### 创建服务器端套接字

*/\*\*  
 \* httpserver程序入口  
 \*  
 \** ***@author*** *nino  
 \** ***@date*** *2019/6/10  
 \*/***public class** BootStrap {  
  
 **public static** Logger *logger* = Logger.*getLogger*(BootStrap.**class**);  
  
 */\*\*  
 \* 主程序  
 \*/* **public static void** main(String[] args) {  
 *start*();  
 }  
  
 */\*\*  
 \* 主程序入口  
 \*/* **public static void** start() {  
 **try** {  
 *logger*.info(**"httpserver start"**);  
  
 *//获取当前时间* **long** start = System.*currentTimeMillis*();  
  
 *//获取系统端口号* **int** port = ServerParser.*getPort*();  
 *logger*.info(**"httpserver-port:"** + port);  
  
 *//创建服务器套接字，并且绑定端口号：8080* ServerSocket serverSocket = **new** ServerSocket(port);  
  
 *//获取结束时间* **long** end = System.*currentTimeMillis*();  
 *logger*.info(**"httpserver started:"** + (end - start) + **"ms"**);  
  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

### 配置端口号到xml文件

创建server.xml文件

*<?***xml version="1.0" encoding="UTF-8"***?>*<**server**>  
 <**service**>  
 <**connector port="8080"**></**connector**>  
 </**service**>  
</**server**>

使用dom4j解析xml（需要两个jar包）

*/\*\*  
 \* 解析server.xml配置文件文件  
 \*  
 \** ***@author*** *nino  
 \** ***@date*** *2019/6/11  
 \*/***public class** ServerParser {  
  
 */\*\*  
 \* 获取服务器端口号  
 \*  
 \** ***@param:*** *[]  
 \** ***@return:*** *int  
 \*/* **public static int** getPort() {  
 *//设置服务器默认端口号：8080* **int** port = 8080;  
 **try** {  
 *//创建解析器* SAXReader saxReader = **new** SAXReader();  
  
 *//通过解析器的read方法将配置文件读取到内存中，生成一个Document[org.dom4j]对象树* Document document = saxReader.read(**"resources/server.xml"**);  
  
 *//获取connector节点元素的路径：server->service->connector  
 //获取connector节点元素的xpath路径：/server/service/connector  
 //获取connector节点元素的xpath路径：/server//connector  
 //获取connector节点元素的xpath路径：//connector* Element connectorElt = (Element) document.selectSingleNode(**"/server/service/connector"**);  
 *//获取port属性的值* port = Integer.*parseInt*(connectorElt.attributeValue(**"port"**));  
  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 }  
 **return** port;  
 }  
}

### 接收客户端消息并打印

*//让服务器一直处理监听状态***while** (**true**) {  
 *//开始监听，此时程序处于等待状态等待接收客户端的消息* clientSocket = serverSocket.accept();  
 *//接收客户端消息* br = **new** BufferedReader(**new** InputStreamReader(clientSocket.getInputStream()));  
 String temp = **null**;  
 **while** ((temp = br.readLine()) != **null**) {  
 System.***out***.println(temp);  
 }  
  
}

注：

解决多次访问的问题：使用死循环while(true)让服务器一直处于监听状态；

记得关闭资源；

### 实现多线程处理客户端请求

*/\*\*  
 \* httpserver程序入口  
 \*  
 \** ***@author*** *nino  
 \** ***@date*** *2019/6/10  
 \*/***public class** BootStrap {  
  
 **public static** Logger *logger* = Logger.*getLogger*(BootStrap.**class**);  
  
 */\*\*  
 \* 主程序  
 \*/* **public static void** main(String[] args) {  
 *start*();  
 }  
  
 */\*\*  
 \* 主程序入口  
 \*/* **public static void** start() {  
 ServerSocket serverSocket = **null**;  
 Socket clientSocket = **null**;  
 BufferedReader br = **null**;  
 **try** {  
 *logger*.info(**"httpserver start"**);  
  
 *//获取当前时间* **long** start = System.*currentTimeMillis*();  
  
 *//获取系统端口号* **int** port = ServerParser.*getPort*();  
 *logger*.info(**"httpserver-port:"** + port);  
  
 *//创建服务器套接字，并且绑定端口号：8080* serverSocket = **new** ServerSocket(port);  
  
 *//获取结束时间* **long** end = System.*currentTimeMillis*();  
 *logger*.info(**"httpserver started:"** + (end - start) + **"ms"**);  
  
 *//让服务器一直处理监听状态* **while** (**true**) {  
 *//开始监听，此时程序处于等待状态等待接收客户端的消息* clientSocket = serverSocket.accept();  
  
 **new** Thread(**new** HandlerRequest(clientSocket)).start();  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **finally** {  
 *//关闭资源* **if** (serverSocket != **null**) {  
 **try** {  
 serverSocket.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

*/\*\*  
 \* 处理客户端请求  
 \*  
 \** ***@author*** *nino  
 \** ***@date*** *2019/6/12  
 \*/***public class** HandlerRequest **implements** Runnable {  
 **public static** org.apache.log4j.Logger *logger* = Logger.*getLogger*(HandlerRequest.**class**);  
  
 **public** Socket **clientSocket**;  
  
 **public** HandlerRequest(Socket clientSocket) {  
 **this**.**clientSocket** = clientSocket;  
 }  
  
 @Override  
 **public void** run() {  
 *//处理客户端请求* BufferedReader br = **null**;  
  
 *logger*.info(**"httpserver thread:"** + Thread.*currentThread*().getName());  
 **try** {  
 *//接收客户端消息* br = **new** BufferedReader(**new** InputStreamReader(**clientSocket**.getInputStream()));  
 *//打印客户端消息* String temp = **null**;  
 **while** ((temp = br.readLine()) != **null**) {  
 System.***out***.println(temp);  
 }  
 } **catch** (Exception e) {  
 e.printStackTrace();  
 } **finally** {  
 **if** (br != **null**) {  
 **try** {  
 br.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **if** (**clientSocket** != **null**) {  
 **try** {  
 **clientSocket**.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
}

### 获取URI