Java 应用系统开发
ServletContext 和 Web 配置

王晓东
wangxiaodong@ouc.edu.cn

中国海洋大学

November 26, 2018

学习目标

Java EE Web 应用需要部署在符合 Java EE 规范的 Web 容器中运行，如何取得 Web 应用本身的信息在编程中非常重要。

1. 掌握 Web 应用对象 ServletContext。
2. 了解 Web 应用的配置方法。
3. 掌握 MVC 模式 Web 开发中发挥核心作用的转发，区别**转发与重定向**。

大纲

接下来…

Web 应用环境对象

将 Web 应用部署到服务器上，启动 Web 服务器后，Web 容器为每个 Web 应用创建一个表达 Web 应用环境的对象（即 ServletContext 对象），并将 Web 应用的基本信息存储在这个 ServletContext 对象中。

❖ Web 应用环境对象的用途

- 所有 Web 组件都可以访问此 ServletContext 对象，进而取得 Web 应用的基本信息。
- ServletContext 还可以作为整个 Web 应用的共享容器对象，能够被所有会话请求共用，保存 Web 应用的共享信息。

Web 应用环境对象的生命周期

ServletContext 对象的生命周期与 Web 应用相同。

创建 Web 容器启动后，自动创建 ServletContext 对象；

销毁 Web 容器停止时，自动销毁 ServletContext 对象。

注意

如果在 ServletContext 对象中保存的对象信息需要长久保存，一般编写 ServletContext 对象的监听器，在此对象销毁之前将其中保存的对象数据进行持久化处理，如保存到数据库或者文件中。

Web 应用环境对象的生命周期

ServletContext 对象的生命周期与 Web 应用相同。

创建 Web 容器启动后，自动创建 ServletContext 对象；

销毁 Web 容器停止时，自动销毁 ServletContext 对象。

注意

如果在 ServletContext 对象中保存的对象信息需要长久保存，一般编写 ServletContext 对象的监听器，在此对象销毁之前将其中保存的对象数据进行持久化处理，如保存到数据库或者文件中。

Web 应用环境对象的类型和取得

Web 应用环境对象是接口**javax.servlet.ServletContext**的实现。

❖ 在 Servlet 内直接取得 ServletContext 接口对象

```
1    ServletContext ctx = this.getServletContext();
```

Web 应用环境对象的功能和方法

❖ Web 级数据共享容器

public void setAttribute(String name, Object object)

对象保存到 ServletContext。

```
1    ServletContext ctx = this.getServletContext();
2    ctx.setAttribute("userId", "Kevin");
3    ctx.setAttribute("age", 20); //自动完成 int 类型转换为 Integer 对象类型
```

public Object getAttribute(String name)

读取保存在 ServletContext 对象中指定名称的属性对象，不存在则返回 null。

```
1    String useId = (String) ctx.getAttribute("userId");
2    int age = (Integer) ctx.getAttribute("age"); //自动拆箱，将 Integer 转为 int
```

Web 应用环境对象的功能和方法

`public void removeAttribute(String name)`

将指定的属性从 ServletContext 对象中删除。

`Enumeration getAttributeNames()`

取得所有属性的名称列表，返回一个枚举器对象，可以用于遍历所有属性名称。

```
1    Enumeration nums = ctx.getAttributeName();
2    while (nums.hasMoreElements()) {
3      System.out.println(nums.nextElements());
4    }
```

注意 ServletContext 大量的方法请自行学习掌握。

接下来…

配置文件 web.xml

Web 的配置文件为**/WEB-INF/web.xml**，/WEB-INF 目录是**被 Web 服务器保护的目录**，客户端浏览器无法直接访问该目录下的任何文件，Struts、Spring 等框架都将配置文件保存在该目录下。

web.xml 的主要配置项

- Servlet 声明（Servlet）
- Servlet 映射（Servlet-mapping）
- Web 级初始参数（context-param）
- 过滤器（filter）
- 过滤器映射（filter-mapping）
- 监听器（listener）
- 异常跳转页面（error-page）
- MIME 类型映射（mime-mapping）
- 会话对象超时（session-config）
- 外部资源声明（resource-ref）
- 外部标记库描述符文件（taglib）

Web 初始参数配置

❖ Web 初始参数配置

```
1     <context-param>
2     <description>数据库驱动</description>
3     <param-name>driverName</param-name>
4     <param-value>sun.jdbc.odbc.jdbcOdbcDriver</param-value>
5     </context-param>
```

❖ Web 组件取得 Web 初始参数

在 Servlet 中可以通过 ServletContext 对象取得 Web 初始参数。

`public String getInitParameter(String name)` 取得指定名称的 Web 初始参数。

```
1     ServletContext ctx = this.getServletContext();
2     String driverName = ctx.getInitParameter("driverName");
```

会话超时配置

❖ 在 Java 代码中配置 HttpSession 对象的超时时间

```
1    HttpSession session = request.getSession();
2    session.setMaxInactiveInterval(15 * 60); // 设置会话超时为15分钟
```

❖ 在 Web 配置文件中进行会话超时配置[1]

```
1    <session-config>
2      <session-timeout>900</session-timeout>
3    </session-config>
```

[1]推荐在 web.xml 中配置会话超时。

接下来…

Servlet 配置对象 ServletConfig

- Java EE 为取得 Servlet 的配置信息，提供一个 Servlet 配置对象 API。
- 该对象在 Servlet 初始化阶段由 Web 容器实例化，并将当前 Servlet 的配置数据写入到此对象，供 Servlet 读取使用。

配置对象的类型和取得

配置对象类型 javax.servlet.ServletConfig 是一个接口，具体实现类由容器厂商实现。
ServletConfig 对象在 Servlet 的 init 方法中取得，由 Web 容器以参数方式注入到 Servlet：

```
1     private ServletConfig config = null;

3     public void init(ServletConfig config) throws ServletException {
4       super.init(config);
5       this.config = config;
6     }
```

1. 要取得 ServletConfig 对象需要重写 init 方法，并传递 ServletConfig 参数。然后在 doGet 和 doPost 方法中即可以使用 config 对象。
2. 与 ServletContext 和 HttpSession 对象不同，Web 容器为每个 Servlet 实例创建一个 ServletConfig 对象，不同 Servlet 之间无法共享此对象。

ServletConfig 功能和方法

public String getInitParameter(String name)

取得指定的 Servlet 配置参数。与 Web 初始参数不同，Servlet 初始参数在 Servlet 声明中定义。

```
<servlet>
  <servlet-name>ServletConfigSample</servlet-name>
  <servlet-class>ouc.javaee.ServletConfigSample</servlet-class>
  <init-parm>
    <param-name>url</param-name>
    <param-value>jdbc:oracle:thin:@210.30.108.5:1521:oracle</param-value>
  </init-parm>
</servlet>
```

☛ 注意 <init-param> 标签要放置在 <servlet-name> 和 <servlet-class> 后，否则编译错误。

❖ 取得配置的 Servlet 初始参数

```
String url = config.getInitParameter("url");
```

ServletConfig 功能和方法

`public Enumeration getInitParameterNames()`

取得所有 Servlet 初始化参数。

`public String getServletName()`

取得 Servlet 的名称。

`public ServletContext getServletContext()`

ServletConfig 对象提供了取得 ServletContext 对象的方法，与在 Servlet 内使用 this.getServletContext() 一样，返回 ServletContext 实例的对象引用。

```
1     ServletContext ctx = config.getServletConfig();
```

接下来…

Web 跳转方式

❶ 重定向（redirect）

典型的重定向跳转方式如下：

- 地址栏手工输入新的 URL 地址；
- 单击超链接；
- 提交 FORM 表单；
- 使用响应对象 response 的 sendRedirect() 方法。

重定向跳转方法都是由客户端浏览器来执行的，由此可见重定向增加了网络的访问流量。

Web 跳转方式

❷ 转发（forward）

- 转发是在服务器端进行页面直接跳转的方法。
- 转发是指 Web 组件在服务器端直接请求到另外 Web 组件的方式。
- 转发在 Web 容器内部完成，不需要通过客户端浏览器，因此客户端浏览器的地址还停留在初次请求的地址上。

Web 开发中应该尽量使用转发实现 Web 组件之间的导航。

实现转发

❖ 取得转发对象

转发对象类型为 javax.servlet.RequestDispatcher。

通过请求对象 HttpServletRequest 取得

```
1    RequestDispatcher rd = request.getRequestDispatcher("main.jsp");
```

使用 ServletContext 对象的方法取得

```
1    RequestDispatcher rd = this.getServletContext().getRequestDispatcher("/main.jsp");
```

实现转发

取得转发对象后，调用转发对象的方法 forward 完成转发。

```
1    RequestDispatcher rd = request.getRequestDispatcher("main.jsp");
2    rd.forward(request, response);
```

☛ 目标页面 main.jsp 的目录说明

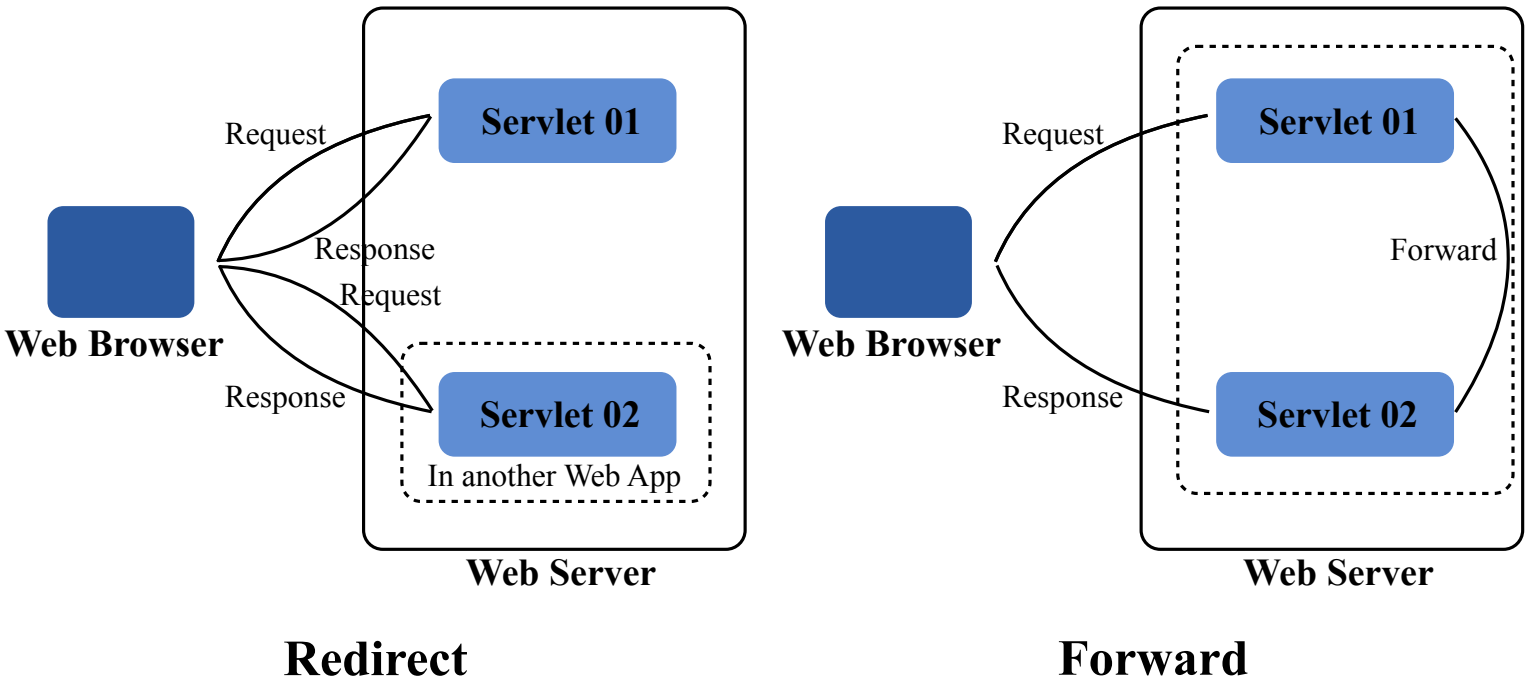
- 如果上述代码所在的 Servlet 被映射到/employee/main.action（一个虚拟请求地址），则转发的目标页面 main.jsp 也需要放到/employee 目录下；
- 如果 main.jsp 和 Servlet 映射地址不在同一目录，就需要使用相对路径定位，如把 main.jsp 放在/department/main.jsp，则取得转发对象需要按照如下示例代码进行修改：

```
1    RequestDispatcher rd =
2    request.getRequestDispatcher("../department/main.jsp");
```

Servlet 之间共享数据的方法总结

1. **使用** ServletContext **对象** 对象生命周期长，会长时间占用内存。
2. **使用会话对象** 对象生命周期较长，会长时间占用内存。
3. **使用请求对象，基于转发传递数据** 对象生命周期短，内存会及时释放。

转发与重定向的区别


Request
Servlet 01
Response
Request
Web Browser
Response
Servlet 02
In another Web App
Web Server
Redirect
Request
Servlet 01
Forward
Web Browser
Response
Servlet 02
Web Server
Forward

转发与重定向的区别

1. **发生的地点不同** 重定向由客户端完成，而转发由服务器完成。
2. **请求/响应的次数不同** 重定向两次请求，创建两个请求对象和响应对象，而转发是一次请求，只创建一个请求对象和响应对象。重定向无法共享请求/响应对象，而转发可以。
3. **目标位置不同** 重定向可以跳转到 Web 应用以外的文档，而转发只能在一个 Web 内部文件中间进行。

注意

转发之前不应有响应发送，否则导致异常 javax.servlet.IllegalStateException 抛出。

转发与重定向的区别

1. **发生的地点不同** 重定向由客户端完成，而转发由服务器完成。
2. **请求/响应的次数不同** 重定向两次请求，创建两个请求对象和响应对象，而转发是一次请求，只创建一个请求对象和响应对象。重定向无法共享请求/响应对象，而转发可以。
3. **目标位置不同** 重定向可以跳转到 Web 应用以外的文档，而转发只能在一个 Web 内部文件中间进行。

注意

转发之前不应有响应发送，否则导致异常 javax.servlet.IllegalStateException 抛出。

转发与重定向的区别

1. **发生的地点不同** 重定向由客户端完成，而转发由服务器完成。
2. **请求/响应的次数不同** 重定向两次请求，创建两个请求对象和响应对象，而转发是一次请求，只创建一个请求对象和响应对象。重定向无法共享请求/响应对象，而转发可以。
3. **目标位置不同** 重定向可以跳转到 Web 应用以外的文档，而转发只能在一个 Web 内部文件中间进行。

👉 注意

转发之前不应有响应发送，否则导致异常 javax.servlet.IllegalStateException 抛出。

接下来…

Web 应用环境对象

Java EE Web 的配置

Servlet 配置对象

转发和重定向

本节习题

本节习题

❖ 简答题

1. 什么是 Web 应用环境对象（ServletContext 对象），它的生命周期如何？
2. web.xml 文件能够完成 Web 应用的那些配置？
3. 什么是 Web 重定向和跳转？两者的区别是怎样的？

❖ 小编程

1. 将 HTTP 会话跟踪技术的小编程作业中的页面重定向改为采用转发实现。

THE END

wangxiaodong@ouc.edu.cn