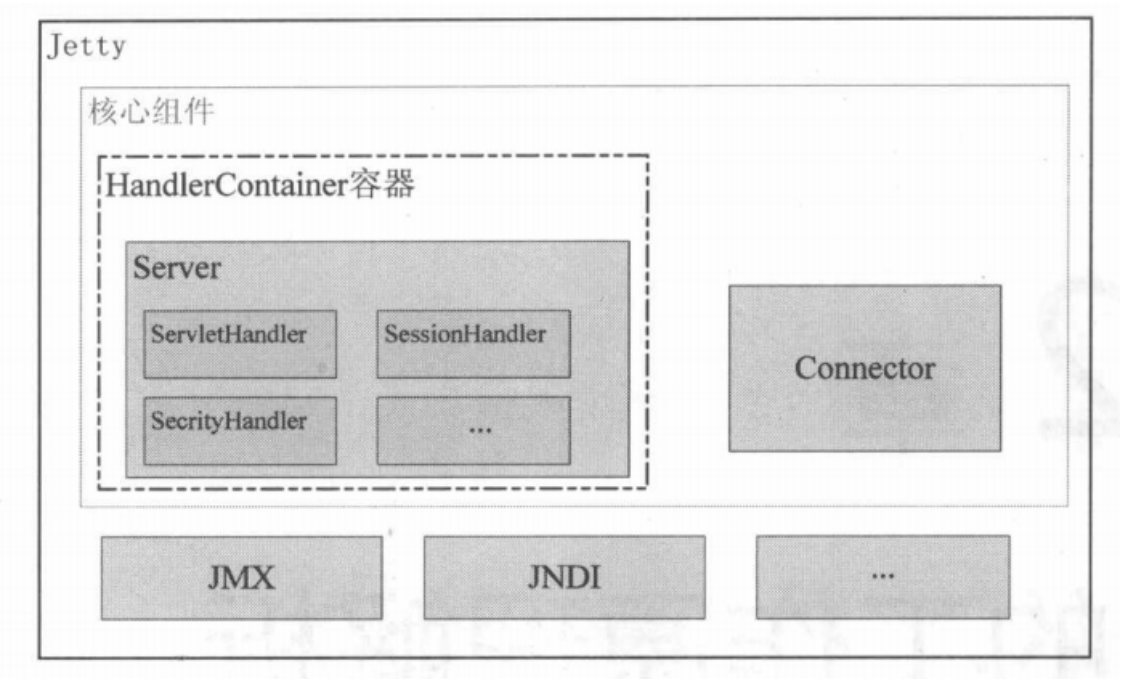
Jetty

基本架构

总览



核心组件：Server和Connector

Server（核心）：基于Handler容器工作

Connector：负责接受客户端的连接请求，并将请求分配给一个处理队列去执行

可有可无的组件：JMX.....

整个Jetty的核心围绕着Server类来构建，Server类继承了Handler，关联了Connector和Container，Container是管理Mbean的容器。Jetty的Server的扩展主要是实现一个个Handler并将Handler加到Server中，Server中提供了调用这些Handler的访问规则。

两类Handler

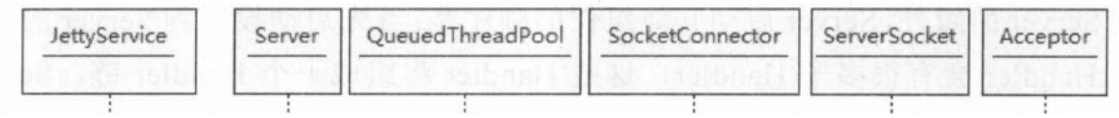
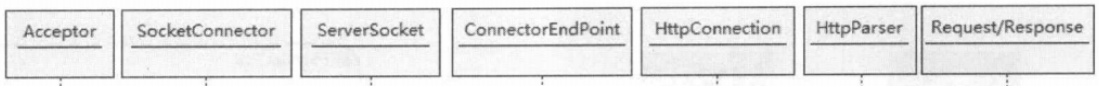
- HandlerWrapper：将一个Handler委托给另外一个类去执行，配合ScopeHandler类可以拦截Handler的执行
    - ScopedHandler是HandlerWrapper 一个抽象实现类，用来提供链式调用时作用域的支持
- HandlerCollection：将多个Handler组装在一起，构成一个Handler链

启动过程

- Jetty的入口是Server类，Server类启动完成了，就代表Jetty可以提供服务了
- Server的start方法会调用所有已经注册到Server的组件
- 其他组件的启动顺序：首先启动Server的Handler，然后依次启动Handler链上的所有Handler，接着会启动注册在Server上JMX的Mbean，最后启动Connector，打开端口接受客户端请求

接受请求

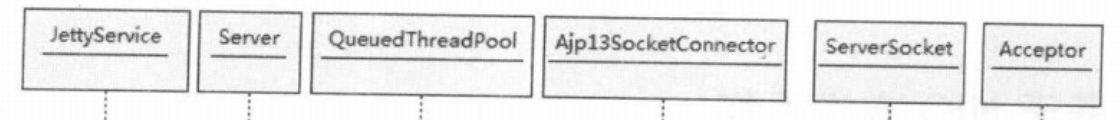
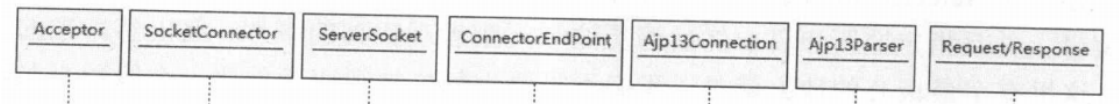
基于HTTP协议

> 创建连接
>
> 
>
>
> 处理连接
>
> 
>
>
> HttpConnection类：解析和封装HTTP协议
>
> Acceptor线程将会为这个请求创建ConnectorEndPoint：以BIO方式处理连接请求

- Jetty创建接受连接环境的三个步骤：
    1. 创建一个队列线程池，用于处理每个建立连接的任务
    2. 创建ServerSocket，用于准备接受客户端的Socket请求
    3. 创建一个或多个监听线程，用来监听访问端口是否有连接进来

AJP协议

- SocketConnector类替换成了Ajp13SocketConnector类

> 创建连接
>
> 
>
>
> 处理连接
>
> 
>

处理请求

- 工作方式：调用第一个Handler的handle (String target, Request baseRequest, HttpServletRequest request, HttpServletResponse response)方法
- 要能接受一个Web请求访问，首先要创建一个ContextHandler

特点总结

- 架构简单，面向Handler的架构
- 从设计模板的角度看
    - 责任链模式：接口类HandlerCollection可以帮助开发者构建一个链，而另一个接口类ScopeHandler可以帮助开发者控制这个链的访问顺序
    - 观察者模式：只要继承了LifeCycle接口，对象就可以交给Jetty来统一管理了
- Jetty适合同时处理大量连接而且可以长时间保持这些连接，例如一些Web聊天应用
- Jetty默认使用NIO技术，在处理I/O请求上更占优势

> BIO(同步阻塞I/O模式)：数据的读取写入必须阻塞在一个线程内等待其完成
>
> NIO(同步非阻塞)：如果还拿烧开水来说，NIO的做法是叫一个线程不断的轮询每个水壶的状态，看看是否有水壶的状态发生了改变，从而进行下一步的操作

常见的各种Handler

- ResourceHandler：处理当前工作路径下的静态资源
- ContextHandler：一种ScopedHandler，只用来响应配匹配指定URI前缀的请求
- ServletContextHandler：一种特殊的ContextHandler，它可以支持标准的sessions 和Servlets
- WebAppContext：ServletContextHandler的扩展，使用标准的web应用组件和web.xml，通过web.xml和注解配置servlet，filter和其它特性

参考代码

```
public class RecSysServer {
    //主函数，创建推荐服务器并运行
    public static void main(String[] args) throws Exception {
        new RecSysServer().run();
    }
    //推荐服务器的默认服务端口6010
    private static final int DEFAULT_PORT = 6010;
    //运行推荐服务器的函数
    public void run() throws Exception{
        int port = DEFAULT_PORT;
        //绑定IP地址和端口，0.0.0.0代表本地运行
        InetSocketAddress inetAddress = new InetSocketAddress("0.0.0.0", port);
        //创建Jetty服务器
        Server server = new Server(inetAddress);
        //创建Jetty服务器的环境handler
        ServletContextHandler context = new ServletContextHandler();
        //获取路径
        context.setContextPath("/");
        context.setWelcomeFiles(new String[] { "index.html" });
        //添加API，getMovie，获取电影相关数据
        context.addServlet(new ServletHolder(new MovieService()), "/getmovie");
        //添加api，getuser，获取用户相关数据
        context.addServlet(new ServletHolder(new UserService()), "/getuser");
        //添加api，getsimilarmovie，获取相似电影推荐
        context.addServlet(new ServletHolder(new SimilarMovieService()),
"/getsimilarmovie");
        //添加api，getrecommendation，获取各类电影推荐
        context.addServlet(new ServletHolder(new RecommendationService()),
"/getrecommendation");
        //设置Jetty的环境handler
        server.setHandler(context);
        //启动Jetty服务器
        server.start();
        server.join();
    }
}
```

主要参考资料

《深入分析Java Web技术内容》

https://www.jianshu.com/p/088277390eb4