Java Web Lesson 6

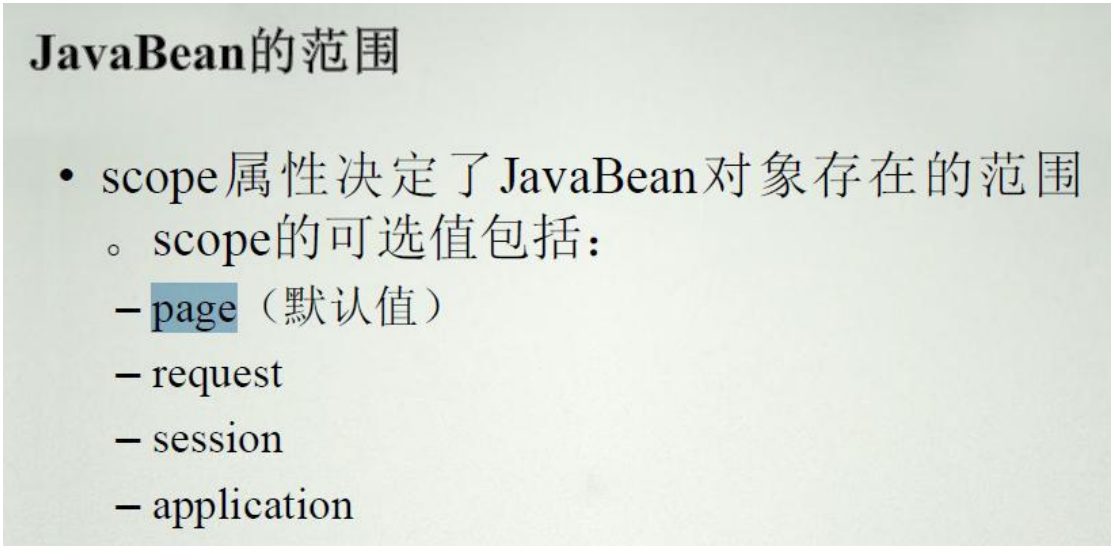
1. 可以在<jsp:setProperty>中通过 param 参数为 bean 的属性动态赋值。

```
<jsp:setProperty property="age" name="person" param="helloworld"/>
```

对应的 Servlet 代码是：

```
      org.apache.jasper.runtime.JspRuntimeLibrary.introspecthelper(_jspx_page
_context.findAttribute("person"), "age", request.getParameter("helloworld"),
request, "helloworld", false);
```

2. JavaBean 的存活范围：



3. 当将 JavaBean 放在 session 范围内时

```
<jsp:useBean id="student" scope="session"
    class="com.shengsiyuan.bean.Student" />
```

生成的 Servlet 源代码是：

```
      com.shengsiyuan.bean.Student student = null;
      synchronized (session) {
        student = (com.shengsiyuan.bean.Student) _jspx_page_context.getAt
tribute("student", PageContext.SESSION_SCOPE);
        if (student == null){
          student = new com.shengsiyuan.bean.Student();
          _jspx_page_context.setAttribute("student", student, PageContext
.SESSION_SCOPE);
        }
      }
```

4. 当将 JavaBean 放在 application 范围内时

```
<jsp:useBean id="student" scope="application"
    class="com.shengsiyuan.bean.Student" />
```

生成的 Servlet 源代码是：

```
      com.shengsiyuan.bean.Student student = null;
      synchronized (application) {
        student = (com.shengsiyuan.bean.Student) _jspx_page_context.getAttrib
ute("student", PageContext.APPLICATION_SCOPE);
        if (student == null){
          student = new com.shengsiyuan.bean.Student();
          _jspx_page_context.setAttribute("student", student, PageContext.APP
LICATION_SCOPE);
        }
      }
```

5. 每一个 Servlet 都必须要实现 Servlet 接口，GenericServlet 是个通用的、不特定于任何协议的 Servlet，它实现了 Servlet 接口，而 HttpServlet 继承于 GenericServlet，因此 HttpServlet 也实现了 Servlet 接口，所以我们定义的 Servlet 只需要继承 HttpServlet 父类即可。
6. Servlet 接口中定义了一个 service 方法，HttpServlet 对该方法进行了实现，实现方式就是将 ServletRequest 与 ServletResponse 转换为 HttpServletRequest 与 HttpServletResponse

```
    public void service(ServletRequest req, ServletResponse res)
        throws ServletException, IOException {

        HttpServletRequest  request;
        HttpServletResponse response;

        try {
            request = (HttpServletRequest) req;
            response = (HttpServletResponse) res;
        } catch (ClassCastException e) {
            throw new ServletException("non-HTTP request or response");
        }
        service(request, response);
    }
}
```

7. 转换完毕后，会调用 HttpServlet 类中自己定义的 service 方法，如下所示

```
protected void service(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {
```

8.在该 service 方法中，首先获得到请求的方法名，然后根据方法名调用对应的 doXXX 方法，比如说请求方法为 GET，那么就去调用 doGet 方法；请求方法为 POST，那么就去调用 doPost 方法。

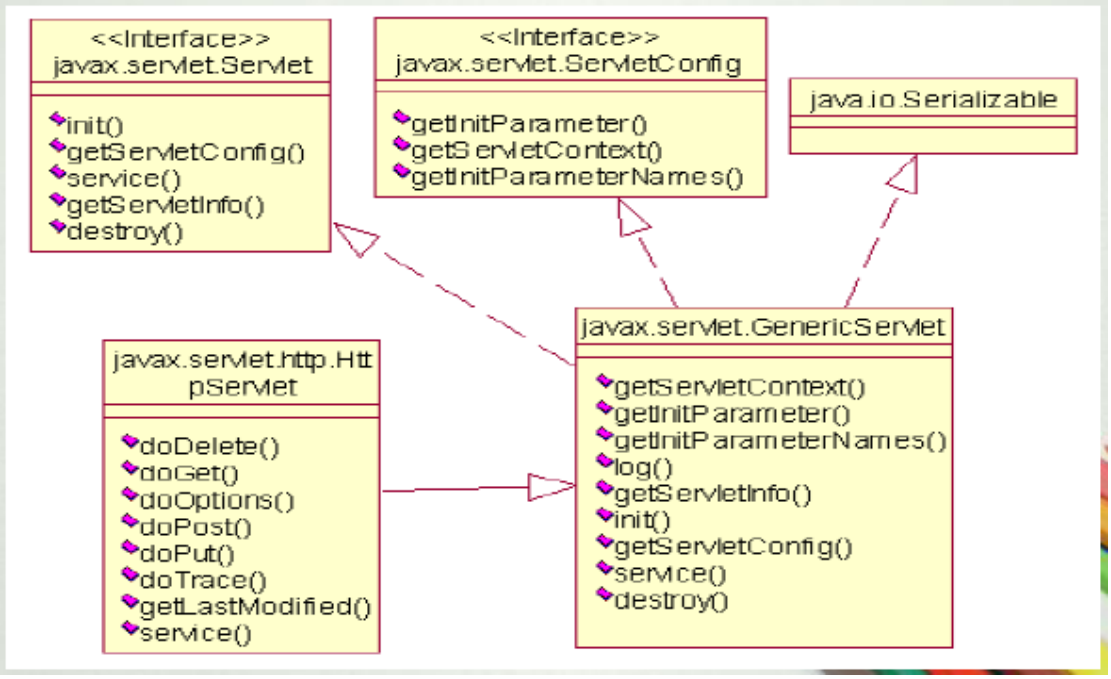
```
String method = req.getMethod();

if (method.equals(METHOD_GET)) {
    long lastModified = getLastModified(req);
    if (lastModified == -1) {
        // servlet doesn't support if-modified-since, no reason
        // to go through further expensive logic
        doGet(req, resp);
    } else {
        long ifModifiedSince = req.getDateHeader(HEADER_IFMODSINCE);
        if (ifModifiedSince < (lastModified / 1000 * 1000)) {
            // If the servlet mod time is later, call doGet()
            // Round down to the nearest second for a proper compare
            // A ifModifiedSince of -1 will always be less
            maybeSetLastModified(resp, lastModified);
            doGet(req, resp);
        } else {
            resp.setStatus(HttpServletResponse.SC_NOT_MODIFIED);
        }
    }
```

9. 在 HttpServlet 类中所提供的 doGet、doPost 等方法都是直接返回错误信息，所以我们需要在自己定义的 Servlet 类中 override 这些方法

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException
{
    String protocol = req.getProtocol();
    String msg = lStrings.getString("http.method_get_not_supported");
    if (protocol.endsWith("1.1")) {
        resp.sendError(HttpServletResponse.SC_METHOD_NOT_ALLOWED, msg);
    } else {
        resp.sendError(HttpServletResponse.SC_BAD_REQUEST, msg);
    }
}
```

10. 源码面前，了无秘密

11. Servelt 核心 API 之间的关系 UML 图：



12. Servlet 的启动

- 在下列时刻Servlet容器装载Servlet：
  - Servlet容器启动时自动装载某些Servlet
  - 在Servlet容器启动后，客户首次向 Servlet 发出请求
  - Servlet的类文件被更新后，重新装载Servlet
- Servlet被装载后，Servlet容器创建一个 Servlet 实例并且调用 Servlet 的 init()方法进行初始化。在Servlet的整个生命周期中，init方法只会被调用一次。