConfig 中定义的方法是（ ） 5 分

✓A、getFilterName()

✓B、getServletContext()

✓C、getInitParameter(java.lang.String name)

✓D、getInitParameterNames()

15、下列关于 URL 重写的描述中，正确的是 （） （多选） 5 分

A、如果浏览器没有禁用 cookie,那么在浏览器中访问时就不会重写 URL

✓B、如果浏览器禁用 cookie 了，那么浏览器每次访问时都会重写 URL

✓C、如果浏览器没有禁用 cookie,那么在浏览器只在第一次访问时才会重写 URL，以后每次访问时都不会重写 URL

D、不管浏览器是否禁用 cookie，都一定要重写 URL

URL 重写是一种在 URL 中添加额外信息的技术，通常用于在无状态的 HTTP 协议下实现会话管理。当浏览器禁用了 cookie 时，服务器无法通过 cookie 来跟踪用户会话，因此会使用 URL 重写的方式将会话信息添加到 URL 中。所以在禁用 cookie 的情况下，浏览器每次访问都会重写 URL 以包含会话信息。当浏览器没有禁用 cookie 时，服务器在第一次访问时会将会话信息添加到 URL 中，之后浏览器会自动在请求中包含 cookie，所以后续访问时不需要再重写 URL。

16、下列选项，生成一次性验证码所用到的主要类和接口的是 （） （多选） 5 分

✓A、BufferedImage

✓B、ImageIO

C、Graphic

D、IOImage

BufferedImage - 用于创建图像对象，可以用来绘制验证码图像。

ImageIO - 提供了读取和写入图像的方法，可以用来将生成的验证码图像保存为文件或输出到流。

17、下面选项中，与过滤器有关的接口是（ ） 5 分

✓A、javax.servlet.Filter

✓B、javax.servlet.FilterChain

✓C、javax.servlet.FilterConfig

D、javax.servlet.ServletConfig

Filter：过滤器是一个对象，它对资源请求（servlet 或静态内容）或资源响应或两者执行过滤任务。过滤器在 doFilter 方法中执行过滤。每个 Filter 都可以访问一个 FilterConfig 对象，从中它可以获取其初始化参数，以及对它可以使用的 ServletContext 的引用，例如，加载过滤任务所需的资源。

FilterChain：FilterChain 是 servlet 容器向开发人员提供的一个对象，用于查看已过滤的资源请求的调用链。过滤器使用 FilterChain 调用链中的下一个过滤器，或者如果调用过滤器是链中的最后一个过滤器，则调用链末尾的资源。

FilterConfig：servlet 容器用于在初始化期间将信息传递给过滤器的过滤器配置对象。

18、下面选项中，属于过滤器 Filter 接口中包含的方法有（ ） 5 分

✓A、init(FilterConfig filterConfig)

✓B、doFilter(ServletRequest req,ServletResponse resp,FilterChain chain)

✓C、destroy()

D、service(ServletRequest req,ServletResponse resp,FilterChain chain)

19、下列关于 URL 重写的描述中，正确的是（） （多选） 5 分

✓A、URL 重写可以在 URL 地址后跟上 JSESSIONID，浏览器即使禁用 cookie 也能在访问服务器时带回 JSESSIONID 的值，从而可以使用 session

✓B、response.encodeRedirectURL(java.lang.String url)和 response. encodeURL(java.lang.String url)都可以实现 URL 重写。

C、URL 重写不需要对所有地址都重写。

✓D、response. encodeURL(java.lang.String url)一旦发现浏览器带来了任何 cookie 信息就认为浏览器没有禁用 cookie，就不会再对传入的 URL 进行 URL 重写了

20、给定一个 Servlet 的代码片段如下所示

Public void doGet（HttpServletRequest request，HttpServletResponse resonse）throws ServletException，IOException{

ArrayList list=new ArrayList();

HttpSession session =request.getSession();

Session.setAttribute(“list”,list);

_____

}

要取出 session 中的值，下划线处的代码可以是（） （多选） 5 分

✓A、Object o=session.getAttribute(“list”);

✓B、Object o=(ArrayList)session.getParameter(“list”);

✓C、ArrayList list2=(ArrayList)session.getAttribute(“list”);

D、ArrayList list2=session.getParameter(“list”)

✓21、HttpSession 对象被创建时，将调用 HttpSessionListener 接口中的 sessionCreated ()方法。 2 分

✓22、Filter 进行初始化代码，就只能在 init()方法中编写，而不能在构造方法中编写。 2 分

✕23、在 web.xml 中，一个 <listener>元素中可以出现多个<listener-class>子元素。 2 分

✕24、Servlet 事件监听器只能监听 Web 应用程序中 ServletContext、HttpSession 和 ServletRequest 等域对象的创建和销毁过程。 2 分

✓25、FilterConfig 接口的 getFilterName()方法用于返回在 web.xml 文件中为 Filter 所设置的名称。 2 分

✓26、过滤器配置中，如果元素<url-pattern>使用通配符“*”来表示，则该过滤器将拦截所有的请求访问。 2 分

✕27、HttpSession 对象被销毁时，将调用 HttpSessionListener 接口中的 sessionCreated ()方法。 2 分

✓28、Filter 可以在访问目标资源之前，进行预处理操作。 2 分

Filter 可以在访问目标资源之前进行预处理操作。Filter 在请求到达目标资源之前拦截请求，并可以对请求进行修改、验证或者其他操作。这样可以实现一些通用的处理逻辑，例如身份验证、日志记录、字符编码转换等。通过在 Filter 中实现预处理逻辑，可以有效地对请求进行过滤和处理，确保请求的正确性和安全性。一旦预处理完成，Filter 可以将请求传递给下一个 Filter 或者目标资源。

✓29、ServletContext 对象被创建时，会调用 ServletContextListener 接口中的 contextInitialized()方法。 2 分

✓30、一次性验证码可以限制人们使用软件来暴力猜测密码，从而保证了用户信息的安全。 2 分

计科 第一次作业

题目数量:25 道分值:100 分截止时间：2023-04-08 19:05:29

1、XML 元素的属性与属性之间隔开采用的符号是（）。 3 分

✓A、A、空格

B、B、逗号

C、C、等号

D、D、双引号或单引号

2、下列选项中，启动 Tomcat 的命令 startup.bat，放在哪个目录中（） 3 分

✓A、A、bin

B、B、lib

C、C、webapps

D、D、work

3、下列选项中，修改 Tomcat 端口号的文件是（） 3 分

A、A、conf.xml

B、B、context.xml

✓C、C、server.xml

D、D、service.xml

4、下面选项中，在 tomcat 上发布 javaweb 应用时，默认在什么目录（）。 3 分

✓A、A、webapps

B、B、conf

C、C、bin

D、D、work

5、Schema 文档使用下列哪种语法编写（）。 3 分

A、A、HTML

✓B、B、XML

C、C、SGML

D、D、DTD

Schema 文档使用 XML 语法编写。 XML Schema 定义了 XML 文档的结构、数据类型和约束规则。它使用 XML 格式来描述所定义的元素、属性、数据类型等信息，以及它们之间的关系和约束。相比于 DTD (Document Type Definition)，XML Schema 提供了更强大和灵活的数据模型，并支持更多的数据类型定义和验证规则。

6、下列 XML 文档声明的格式中，正确的是（ ）。 3 分

✓A、A、<?xml version="1.0" encoding="GBK" ?>

B、B、<?xml version="1.0" encoding="GBK">

C、C、<!xml version="1.0"encoding="GBK" !>

D、D、<! --xml version="1.0" encoding="GBK" --!>

7、下列选项，关于 HTTP 消息描述正确的是（） 3 分

✓A、A、HTTP 请求消息和 HTTP 响应消息统称为 HTTP 消息

B、B、浏览器向服务器发送数据称为 HTTP 响应消息

C、C、服务器向浏览器发送数据称为 HTTP 请求消息

D、D、在 HTTP 消息中，所有信息对用户都是不可见的

8、关于配置 Path 环境变量的路径写法，正确的是（ ）。 3 分

A、A、"%JAVA_HOME%bin;"

✓B、B、"%JAVA_HOME%\bin;"

C、C、"%JAVA_HOME%\bin"

D、D、以上都不是

9、关于 Tomcat 的介绍，正确的是（ ）。 3 分

A、A、Tomcat 运行稳定并且可靠，但是效率比较低

B、B、Tomcat 不能作为独立的 Web 服务器软件

✓C、C、Tomcat 是 Apache 组织的 Jakarta 项目中的一个重要的子项目，它的源代码是完全公开的

D、D、Tomcat 中并没有提供数据库连接池的功能

10、下列选项中，哪一个可以限定 letter 的元素中可接受的值只能是字母 a-z 其中一个（ ）。 3 分

A、A、<xs:element name="letter"> <xs:simpleType> <xs:restrict base="xs:string"> <xs:pattern value="[a-z]"/> </xs:restrict> </xs:simpleType> </xs:element>

B、B、<xs:element name="later"> <xs:simpleType> <xs:restrict base="xs:string"> <xs:pattern value="[a

-z]"/> </xs:restrict> </xs:simpleType> </xs:element>

✓C、C、<xs:element name="letter"> <xs:simpleType> <xs:restriction base="xs:string"> <xs:pattern value="[a-z]"/> </xs:restriction> </xs:simpleType> </xs:element>

D、D、以上说法都不正确

通过使用 xs:restriction 元素和 xs:pattern 元素，可以对元素的取值进行限制。在这种情况下，使用正则表达式"[a-z]"来限制 letter 元素只能接受字母 a-z 中的一个。

11、关于 HTTP 1.1 优点的描述，下列说法正确的是（）。 10 分

✓A、A、客户端向服务器发送多个请求时，无需等待上次请求的返回结果

✓B、B、减少了浏览器与服务器交互所需的时间

✓C、C、在一个 TCP 连接上可以传送多个 HTTP 请求和响应

D、D、建立一个 TCP 连接后，只能传送 1 个 HTTP 请求和响应

HTTP 1.1 引入了持久连接（Persistent Connection）的概念，允许在一个 TCP 连接上传送多个 HTTP 请求和响应，减少了建立和关闭连接的开销，提高了网络传输的效率。

客户端与服务器建立连接之后，客户端可以向服务器端发送多个请求，并且在发送下个请求时，无须等待上次请求的返回结果。但是服务器必须按照接受客户端请求的先后顺序依次返回相应结果，以保证客户端能够区分出每次请求的响应内容。

12、下列通过哪个属性来使用名称空间引入 XML Schema 文档（）。 10 分

✓A、A、通过属性 xsi:schemaLocation 引入名称空间的文档

B、B、通过属性 xmlns:schemaLocation 来声明名称空间的文档

✓C、C、通过属性 xsi:noNamespaceSchemaLocation 属性直接指定

D、D、以上说法都不正确

在 XML 文档中引入 XML Schema 文档，不仅可以通过 xsi:schemaLocation 属性引入名称空间的文档，还可以通过 xsi:noNamespaceSchemaLocation 属性直接指定，noNamespaceSchemaLocation 属性也是在标准名称空间“http://www.w3.org/2001/XMLSchema-instance”中定义的，它用于定义指定文档的位置。

13、下面关于 POST 请求的说法中，正确的是（）。 10 分

✓A、A、post 方式传可以传输大数据

B、B、post 方式会将请求信息在地址栏上显示

✓C、C、post 方式不会将请求信息在地址栏上显示

✓D、D、post 方式提交数据相对于 get 方式安全些

14、下面关于响应头字段的说法中，正确的是（ ）。 10 分

A、A、Location 头字段中的 URL 值是一个使用相对路径的 URL 地址

✓B、B、Refresh 头字段的作用是告诉浏览器自动刷新页面的时间

✓C、C、使用 Content-Disposition 头字段可以让用户选择将响应的实体内容保存到一个文件中，而不是浏览器直接处理相应的实体内容

D、D、以上说法都不正确

15、关于 Schema 名称空间的描述中，正确的是（ ）。 10 分

✓A、A、如果有两个 URI 并且其组成的字符完全相同，就可以认为它们标识的是同一个名称空间

B、B、在声明名称空间时，可以同时有 xml 和 xmlns 两个前缀

✓C、C、名称空间声明的语法格式是<元素名 xmlns:prefixname="URI">

✓D、D、在声明名称空间时，xmlns 前缀仅用于声明名称空间的绑定

在声明名称空间时，有两个前缀是不允许的，它们是 xml 和 xmlns。

✓16、Tomcat 服务器默认的端口号是 8080。 2 分

✓17、如果 Tomcat 使用默认端口号，Tomcat 成功启动后，在浏览器地址栏中输入 http://localhost:8080 将能够访问 Tomcat 首页。 2 分

✓18、XML 文档的注释以字符串“<!--”开始，以字符串“-->”结束。 2 分

✓19、基于 HTTP 1.0 协议的客户端与服务器在交互的过程中需要经过建立连接、发送请求信息、回送响应信息、关闭连接 4 个步骤。 2 分

✓20、HTTP 响应状态行包括：HTTP 版本、一个表示成功或错误的整数代码（状态码）和对状态码进行描述的文本信息 3 个部分。 2 分

✓21、B/S 架构中，浏览器并不是直接与数据库服务器建立连接，而是通过 Web 服务器与数据库服务器需要建立连接。 2 分

✓22、在使用名称空间时，必须先声明名称空间。 2 分

✓23、HTTP 的状态码反应 Web 服务器处理客户端请求的状态，如果客户端显示的状态码是 500，表示服务器内部出现错误，无法处理请求。 2 分

✗24、客户端向服务器请求服务时，请求方式只有 GET、POST 两种。　2 分

在 HTTP 的请求消息中，请求方式有 GET、POST、HEAD、OPTIONS、DELETE、TRACE、PUT 和 CONNECT 共 8 种，每种方式都指明了操作服务器中指定 URI 资源的方式。

✓25、客户通常使用 HTML 表单向服务器的页面提交信息。　2 分

计科 第二次作业

题目数量:30 道分值:100 分截止时间：2023-06-12 15:09:42

1、下面用于获取文件上传字段中的文件名的方法是（）。 3 分

✓A、A、getName()

B、B、getType()

C、C、getContentType()

D、D、getString()

getName()方法用于获取文件上传字段中的文件名。 getContentType()方法用于获得上传文件的类型 getString()方法用于将 FileItem 对象中保存的数据流内容以一个字符串形式返回。

2、若想修改 Tomcat 服务器的默认会话时间，则需要进入下列哪个文件中修改？（ ） 3 分

A、A、在<tomcat 安装目录>\conf\context.xml 文件中修改

✓B、B、在<tomcat 安装目录>\conf\web.xml 文件中修改

C、C、在<tomcat 安装目录>\conf\server.xml 文件中修改

D、D、在<tomcat 安装目录> \conf\当前应用\web.xml 文件中修改

3、下列哪个方法不是 Filter 接口中定义的方法（）。 3 分

A、A、init()

B、B、doFilter()

✓C、C、help()

D、D、destroy()

init()方法是 Filter 的初始化方法。 doFilter()方法完成实际的过滤操作。 destroy() 该方法用于释放被 Filter 对象打开的资源。

4、下面 FileItem 类的方法中，用于获得上传文件的类型的方法是（）。 3 分

A、A、isFormField()

B、B、getFieldName()

✓C、C、getContentType()

D、D、getName()

getContentType()方法用于获得上传文件的类型，即表单字段元素描述头属性“Content-Type”的值，如“image/jpeg”。如果 FileItem 类对象对应的是普通表单字段，该方法将返回 null。

5、已知 web.xml 中存在如下配置： <session-config> <session-timeout>2</session-timeout> </session-config> 下面的说法，正确的是（）。 3 分

A、A、在空闲状态下，2 秒后将导致 session 对象销毁

✓B、B、在空闲状态下，2 分钟后将导致 session 对象销毁

C、C、在空闲状态下，2 毫秒后将导致 session 对象销毁

D、D、在空闲状态下，2 小时后将导致 session 对象销毁

在项目的 web.xml 文件中配置 Session 的失效时间单位为分钟。默认为 30 分钟

6、在 Java EE 中，定义了 getSession()方法的接口是（） 3 分

A、A、HttpServlet

B、B、HttpSession

✓C、C、HttpServletRequest

D、D、HttpServletResponse

7、下列对于 setMaxAge(-1)方法的描述中，正确的是（） 3 分

A、A、表示通知浏览器保存这个 Cookie 信息

B、B、表示通知浏览器立即删除这个 Cookie 信息

✓C、C、表示当浏览器关闭时，Cookie 信息会被删除

D、D、以上都不正确

8、在一个 Cookie 对象中，若调用了 setMaxAge(0)方法，表示（） 3 分

✓A、A、将 Cookie 的持久化时间设置为 0，意味着删除 Cookie

B、B、Cookie 永久生效

C、C、Cookie 在 10 分钟后失效

D、D、Cookie 在 30 分钟后失效

public void setMaxAge(int expiry)

设置此 Cookie 的最长使用期限（以秒为单位）。

正值表示 cookie 将在过了那么多秒后过期。请注意，该值是 cookie 过期的最大期限，而不是 cookie 的当前期限。

负值表示 cookie 未持久存储，并且在 Web 浏览器退出时将被删除。零值会导致 cookie 被删除。

9、下列选项中，正确设置 Set-Cookie 响应头字段的是（） 3 分

A、A、Set-Cookie; user=itcast; Path=/;

B、B、Set-Cookie user=itcast; Path=/;

✓C、C、Set-Cookie: user=itcast; Path=/;

D、D、Set-Cookie: user=itcast Path=/;

服务器向客户端发送 Cookie 时，会在 HTTP 响应头字段中增加 Set-Cookie 响应头字段。Set-Cookie 头字段中设

置的 Cookie 的具体示例如下： Set-Cookie: user=itcast; Path=/;

10、关于 Filter 的生命周期，下列说法错误的是 （）。 3 分

A、A、创建一个 Filter 对象之后，服务器调用 init()方法对该对象初始化

✓B、B、在 Filter 的生命周期中，init()方法将被执行很多次

C、C、Filter 的生命周期与其接口中的三个方法对应

D、D、在 Filter 的生命周期中，doFiltert()方法将被执行很多次

在一次完整的请求当中，Filter 对象只会创建一次，init()方法也只会执行一次。

11、创建 HttpSession 监听器后，会默认实现接口的监听器初始化和销毁两个方法，这两个方法是（）。 5 分

✓A、A、sessionCreated(HttpSessionEvent hts)

✓B、B、sessionDestroyed (HttpSessionEvent hts)

C、C、sessionInitialized (HttpSessionEvent hts)

D、D、httpSessionDestroyed(HttpSessionEvent hts)

12、下列选项中，适合将信息存入 Session 的是（） 5 分

✓A、A、用户登录信息

✓B、B、验证码

✓C、C、购物车

D、D、聊天室

13、关于 Commons-FileUpload 实现文件上传需要的 jar 文件有（）。 5 分

✓A、A、commons-fileupload.jar

✓B、B、commons-io.jar

C、C、commons-logging.jar

D、D、commons-lang.jar

Commons-FileUpload 实现文件上传需要将 commons-fileupload 和 commons-io 的 jar 包导入到项目

14、下面选项中，关于 SessionID 的说法正确的是（） 5 分

✓A、A、每个 HttpSession 对象都有唯一的 Session ID

✓B、B、SessionID 由 Servlet 容器创建

C、C、SessionID 必须保存在客户端的 cookie 文件中

✓D、D、Servlet 容器会把 Session ID 作为 Cookie 或者 URL 的一部分发送到客户端

15、下列关于 Cookie 的说法中正确的是（） 5 分

✓A、A、Cookie 是基于 HTTP 协议中的 Set-Cookie 响应头和 Cookie 请求头进行工作的

✓B、B、浏览器对 Cookie 具有严格的限制，一个网站能在浏览器中保存多少 Cookie 是有限制的

✓C、C、默认情况下 HttpSession 是基于一个名称为 JSESSIONID 的特殊 Cookie 工作的

✓D、D、一个浏览器可能保存着多个名称为 JSESSIONID 的 Cookie

16、下列关于 URL 重写的描述中，正确的是（） 5 分

A、A、如果浏览器没有禁用 cookie，那么在浏览器中访问时就不会重写 URL

✓B、B、如果浏览器禁用 cookie 了，那么浏览器每次访问时都会重写 URL

✓C、C、如果浏览器没有禁用 cookie,那么在浏览器只在第一次访问时才会重写 URL，以后每次访问时都不会重写 URL

D、D、不管浏览器是否禁用 cookie，都一定要重写 URL

17、按照监听对象分类，以下用于监听域对象创建和销毁的监听器有（）。 5 分

✓A、A、ServletContextListener

✓B、B、HttpSessionListener

C、C、ServletContextAtrributeListener

D、D、HttpSessionAtrributeListener

ServletContextListener 用于监听 ServletContext 对象的创建与销毁过程 HttpSessionListener 用于监听 HttpSession 对象的创建和销毁过程 ServletContextAttributeListener 用于监听 ServletContext 对象中的属性变更 HttpSessionAttributeListener 用于监听 HttpSession 对象中的属性变更

18、下列关于 HttpSession 对象的描述中，说法正确的是（） 5 分

✓A、A、如果两次访问时间间隔超过 session 定义的非活动时间间隔，则 HttpSession 对象将失效

B、B、用户每次做出请求时都会创建一个新的会话

✓C、C、同一个浏览器做出的多个请求可以访问同一个会话对象

D、D、SessionID 保存在服务器端，HttpSession 对象保存在客户的浏览器

19、下面关于 Session 域的说法中，正确的是（） 5 分

✓A、A、Session 域的作用范围为整个会话

B、B、Session 域中的数据只能存在 30 分钟，这个时间不能修改

✓C、C、可以调用 HttpSession 的 invalidate 方法，立即销毁 Session 域

✓D、D、当 web 应用被移除出 web 容器时，该 web 应用对应的 Session 跟着销毁

A、Session 域的作用范围为整个会话。Session 域是在客户端与服务器之间建立的一个会话，并且在整个会话期间保持有效。它可以用来存储和共享用户的会话数据，这些数据在不同的页面请求之间保持一致。

C、可以调用 HttpSession 的 invalidate 方法，立即销毁 Session 域。HttpSession 的 invalidate 方法用于立即销毁当前会话的 Session 域，并释放相关的资源。一旦调用 invalidate 方法，当前会话的 Session 将不再有效。

D、当 web 应用被移除出 web 容器时，该 web 应用对应的 Session 跟着销毁。当一个 web 应用被移除或卸载出 web 容器时，与该应用相关联的所有 Session 也将被销毁。这是因为 Session 是与应用程序相关联的，当应用程序被移除时，与之关联的 Session 也随之销毁。

20、下面选项中，是 Servlet3.0 中的注解的有（）。 5 分

✓A、A、@WebServlet

✓B、B、@WebFilter

C、C、@Servlet

✓D、D、@WebInitParam

Servlet 3.0 常见的注解主要有：@WebServlet、@WebFilter、@WebListener、@WebInitParam、@MultipartConfig、@ServletSecurity

✓21、用于监听 HttpSession 对象生命周期的接口是 HttpSessionListener。 2 分

✓22、一次性验证码可以限制人们使用软件来暴力猜测密码，从而保证了用户信息的安全。 2 分

✓23、Session 可以将会话数据保存到服务器。 2 分

✗24、Cookie 的 domain 属性是用来指定浏览器访问的域，设置 domain 属性时必须以"."开头。 2 分

出处：https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Headers/Set-Cookie#cookie_%E5%89%8D%E7%BC%80

// 当响应来自于一个安全域（HTTPS）的时候，二者都可以被客户端接受

Set-Cookie: __Secure-ID=123; Secure; Domain=example.com

Set-Cookie: __Host-ID=123; Secure; Path=/

补充：https://segmentfault.com/q/1010000005725117

不是必须的。前置 "." 是旧规范 RFC 2109 （已作废）的要求。

An explicitly specified domain must always start with a dot.

而当前规范 RFC 6265 说了 Set-Cookie 时，domain 属性的前置 "." 会被忽略

Note that a leading %x2E ("."), if present, is ignored even though that character is not permitted

因此第二种写法是没有问题的

直接使用 edge 打开 qq.com，可以看到 https://pacaio.match.qq.com/stat/only?callback=isNaN 响应头中包含

Set-Cookie:iip=0; Path=/; Domain=qq.com; Expires=Fri, 10 Jun 2033 15:14:07 GMT

✗ 25、FileItem 类的 getContentType()方法获取普通表单字段将抛出异常。 2 分

FileItem 类的 getContentType()方法获取普通表单字段不会抛出异常。getContentType()方法用于获取上传文件的内容类型，而不是普通表单字段。对于普通表单字段，应使用 isFormField()方法进行判断，并使用 getString()方法获取其值。getContentType()方法仅适用于获取上传文件的内容类型。

✗ 26、Cookie 的 domain 属性是用来指定浏览器访问的域，设置 domain 属性时严格区分大小写。 2 分

✓ 27、FileItem 类实现了序列化接口 Serializable，因此，FileItem 类支持序列化操作。 2 分

✓ 28、Tomcat 容器中如果将元素中的时间值设置成 0 或一个负数，则表示会话永不超时。 2 分

✓ 29、Servlet 提供了两个用于保存会话数据的对象，分别是 Cookie 和 Session。 2 分

✓ 30、为了防止上传文件名重复，在上传文件的名称前面可添加 UUID 前缀。 2 分

JavaEE（1）考试重点

JavaEE（1）考试重点：

1、请求转发好请求重定向，以及这两种技术的使用场景

2、SQL 注入

3、数据库连接以及数据库连接池

4、 HttpServletRequest 和 HttpServletResponse 头字段的操作

5、文件的上传和下载

6、Filter Path 的配置

7、C3P0 数据源

8、HttpServletRequest 参数的获取

9、使用数据库连接池插入数据到数据库

10、会话技术 Session 设置属性。

11、DBUtils 数据库的增删改查

12、PreparedStatement 数据库的增删改查

13、Filter 拦截与放行

1、请求转发好请求重定向，以及这两种技术的使用场景

P95,P88

2、SQL 注入

P217（预编译 PreparedStatement 对象）

3、数据库连接以及数据库连接池

P211(数据库连接) P234(DBCP 数据库连接池) P236(C3P0 数据库连接池)

4、HttpServletRequest 和 HttpServletResponse 头字段的操作

P86,P94

5、文件的上传和下载

P204

6、Filter Path 的配置

第一种：@WebFilter(“”)

1.具体的资源路径：/index.jsp 只有访问 index.jsp 资源时，过滤器才会被执行

2.目录拦截:/user/* 访问/user 下的所有资源时，过滤器都会被执行

3.后缀名拦截：*.jsp 访问所有后缀名 jsp 资源时，过滤器都会被执

```
4.拦截所有资源：/ * 访问所有资源时，过滤器都会被执行
第二种：<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app
_3_1.xsd"
         version="3.1">
    <filter>
        <filter-name>demo1</filter-name>
        <filter-class>Filter1</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>demo1</filter-name>
        //设置拦截路径
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

7、C3P0 数据源

P236

8、HttpServletRequest 参数的获取

P97

9、使用数据库连接池插入数据到数据库

10、会话技术 Session 设置属性。

P107(表格 5-2 各种常用方法)

11、DBUtils 数据库的增删改查

P244(动手实践：使用 DBUtils)

12、PreparedStatement 数据库的增删改查

```
import java.sql.*;
public class Example {
    public static void main(String[] args) {
        Connection conn = null;
```

```
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try {
            // 1. 建立数据库连接
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydatabase", "username",
"password");
            // 插入操作示例
            String insertQuery = "INSERT INTO users (id, name) VALUES (?, ?)";
            stmt = conn.prepareStatement(insertQuery);
            stmt.setInt(1, 1);
            stmt.setString(2, "John Doe");
            int rowsAffected = stmt.executeUpdate();
            System.out.println("插入成功，受影响的行数: " + rowsAffected);
            // 查询操作示例
            String selectQuery = "SELECT * FROM users";
            stmt = conn.prepareStatement(selectQuery);
            rs = stmt.executeQuery();
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                System.out.println("ID: " + id + ", Name: " + name);
            }
            // 更新操作示例
            String updateQuery = "UPDATE users SET name = ? WHERE id = ?";
            stmt = conn.prepareStatement(updateQuery);
            stmt.setString(1, "Jane Smith");
            stmt.setInt(2, 1);
            rowsAffected = stmt.executeUpdate();
            System.out.println("更新成功，受影响的行数: " + rowsAffected);
            // 删除操作示例
            String deleteQuery = "DELETE FROM users WHERE id = ?";
            stmt = conn.prepareStatement(deleteQuery);
            stmt.setInt(1, 1);
            rowsAffected = stmt.executeUpdate();
            System.out.println("删除成功，受影响的行数: " + rowsAffected);

            // 5. 关闭连接和资源
```

```
                stmt.close();
                conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            } finally {
                try {
                    if (rs != null) {
                        rs.close();
                    }
                    if (stmt != null) {
                        stmt.close();
                    }
                    if (conn != null) {
                        conn.close();
                    }
                } catch (SQLException e) {
                    e.printStackTrace();
                }
            }
        }
}
```

13、Filter 拦截与放行

放行：chain.doFilter(request,response);　　参照 P190

学习通实验

实验一 Servlet 基础

实验二 Serlvet 基础与会话技术

实验三 Servlet 高级技术

实验四 JDBC 和数据库连接池

实验一 Servlet 基础

~~上机一：（考察知识点为 Schema 约束）~~

~~根据下图的树状结构，设计一个国家 country.xsd 约束文档。~~

~~要求：国家的名称和城市的名称用 attribute。~~

~~最少有一个国家，每个国家下最少有一个城市。~~

参考答案：

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- 定义国家元素 -->
  <xs:element name="country">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="province" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="city" minOccurs="1" maxOccurs="unbounded" />
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="name" type="xs:string" use="required" />
    </xs:complexType>
  </xs:element>

</xs:schema>
```

参考解析：

1. 根元素：`country`

- `country` 元素是根元素，表示一个国家。
- 该元素具有一个必需的 `name` 属性，用于表示国家的名称。

2. 子元素：`province`
- `country` 元素下可以包含一个或多个 `province` 元素，表示国家的省份。
- 每个 `province` 元素具有一个必需的 `name` 属性，用于表示省份的名称。

3. 孙元素：`city`
- `province` 元素下可以包含一个或多个 `city` 元素，表示省份的城市。

根据该约束文件，可以创建符合约束的 XML 实例，如下所示：

```
<country name="China">
  <province name="Beijing">
    <city name="Beijing City" />
  </province>
  <province name="Hunan">
    <city name="Xiangtang City" />
    <city name="Changsha City" />
  </province>
</country>
```

在 `country.xsd` 约束文件中，包含了以下元素和属性的定义：

1. `xs:schema` 元素：
- `xs` 是 XML Schema 命名空间的前缀。
- `schema` 元素是根元素，用于定义整个约束文档的结构。
- 该元素没有特定的属性，因为它是顶级元素。

2. `xs:element` 元素：
- `element` 元素用于定义一个 XML 元素。
- `name` 属性指定元素的名称。
- `complexType` 子元素定义元素的复杂类型。

3. `xs:complexType` 元素：
- `complexType` 元素用于定义元素的复杂类型，即元素的结构。
- `sequence` 子元素用于指定元素内子元素的顺序和出现次数。

4. `xs:sequence` 元素：
- `sequence` 元素用于定义一组元素的顺序和出现次数。
- `element` 子元素用于定义一个子元素。
- `minOccurs` 属性指定子元素的最小出现次数。
- `maxOccurs` 属性指定子元素的最大出现次数。

5. `xs:attribute` 元素：
- `attribute` 元素用于定义元素的属性。

- `name` 属性指定属性的名称。
- `type` 属性指定属性的数据类型。
- `use` 属性指定属性的使用方式，可以是 `required`（必需）或 `optional`（可选）。

总体而言，`xs:schema` 元素用于定义整个约束文档的结构，`xs:element` 元素用于定义 XML 元素，`xs:complexType` 元素用于定义元素的复杂类型，`xs:sequence` 元素用于定义元素内子元素的顺序和出现次数，`xs:attribute` 元素用于定义元素的属性。

上机二：（考察知识点为 ServletContext 的使用）

编写一段程序，读取工程 WebContent/test 文件夹下的 hello.propertites(内容自己设计)文件信息，并把文件内容输出到控制台和页面。

参考答案：

```
package com.example.expire1.demo2;

import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;
import jakarta.servlet.*;

import java.io.*;
import java.util.*;

@WebServlet(value = "/expire1/demo2/content")
public class ContentServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, java.io.IOException {
        // 获取属性文件的输入流
        InputStream is = this.getServletContext()
            .getResourceAsStream("/WEB-INF/classes/WebContent/test/hello.properties");

        // 创建 Properties 对象并加载属性文件
        Properties properties = new Properties();
        properties.load(is);

        // 遍历属性文件中的键值对，并打印到控制台和响应中
        properties.forEach((k, v) -> {
```

```
            // 打印到控制台
            System.out.println(String.format("%s=%s", k, v));

            try {
                // 将键值对以 HTML 格式输出到响应中
                resp.getWriter().print(String.format("%s=%s<br>", k, v));
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        });
    }
}
```

参考解析：

逐行解释这段代码的含义：

1. `InputStream is = this.getServletContext().getResourceAsStream("/WEB-INF/classes/WebContent/test/hello.properties");`
   - `this` 是当前 Servlet 的实例。
   - `getServletContext()` 方法返回 Servlet 上下文对象，它提供了访问 Web 应用程序环境的方法。
   - `getResourceAsStream()` 方法用于获取位于 Web 应用程序中的资源的输入流。
   - 参数 "/WEB-INF/classes/WebContent/test/hello.properties" 指定了要获取的资源的路径。在这个例子中，它指定了一个名为 `hello.properties` 的属性文件。
2. `Properties properties = new Properties();`
   - 创建一个 `Properties` 对象，用于存储属性键值对。
3. `properties.load(is);`
   - 使用 `load()` 方法，从输入流 `is` 中加载属性文件的内容到 `Properties` 对象中。这将解析属性文件的内容，并将其存储在 `Properties` 对象中，使得可以通过键来访问属性值。

上机三：（考察知识点为请求重定向）

请按照以下要求编写程序。

1) 编写用户登录的界面 login.html 和登录成功的界面 welcome.html。（最简单的实现方式即可）

2) 编写处理用户登录请求的 Servlet 类 Login。

3) Login 类中判断表单中如果输入的用户名为“itcast”，密码为“itcast”，将请求重定向到 welcome.html 页面，否则重定向到 login.html 页面。

参考答案：

```
@WebServlet(value = "/expire1/demo2/login")
public class LoginServlet extends HttpServlet {
```

```
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletExcept
ion, IOException {
        resp.sendRedirect("/login.html");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletExcep
tion, IOException {
        String username = req.getParameter("username");
        String password = req.getParameter("pwd");
        if ("itcast".equals(username) && "itcast".equals(password)) {
            resp.sendRedirect("/welcome.html");
        } else {
            resp.sendRedirect("/login.html");
        }
    }
}
```

参考解析：

实验二 Serlvet 基础与会话技术

上机一：（考察知识点请求转发）

请按照以下要求编写程序：

1）编写一个名称为 ForwardServlet 的 Servlet 类，在类中向 request 对象中增加一些信息，然后使用 forward() 方法将请求转发到 ResultServlet 类中。

参考答案：

```
@WebServlet("/expire2/demo1/ForwardServlet")
public class ForwardServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletExcept
ion, IOException {
        req.setAttribute("custom", "test");
        req.getRequestDispatcher("/expire2/demo1/ResultServlet").forward(req, resp);
    }
}
```

参考解析：

2）编写一个名称为 ResultServlet 的 Servlet 类，在类中获取 request 对象中增加的信息并输出。

参考答案：

```
@WebServlet("/expire2/demo1/ResultServlet")
public class ResultServlet extends HttpServlet {


    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletExcept
ion, IOException {
        String customMsg = (String) req.getAttribute("custom");
        resp.getWriter().print(customMsg);
    }
}
```

参考解析：

~~上机二：（考察知识点为请求重定向）~~ ~~参考实验一上机三~~

~~请按照以下要求编写程序。~~

~~1) 编写用户登录的界面 login.html 和登录成功的界面 welcome.html。（最简单的实现方式即可）~~

~~2) 编写处理用户登录请求的 Servlet 类 Login。~~

~~3) Login 类中判断表单中如果输入的用户名为“itcast”，密码为“itcast”，将请求重定向到 welcome.html 页面，否则重定向到 login.html 页面。~~

上机三：（考察知识点 Session 实现用户自动登录）

在实际运用中的用户登陆网站中，多提供有记住密码和自动登陆等功能，方便同一用户短时间内不用再输入用户名和密码等繁琐信息可以快捷登陆。模拟用户自动登陆功能。

参考答案：

```
@WebServlet(value = "/expire2/demo3/KeepLoginServlet")
public class KeepLoginServlet extends HttpServlet {
```

```
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // 判断用户是否已经登录
        if (req.getSession().getAttribute("username") != null) {
            // 将用户名、会话 ID 和最近登录时间设置为请求属性
            req.setAttribute("username", req.getSession().getAttribute("username"));
            req.setAttribute("sessionID", req.getSession().getId());
            req.setAttribute("latestLoginTime",
                (new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")).format(req.getSession().getCreationTime()));
            // 将请求转发到 welcome.jsp 页面
            req.getRequestDispatcher("/WEB-INF/welcome.jsp").forward(req, resp);
            return;
        }
        // 如果用户未登录，则重定向到 login.html 页面
        resp.sendRedirect("/expire2/demo3/login.html");
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // 获取表单参数中的用户名和密码
        String username = req.getParameter("username");
        String password = req.getParameter("pwd");
        // 检查用户名和密码是否正确
        if ("itcast".equals(username) && "itcast".equals(password)) {
            // 如果用户名和密码正确，将用户名存储在会话中
            req.getSession().setAttribute("username", username);
            // 调用 doGet 方法处理 GET 请求
            doGet(req, resp);
        } else {
            // 如果用户名或密码错误，则重定向到 login.html 页面
            resp.sendRedirect("/expire2/demo3/login.html");
        }
    }
}
```

参考解析：

1. `@WebServlet(value = "/expire2/demo3/KeepLoginServlet")`
   - 这是一个注解，指示该 Servlet 将处理 URL 路径为"/expire2/demo3/KeepLoginServlet"的请求。
2. `public class KeepLoginServlet extends HttpServlet`
   - 这是一个 Java 类，继承自 HttpServlet 类，表示这是一个 Servlet 类用于处理 HTTP 请求。
3. `protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException`
   - 这是一个覆盖（override）了父类的 doGet 方法的方法。在这个方法中处理 HTTP GET 请求。
4. `protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException`
   - 这是一个覆盖了父类的 doPost 方法的方法。在这个方法中处理 HTTP POST 请求。
5. 在 doGet 方法中，首先检查用户是否已经登录：
   - 通过`req.getSession().getAttribute("username")`获取会话中存储的"username"属性值。
   - 如果"username"属性不为 null，表示用户已经登录。
   - 将用户名、会话 ID 和最近登录时间设置为请求属性，以便在转发到其他页面时使用。
   - 将请求转发到"/WEB-INF/welcome.jsp"页面，显示欢迎页面。
   - 如果用户未登录，则通过`resp.sendRedirect("/expire2/demo3/login.html")`将用户重定向到登录页面。
6. 在 doPost 方法中，处理用户提交的登录表单：
   - 通过`req.getParameter("username")`和`req.getParameter("pwd")`获取表单中的用户名和密码。
   - 检查用户名和密码是否正确（在这个例子中，用户名和密码都是"itcast"）。
   - 如果用户名和密码正确，将用户名存储在会话中，以保持登录状态。
   - 调用 doGet 方法处理 GET 请求，将用户重定向到欢迎页面。
   - 如果用户名或密码错误，通过`resp.sendRedirect("/expire2/demo3/login.html")`将用户重定向到登录页面。

实验三 Servlet 高级技术

上机一：（考察知识点 Filter 过滤器）

过滤器实现统一全站编码功能。

参考答案：

```
@WebFilter("/*")
public class EncodeFilter implements Filter {


    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        // 设置响应的内容类型为"text/html; charset=utf-8"
```

```
        response.setContentType("text/html; charset=utf-8");
        // 调用 FilterChain 的 doFilter 方法，继续处理请求和响应
        chain.doFilter(request, response);
    }
}
```

参考解析：

1. `@WebFilter("/*")`
   - 这是一个注解，指示该过滤器将应用于所有 URL 路径。
2. `public class EncodeFilter implements Filter`
   - 这是一个 Java 类，实现了 Filter 接口，表示这是一个过滤器类。
3. `public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)`
   - 这是一个覆盖（override）了 Filter 接口的 doFilter 方法的方法。在这个方法中执行过滤器的逻辑。
4. 在 doFilter 方法中，设置响应的内容类型为"text/html; charset=utf-8"：
   - 通过`response.setContentType("text/html; charset=utf-8")`设置响应的内容类型为 HTML，并指定字符编码为 UTF-8。
5. 调用 FilterChain 的 doFilter 方法，继续处理请求和响应：
   - 通过`chain.doFilter(request, response)`调用 FilterChain 的 doFilter 方法，将请求和响应传递给下一个过滤器或 Servlet 进行处理。

上机二：（考察知识点文件上传下载）

在上网时，上传文件的操作随处可见，例如将照片上传到空间，将文件保存到云盘等。为了提高上传效率，在进行上传时，都是将多个文件一起上传的，将文件上传后，可能还需要将所上传的文件下载下来使用。请参考该场景，编程实现

1、上传的图片文件保存在应用的 temp 文件夹下。

参考答案：

```
@WebServlet("/expire3/UploadServlet")
@MultipartConfig
public class UploadServlet extends HttpServlet {

    private static final String UPLOAD_DIRECTORY = "temp";

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // 重定向到上传页面
        resp.sendRedirect("/expire3/upload.html");
```

```
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException
{
        // 获取上传文件的存储路径
        String uploadPath = req.getServletContext().getRealPath(".") + File.separator + UPLOAD_DI
RECTORY;

        // 如果目录不存在则创建
        File uploadDir = new File(uploadPath);
        if (!uploadDir.exists()) {
            uploadDir.mkdir();
        }

        try {
            // 处理上传的文件
            for (Part file : req.getParts()) {
                // 将文件写入指定路径
                file.write(uploadPath + File.separator + file.getSubmittedFileName());
            }
        } catch (Exception e) {
            // 上传失败处理
            resp.getWriter().print("上传失败");
            System.err.println(e);
            return;
        }

        // 上传成功处理
        resp.getWriter().print("上传完成");
    }
}
```

参考解析：

1. `@WebServlet("/expire3/UploadServlet")`
   - 这是一个注解，指示该 Servlet 将处理 URL 路径为"/expire3/UploadServlet"的请求。
2. `@MultipartConfig`
   - 这是一个注解，指示该 Servlet 支持文件上传。

3. `public class UploadServlet extends HttpServlet`
   - 这是一个 Java 类，继承自 HttpServlet 类，表示这是一个 Servlet 类用于处理 HTTP 请求。
4. `private static final String UPLOAD_DIRECTORY = "temp"`
   - 这是一个常量，指定上传文件的存储目录。
5. `protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException`
   - 这是一个覆盖（override）了父类的 doGet 方法的方法。在这个方法中处理 HTTP GET 请求。
6. 在 doGet 方法中，将请求重定向到上传页面：
   - 通过`resp.sendRedirect("/expire3/upload.html")`将响应重定向到"/expire3/upload.html"页面，用于展示上传页面。
7. `protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException`
   - 这是一个覆盖了父类的 doPost 方法的方法。在这个方法中处理 HTTP POST 请求。
8. 在 doPost 方法中，获取上传文件的存储路径：
   - 通过`req.getServletContext().getRealPath(".") + File.separator + UPLOAD_DIRECTORY`获取存储路径。
   - `req.getServletContext().getRealPath(".")`获取当前 Web 应用的真实路径。
   - `File.separator`是文件分隔符，用于拼接路径。
   - `UPLOAD_DIRECTORY`是存储文件的目录名称。
9. 如果目录不存在则创建：
   - 通过`File uploadDir = new File(uploadPath)`创建 File 对象表示存储目录。
   - 通过`uploadDir.exists()`判断目录是否已存在。
   - 如果目录不存在，则通过`uploadDir.mkdir()`创建目录。
10. 尝试处理上传的文件：
    - 通过`req.getParts()`获取上传文件的 Part 对象列表。
    - 使用增强的 for 循环遍历每个上传文件的 Part 对象。
    - 通过`file.write(uploadPath + File.separator + file.getSubmittedFileName())`将文件写入指定路径。
11. 异常处理：
    - 如果上传过程中出现异常，通过`resp.getWriter().print("上传失败")`向响应输出上传失败的信息。
    - 通过`System.err.println(e)`将异常信息打印到控制台。
    - 使用`return`语句提前结束方法。
12. 处理上传成功：
    - 如果上传过程中没有出现异常，通过`resp.getWriter().print("上传完成")`向响应输出上传完成的信息。

这段代码实现了一个简单的文件上传功能。用户通过访问上传页面，选择要上传的文件，然后提交表单。在后台，Servlet 接收到文件上传的请求，将上传的文件写入指定的存储目录中。如果上传过程中出现异常，向响应输出上传失败的信息；如果上传成功，向响应输出上传完成的信息。

2、把上传成功的 txt 文件中的大写字母改成小写字母后，下载到客户端。

参考答案：

```
@WebServlet("/expire3/demo2/DownloadServlet")
public class DownloadServlet extends HttpServlet {

    private static final String UPLOAD_DIRECTORY = "temp";

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        // 获取上传文件的存储路径
        String uploadPath = req.getServletContext().getRealPath(".") + File.separator + UPLOAD_DIRECTORY;

        // 下载文件
        // 创建文件对象
        File file = new File(uploadPath + File.separator + "demo.txt");
        if (file.isFile() && file.canRead()) {
            // 设置响应的字符编码和内容类型
            resp.setCharacterEncoding("utf-8");
            resp.setContentType("application/octet-stream");

            // 设置响应头，指定文件的名称和长度
            resp.setHeader("Content-Disposition", String.format("attachment;filename=\"%s\"", file.getName()));
            resp.setHeader("Content-Length", String.valueOf(file.length()));

            // 创建输入流和输出流对象
            BufferedReader br = new BufferedReader(new FileReader(file, StandardCharsets.UTF_8));
            BufferedOutputStream bos = new BufferedOutputStream(resp.getOutputStream());

            // 读取文件内容并写入响应输出流
            for (int c = br.read(); c != -1; c = br.read()) {
                char ch = (char) c;
                bos.write(Character.toLowerCase(ch));
            }
```

```
            // 关闭流资源
            br.close();
            bos.close();
        }
    }
}
```

参考解析：

1. `String uploadPath = req.getServletContext().getRealPath(".") + File.separator + UPLOAD_DIRECTORY;`
   - 获取文件上传的存储路径。
   - `req.getServletContext().getRealPath(".")` 返回 Web 应用程序的根目录的真实路径。
   - `File.separator` 是文件路径分隔符。
   - 将存储目录和文件分隔符拼接起来，得到完整的上传文件路径。
2. `File file = new File(uploadPath + File.separator + "demo.txt");`
   - 创建一个文件对象，表示要下载的文件。
   - 在这个例子中，文件名为 "demo.txt"。
3. `if (file.isFile() && file.canRead()) { ... }`
   - 检查文件是否存在并且可读。
4. `resp.setCharacterEncoding("utf-8");`
   - 设置响应的字符编码为 UTF-8。
5. `resp.setContentType("application/octet-stream");`
   - 设置响应的内容类型为二进制流文件。
6. `resp.setHeader("Content-Disposition", String.format("attachment;filename=\"%s\"", file.getName()));`
   - 设置响应头，指定下载文件的名称。
   - 使用 `String.format()` 方法将文件名格式化为 `"attachment;filename=\"文件名\""` 的形式。
7. `resp.setHeader("Content-Length", String.valueOf(file.length()));`
   - 设置响应头，指定下载文件的长度。
8. `BufferedReader br = new BufferedReader(new FileReader(file, StandardCharsets.UTF_8));`
   - 创建一个用于读取文件的字符输入流。
   - 使用 `BufferedReader` 包装 `FileReader`，并指定字符编码为 UTF-8。
9. `BufferedOutputStream bos = new BufferedOutputStream(resp.getOutputStream());`
   - 创建一个用于写入响应输出流的字节输出流。
   - 使用 `BufferedOutputStream` 包装响应的输出流。
10. `for (int c = br.read(); c != -1; c = br.read()) { ... }`
    - 读取文件的内容并写入响应输出流。

- 使用 `BufferedReader` 的 `read()` 方法逐字符读取文件内容。
- 将读取的字符转换为小写字符，并写入响应输出流。

11. `br.close();`
    - 关闭输入流。
12. `bos.close();`
    - 关闭输出流。

这段代码定义了一个用于实现文件下载功能的 Servlet。它通过获取文件的存储路径，创建文件对象，设置响应的字符编码、内容类型和头部信息，然后将文件的内容逐字符写入响应的输出流中，实现文件的下载。

实验四 JDBC 和数据库连接池

程序开发中，经常需要实现用户权限控制，比如用户的注册、登陆和注销。模拟用户注册功能、登陆功能、修改密码和注销功能。

（1）用 PreparedStatement 方式实现用户注册功能。

参考答案：

```
package com.example.expire4;

import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;

import java.io.*;
import java.sql.*;



@WebServlet("/expire4/UserRegistration")
public class UserRegistration extends HttpServlet {

    private static final String DB_URL = "jdbc:mysql://localhost:3306/mydatabase";
    private static final String DB_USERNAME = "root";
    private static final String DB_PASSWORD = "root";

    protected boolean registerUser(String username, String password) throws ClassNotFoundExce
ption, SQLException {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn = DriverManager.getConnection(DB_URL, DB_USERNAME, DB_PASSWO
```

```
RD);
        String sql = "INSERT INTO users (username, password) VALUES (?, ?)";
        PreparedStatement stmt = conn.prepareStatement(sql);
        stmt.setString(1, username);
        stmt.setString(2, password);
        if (stmt.executeUpdate() != 0) {
            System.out.println(username + "用户注册成功");
            return true;
        }
        return false;
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) {
        try {
            if (this.registerUser(req.getParameter("username"), req.getParameter("password"))) {
                resp.getWriter().print("用户注册成功");
                return;
            }
            resp.getWriter().print("用户注册失败");
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletExcept
ion, IOException {
        this.doPost(req, resp);
    }

}
```

参考解析：

（2）用 C3p0 数据源实现用户登录功能。

参考答案：

```
@WebServlet("/expire4/UserLogin")
public class UserLogin extends HttpServlet {

    protected boolean loginUser(String username, String password) {
        ComboPooledDataSource dataSource = DataSourceSingleton.getDataSource();

        try (Connection conn = dataSource.getConnection()) {
            String sql = "SELECT * FROM users WHERE username = ? AND password = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, username);
            stmt.setString(2, password);
            ResultSet rs = stmt.executeQuery();
            return rs.next();
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletExcept
ion, IOException {
        this.doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletExcep
tion, IOException {
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        boolean loggedIn = this.loginUser(username, password);
        if (loggedIn) {
            resp.sendRedirect("/welcome.html");
            return;
        }
```

```
            resp.getWriter().print("登录失败");
        }
}
```

参考解析：

（3）用 C3p0+DBUtils 实现用户注销功能。

参考答案：

```
package com.example.expire4;

import com.mchange.v2.c3p0.ComboPooledDataSource;
import jakarta.servlet.*;
import jakarta.servlet.annotation.*;
import jakarta.servlet.http.*;
import org.apache.commons.dbutils.*;

import java.io.*;
import java.sql.*;



@WebServlet("/expire4/UserLogout")
public class UserLogout extends HttpServlet {

    protected boolean logoutUser(String username) {
        ComboPooledDataSource dataSource = DataSourceSingleton.getDataSource();
        try {
            Connection conn = dataSource.getConnection();
            QueryRunner queryRunner = new QueryRunner();
            String sql = "DELETE FROM users WHERE username = ?";
            queryRunner.update(conn, sql, username);
            System.out.println("用户退出成功");
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
```

```
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        this.doGet(req, resp);
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        if (this.logoutUser(req.getParameter("username"))) {
            resp.sendRedirect("/login-expire4.html");
        }
    }

}
```

参考解析：

（4）用 C3p0+PreparedStatement 实现用户修改密码功能。

参考答案：

```
@WebServlet("/expire4/ChangePassword")
public class ChangePassword extends HttpServlet {

    protected boolean changePassword(String username, String newPassword) {
        ComboPooledDataSource dataSource = DataSourceSingleton.getDataSource();

        try (Connection conn = dataSource.getConnection()) {
            String sql = "UPDATE users SET password = ? WHERE username = ?";
            PreparedStatement stmt = conn.prepareStatement(sql);
            stmt.setString(1, newPassword);
            stmt.setString(2, username);
            int rowsUpdated = stmt.executeUpdate();
            if (rowsUpdated > 0) {
```

```
                System.out.println("密码修改成功");
                return true;
            } else {
                System.out.println("用户名或密码错误");
                return false;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return false;
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        if (changePassword(req.getParameter("username"), req.getParameter("password"))) {
            resp.getWriter().print("修改成功");
            return;
        }
        resp.getWriter().print("修改失败");
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        this.doPost(req, resp);
    }

}
```

参考解析：

2022 年考纲

1、请求和响应（分析题，需要写代码或分析）

主要考请求，注意参数怎么传递，包括一个参数传一个或多个值，如何获取参数

乱码问题如何解决

重定向、请求转发和请求包含，以及他们的区别

2、安全问题

JDBC 操作那章 210

什么是 sql 注入，举例子，如何解决 sql 注入

1、参数获取

1. 通过 URL 查询参数传递参数：

```
// 示例 URL：http://example.com/myServlet?param1=value1&param2=value2

// 获取单个参数值
String param1 = request.getParameter("param1");

// 获取多个参数值
String[] param2Values = request.getParameterValues("param2");
```

2. 通过 POST 请求体中的表单参数传递参数：

```
// 设置请求体的字符编码为 UTF-8，确保正确解析参数数据
request.setCharacterEncoding("UTF-8");

// 获取单个参数值
String param1 = request.getParameter("param1");

// 获取多个参数值
String[] param2Values = request.getParameterValues("param2");
```

3. 通过请求头传递参数（自定义请求头）：

```
// 示例请求头：Custom-Header: value

// 获取单个请求头值
String headerValue = request.getHeader("Custom-Header");
```

请注意，在以上示例中，`request` 是 `HttpServletRequest` 对象，可以在 `doGet()` 或 `doPost()` 方法中获取。对于单个参数，可以使用 `getParameter()` 方法获取其值；对于多个参数，可以使用 `getParameterValues()` 方法获取参数值的数组。

2、乱码问题

P90、P99

3、重定向、转发和包含

在 `HttpServlet` 中，重定向、转发和包含是三种常见的请求处理和页面跳转方式。它们在功能和使用方式上有一些区别和联系：

1. 重定向（Redirect）：
   - 功能：重定向是通过发送一个特殊的响应给客户端来告诉它重新发送一个新的请求。客户端会根据重定向响应中的 URL 发起新的请求。
   - 使用方式：
     ```
     response.sendRedirect("newpage.jsp");
     ```
   - 特点：
     - 重定向是两次请求，第一次请求处理完毕后，服务器发送一个特殊的响应给客户端，然后客户端再发送一个新的请求。
     - 重定向可以跳转到不同的域名或服务器。
     - 重定向可以处理静态资源或其他 Web 应用的页面。
2. 转发（Forward）：
   - 功能：转发是在服务器内部将请求传递给其他资源进行处理，处理完毕后将结果直接返回给客户端。
   - 使用方式：
     ```
     request.getRequestDispatcher("newpage.jsp").forward(request, response);
     ```
   - 特点：
     - 转发是一次请求，请求和处理过程都在服务器内部完成。
     - 转发只能在同一个 Web 应用内进行，不能跨域名或服务器。
     - 转发可以传递请求和响应对象，共享数据和状态。
3. 包含（Include）：
   - 功能：包含是在当前页面中嵌入其他资源的内容，相当于在当前页面中插入其他页面的内容。
   - 使用方式：
     ```
     request.getRequestDispatcher("includedpage.jsp").include(request, response);
     ```
   - 特点：
     - 包含是一次请求，请求和处理过程都在服务器内部完成，但响应会将被包含的资源的内容合并到当前页面中。

- 包含只能在同一个 Web 应用内进行，不能跨域名或服务器。
- 包含可以传递请求和响应对象，共享数据和状态。

在使用这些方式时，需要根据具体需求和场景进行选择：

- 如果需要将请求重定向到不同的页面或服务器，可以使用重定向。
- 如果需要在服务器内部将请求传递给其他资源进行处理，然后将结果返回给客户端，可以使用转发。
- 如果需要将其他资源的内容嵌入到当前页面中，可以使用包含。

4、SQL 注入

JDBC SQL 注入是一种常见的安全漏洞，它发生在应用程序使用用户提供的数据直接构造 SQL 查询或语句时，而没有对用户输入进行适当的验证和转义。攻击者可以通过恶意构造的输入数据来修改 SQL 查询的结构，从而执行未经授权的数据库操作。

解决 JDBC SQL 注入的最佳方式是使用参数化查询（Prepared Statement）或使用安全的 ORM（对象关系映射）框架，如 Hibernate。下面是一些解决 JDBC SQL 注入的方法：

1. 参数化查询（Prepared Statement）：使用参数化查询可以将 SQL 查询和用户提供的数据分离开来，确保用户输入的数据不会直接插入到 SQL 查询中。参数化查询通过占位符（?）来表示需要插入的数据，然后使用绑定参数的方式将数据安全地传递给查询。示例代码如下：

```
String sql = "SELECT * FROM users WHERE username = ? AND password = ?";
PreparedStatement statement = connection.prepareStatement(sql);
statement.setString(1, username); // 绑定第一个参数
statement.setString(2, password); // 绑定第二个参数
ResultSet resultSet = statement.executeQuery();
```

2. 输入验证和转义：对用户输入进行合适的验证和转义，确保输入的数据符合预期的格式和内容，并且不包含恶意字符。可以使用输入验证技术（如正则表达式）对输入进行验证，并使用数据库提供的转义函数（如 `PreparedStatement` 的 `setString()` 方法）对特殊字符进行转义。

3. 使用安全的 ORM 框架：ORM 框架（如 Hibernate）可以自动处理 SQL 查询和参数的转义和拼接，减少了手动构造 SQL 查询的风险。ORM 框架会负责将实体对象与数据库记录进行映射，并自动处理参数化查询，从而避免了 SQL 注入的问题。

4. 最小权限原则：确保数据库连接使用的账户具有最小的权限，限制对数据库的操作范围，以降低潜在攻击者对数据库的影响。

补充内容

Tomcat 目录结构 P54