[实现一个简单的Servlet容器](http://blog.csdn.net/bingduanlbd/article/details/38349737)

2014-08-02 17:49 1457人阅读 [评论](http://blog.csdn.net/bingduanlbd/article/details/38349737#comments)(3) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/bingduanlbd/article/details/38349737#report)

版权声明：本文为博主原创文章，未经博主允许不得转载。

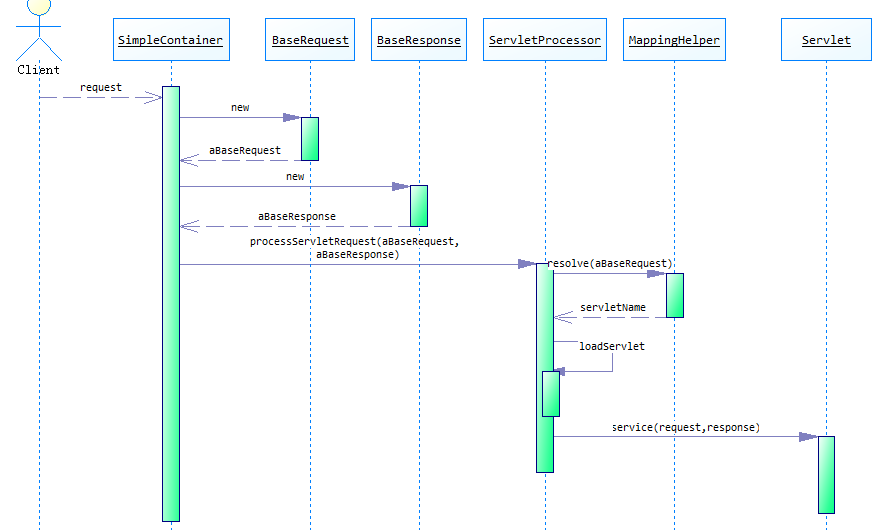
Servlet规范定义了一个API标准，这一标准的实现通常称为Servlet容器，比如开源的Tomcat、JBoss。容器提供一些基本的服务，使得开发者专注于业务领域。当一个请求到达时，监听在相应端口的容器会接受该请求，将请求的信息转化成一个ServletRequest对象，同时创建一个ServletResponse对象，我们在开发Servlet应用时，就可以直接使用这两个对象（容器传递到service（）方法的参数）。因此，所谓容器（服务器、中间件等），就是提供一些底层的、业务无关的基本功能，为真正的Servlet提供服务。

例如，在Http请求到达的时候，请求是一个标准的Http请求，包括请求方法、URI、协议、客户端环境参数、请求参数等等。容器通过Socket接收这些信息，并将其转化成特定于Servlet的ServletRequest对象。我们无需关注底层的网络Socket细节就可以直接使用。同样，容器也创建好一个ServletResponse，它帮我们设置了一些返回操作的基本内容，比如用于写入的OutputStream，容器已经把这个输出流定向到客户端的机器上，因此我们操作ServletResponse对象时，是在容器提供的服务至上的。

容器还负责根据请求的信息找到对应的Servlet，传递Request和Response参数，调用Servlet的service方法，完成请求的响应。

接下来我们事先一个简单的、最基本的容器：

时序图如下：



下面给出源码：

SimpleContainer:

**[java]** [view plain](http://blog.csdn.net/bingduanlbd/article/details/38349737) [copy](http://blog.csdn.net/bingduanlbd/article/details/38349737)

1. **package** simpleserver.simplecontainer;
3. **import** **static** simpleserver.simplecontainer.Constant.HOSTNAME;
4. **import** **static** simpleserver.simplecontainer.Constant.LOG\_BAKC;
5. **import** **static** simpleserver.simplecontainer.Constant.PORT;
7. **import** java.io.IOException;
8. **import** java.io.InputStream;
9. **import** java.io.OutputStream;
10. **import** java.net.InetAddress;
11. **import** java.net.ServerSocket;
12. **import** java.net.Socket;
14. **import** org.apache.commons.logging.Log;
15. **import** org.apache.commons.logging.LogFactory;
17. **public** **class** SimpleContainer {
19. **private** **boolean** shutdown = **false**;
20. **private** Log log = LogFactory.getLog(**this**.getClass());
22. **public** **static** **void** main(String[] args) {
23. SimpleContainer server = **new** SimpleContainer();
25. server.start();
27. }
29. **private** **void** start() {
30. ServerSocket serverSocket = **null**;
31. **try** {
32. serverSocket = **new** ServerSocket(PORT, LOG\_BAKC,
33. InetAddress.getByName(HOSTNAME));
34. } **catch** (IOException e) {
35. System.out.println("Server starts failed");
36. e.printStackTrace();
37. System.exit(1);
38. }
39. log.info("Server starts successfully.");
40. service(serverSocket);
42. }
44. **private** **void** service(ServerSocket serverSocket) {
45. **while** (!shutdown) {
46. **try** {
47. processRequest(serverSocket);
48. } **catch** (Exception e) {
49. e.printStackTrace();
50. }
51. }
53. }
55. **private** **void** processRequest(ServerSocket serverSocket) **throws** IOException {
57. log.info("waitting for incoming request ... ");
58. Socket socket = serverSocket.accept();
59. log.info("receive a request from "
60. + socket.getRemoteSocketAddress().toString());
61. InputStream in = socket.getInputStream();
62. BaseRequest request = **new** BaseRequest(in);
63. log.info("Request Object ready!");
65. OutputStream out = socket.getOutputStream();
66. BaseResponse response = **new** BaseResponse(out);
67. ServletProcessor.processServletRequest(request, response);
69. socket.close();
71. }
72. }

BaseRequest:只给出部分实现，其他都是ServletRequest的空实现

**[java]** [view plain](http://blog.csdn.net/bingduanlbd/article/details/38349737) [copy](http://blog.csdn.net/bingduanlbd/article/details/38349737)

1. **package** simpleserver.simplecontainer;
3. **import** java.io.BufferedReader;
4. **import** java.io.IOException;
5. **import** java.io.InputStream;
6. **import** java.io.UnsupportedEncodingException;
7. **import** java.util.Enumeration;
8. **import** java.util.Locale;
9. **import** java.util.Map;
11. **import** javax.servlet.AsyncContext;
12. **import** javax.servlet.DispatcherType;
13. **import** javax.servlet.RequestDispatcher;
14. **import** javax.servlet.ServletContext;
15. **import** javax.servlet.ServletInputStream;
16. **import** javax.servlet.ServletRequest;
17. **import** javax.servlet.ServletResponse;
19. **public** **class** BaseRequest **implements** ServletRequest {
21. **private** InputStream inputStream;
22. **private** String uri;
23. **private** StringBuffer requestContentBuffer = **new** StringBuffer(2048);
25. **public** BaseRequest(InputStream in) {
26. **this**.inputStream = in;
27. prepareContent();
28. parseAndSetUri();
29. }


33. **private** **void** prepareContent() {
35. **byte**[] buffer = **new** **byte**[2048];
36. **int** i = -1;
38. **try** {
39. i = inputStream.read(buffer);
40. } **catch** (IOException e) {
41. e.printStackTrace();
42. }
43. **for** (**int** k = 0; k < i; k++) {
44. requestContentBuffer.append((**char**) buffer[k]);
45. }
46. System.out.println(requestContentBuffer.toString());
48. }
50. **private** **void** parseAndSetUri() {
51. String requestString = requestContentBuffer.toString();
52. **int** index1 = requestString.indexOf(' ');
53. **int** index2 = -1;
54. **if** (index1 != -1) {
55. index2 = requestString.indexOf(' ', index1 + 1);
56. }
57. **this**.uri = (index2 > index1) ? requestString.substring(index1 + 1,
58. index2) : **null**;
60. }
62. **public** String getUri() {
63. **return** **this**.uri;
64. }
66. ......
67. }

BaseResponse:

**[java]** [view plain](http://blog.csdn.net/bingduanlbd/article/details/38349737) [copy](http://blog.csdn.net/bingduanlbd/article/details/38349737)

1. **public** **class** BaseResponse **implements** ServletResponse {
3. **private** OutputStream outputStream;
5. **public** BaseResponse(OutputStream out) {
6. **this**.outputStream = out;
7. }
8. ......
9. }

ServletProcessor:

**[java]** [view plain](http://blog.csdn.net/bingduanlbd/article/details/38349737) [copy](http://blog.csdn.net/bingduanlbd/article/details/38349737)

1. **package** simpleserver.simplecontainer;
3. **import** java.io.File;
4. **import** java.io.IOException;
5. **import** java.net.URL;
6. **import** java.net.URLClassLoader;
7. **import** java.net.URLStreamHandler;
9. **import** javax.servlet.Servlet;
10. **import** javax.servlet.ServletException;
11. **import** javax.servlet.ServletRequest;
12. **import** javax.servlet.ServletResponse;
14. **import** org.apache.commons.logging.Log;
15. **import** org.apache.commons.logging.LogFactory;
17. **public** **class** ServletProcessor {
19. **private** **static** Log log = LogFactory.getLog(ServletProcessor.**class**);
21. **public** **static** **void** processServletRequest(BaseRequest request,
22. BaseResponse response) {
23. String uri = request.getUri();
24. String servletName = MappingHelper.resolve(uri);
25. log.info("Processing servlet: " + servletName);
27. Servlet servlet = loadServlet(servletName);
29. callService(servlet, request, response);
31. }
33. **private** **static** URLClassLoader createUrlClassLoader() {
35. URLClassLoader loader = **null**;
36. **try** {
37. URL[] urls = **new** URL[1];
38. URLStreamHandler streamHandler = **null**;
39. File classPath = **new** File(Constant.RESOURCE\_ROOT);
41. // org.apache.catalina.startup.ClassLoaderFactory
42. String repository = (**new** URL("file", **null**,
43. classPath.getCanonicalPath() + File.separator)).toString();
44. urls[0] = **new** URL(**null**, repository, streamHandler);
45. loader = **new** URLClassLoader(urls);
46. } **catch** (IOException e) {
47. e.printStackTrace();
48. }
49. **return** loader;
50. }
52. **private** **static** Servlet loadServlet(String servletName) {
53. URLClassLoader loader = createUrlClassLoader();
54. Servlet servlet = **null**;
55. **try** {
56. @SuppressWarnings("unchecked")
57. Class<Servlet> servletClass = (Class<Servlet>) loader
58. .loadClass(servletName);
59. servlet = (Servlet) servletClass.newInstance();
60. } **catch** (ClassNotFoundException e) {
61. e.printStackTrace();
62. } **catch** (InstantiationException e) {
63. e.printStackTrace();
64. } **catch** (IllegalAccessException e) {
65. e.printStackTrace();
66. }
67. **return** servlet;
68. }
70. **private** **static** **void** callService(Servlet servlet, ServletRequest request,
71. ServletResponse response) {
72. **try** {
73. servlet.service(request, response);
74. } **catch** (ServletException e) {
75. e.printStackTrace();
76. } **catch** (IOException e) {
77. e.printStackTrace();
78. }
80. }
82. }

MappingHeler:

**[java]** [view plain](http://blog.csdn.net/bingduanlbd/article/details/38349737) [copy](http://blog.csdn.net/bingduanlbd/article/details/38349737)

1. **package** simpleserver.simplecontainer;
3. **import** java.io.FileInputStream;
4. **import** java.io.IOException;
5. **import** java.util.Properties;
7. **public** **class** MappingHelper {
9. **public** **static** Properties requestMapping;
10. **static** {
11. requestMapping = **new** Properties();
12. **try** {
13. requestMapping.load(**new** FileInputStream(Constant.MAPPING\_FILE));
14. } **catch** (IOException e) {
15. e.printStackTrace();
16. }
17. }
19. **public** **static** String resolve(String requestPath) {
20. **return** requestMapping.getProperty(requestPath);
21. }
23. }

该类负责请求URL到Servlet的映射，读入一个属性文件，如下：

**[java]** [view plain](http://blog.csdn.net/bingduanlbd/article/details/38349737) [copy](http://blog.csdn.net/bingduanlbd/article/details/38349737)

1. /servlet/PrimitiveServlet=simpleserver.simplecontainer.webapp.PrimitiveServlet
2. /servlet/Test=simpleserver.simplecontainer.webapp.TestServlet

另外还有一个帮助类定义常量。

欢迎交流。