Tomcat的相关理论

1.tomcat的核心组件

- engine　tomcat请求过来会先经过engine这个容器，也就是接收所有的请求，根据请求找到一个对应的host阀门
- host　　处理同一个同一域名下的所有应用，tomcat可以支持不同域名的请求，在server.xml中进行配置，寻找对应的应用阀门
- context　List<wrapper> wrapper　同一个应用下的servlet在线程安全的情况下，不安全就是wrapper，找到对应的wrapper
- wrapper　同一个wrapper下找到对应的servlet，servlet调用service-->调用restful风格的请求（源码支持）

2.tomcat的长连接和浏览器

2.1.http1.1请求：

请求行：请求方式 空格　请求路径　空格 协议 回车换行

请求头：connection ： keep-alive 等

请求体：username: guihaole,password:hahaha

2.2.浏览器和服务器长连接

说明：

1.客户端向服务端建立socket连接（bio），客户端发送的数据都会存放在服务端的recvbuf，注意这个缓冲区是操作系统的范畴，假如服务端没有读数据操作，那客户端一般发送连接到达一定程度，recvbuf会存满。如果服务端读取recvbuf中的数据，那么client会接着发送数据（长连接过程）；

2.如果是浏览器和tomcat，浏览器发送数据，在头中带一个connection:keepalive,告诉tomcat,这是一个长连接，不要关闭连接，这样就可以持续发送数据，如果connection:close，这就是一个短链接

3.tomcat请求流程解析

浏览器向tomcat发送请求建立socket连接，过程操作系统的数据接收，如何在socket中获取数据,tomcat这边会有一个keepalive属性，用于标记是否是长连接，默认为true

```java
endpoint为请求入口，就是一个对象，tomcat启动的时候，对象里面的线程就会跑起来，等着接收socket
keepalive=true
while(keepalive){
  setRequestLineReadTimeOut();//超时处理
  getInputBuffer.parseRequestLine(keepalive);//~~请求行
  getInputBuffer.parseHeaders();//从socket中解析请求头
  prepareRequest();//获取请求头中的connection
  keepalive=false/true;//一个请求头close为false;socket最大请求数达到为false;当线程池的忙碌线程>线程工作阈值75默认100也会变为false
  //封装成request对象
  adapter.service(request,response);
}
```

4.tomcat请求大体架构流程

1. 接收到socket
2. 将socket交给线程池
3. 一个线程处理一个socket连接
4. 开始从socket中获取数据
5. 解析请求行----
6. 解析请求头
7. 根据请求头解析Connection对应的值是keepalive还是close
8. 请求行和请求头解析后会设置到Request对象中
9. 将Request对象交给容器进行处理
10. 容器最终会交给对应的servlet进行处理
11. servlet中可以获取请求的各种信息，包括获取请求体
12. servlet中也可以使用response对象想客户端返回响应
13. servlet中的代码都执行完成后，相当于容器中已经处理完了请求，相当于请求的核心逻辑已经执行完了
14. 处理InputBuffer中的pos和lastValid，以便能够处理下一个请求

5.tomcat中的门面模式调用过程

我们程序员使用的request----requestFacade

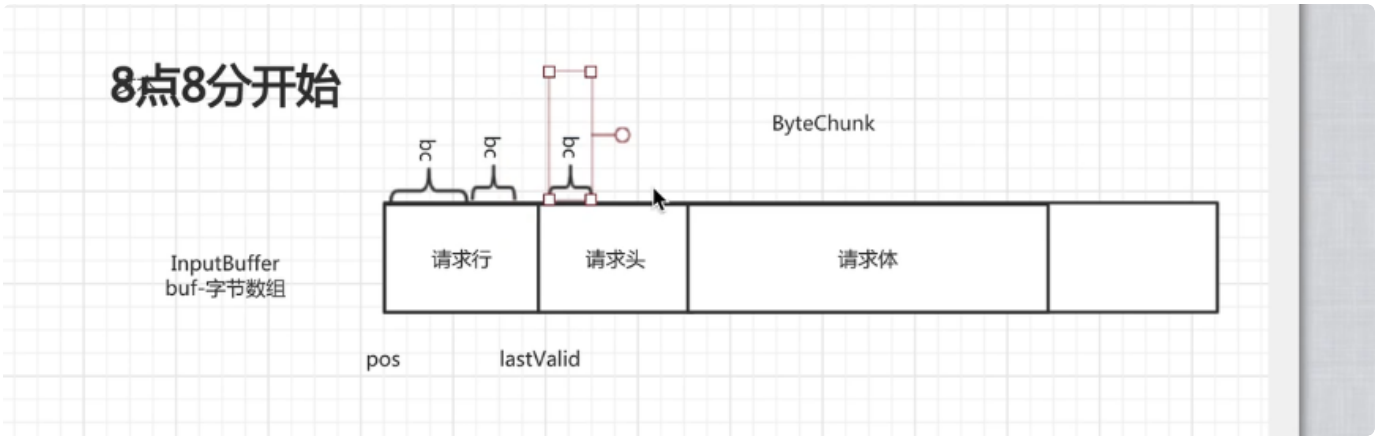
门面的request-----connector.request

底层的request-------coyote.Request

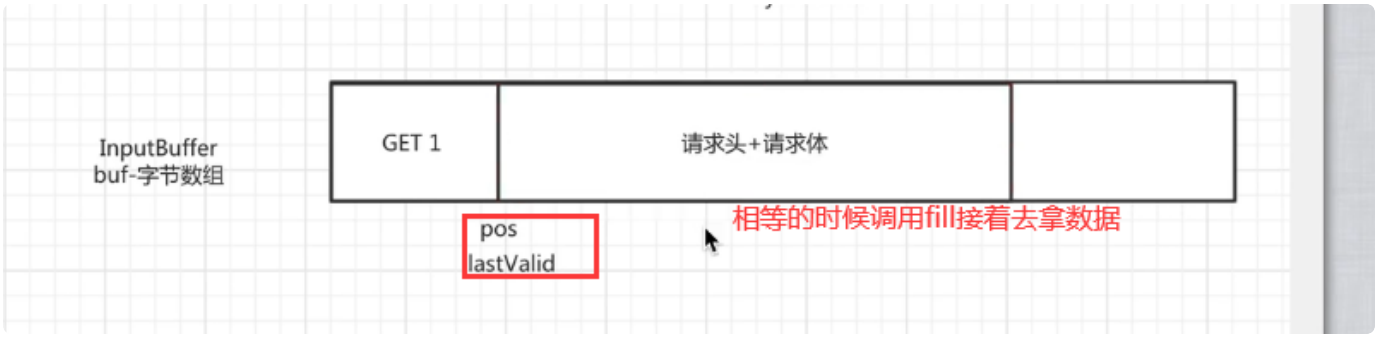
```
Request
    method
    url
    header

getMethod
RequestFacade---->connector.Request--->coyote.Request
```

6.tomcat如何获取请求行中的数据



说明：tomcat在解析自己缓冲区中的数据的时候会使用字节块byteChunk,我们的请求都是有固定格式的，空格用bc标记一个请求块，设置到底层的request中，最终当程序员调用的时候使用门面模式进行，转换成为字符串



说明：我们在解析请求的时候会使用两个下标,pos代表遍历到的字节位置，lastValid代表缓冲区中的字节长度，如果pos==lastValid的话，整明缓冲区中的数据已经遍历结束了，我们就调用fill方法去在从操作系统的临时缓冲区去获取数据.

7.tomcat如何区分第一个请求还是第二个请求

- context-length:请求头中代表传递请求体的长度，如果你的请求体为12345,而你的请求长度只有3，那么tomcat认为你的请求体是3，第二个请求解析也会出错，所以tomcat也没有那么智能
- transfer-encoding:分块传输,2/r/n;aa/r/n;0/r/n;/r/n这个为结束标志，tomcat也会一块一块的读取

8.tomcat是如何通过dispatchServlet创建spring容器的

我们在使用springMvc的时候，我们需要在web.xml中配置一个dispatchServlet,tomcat最关心的是什么，他最关心的是，如果向tomcat端口发送一个http请求，建立长连接，操作系统会通过字节流接收到数据，放在服务端的缓冲池中，这个缓冲池是操作系统层面的，而tomcat通过JENpoint,接收到socket之后，将socket丢给线程池，一个线程处理一个socket,tomcat会先创建一个自己的缓冲区，从操作系

统层面的缓冲池中读数据，然后根据http格式进行解析，用字节块进行一个一个的标记，通过门面模式，让程序员在调用的时候将字节块转换成字符串，当然如果是长连接，tomcat通过分块传输请求体，或者请求体长度去解析。将所有的字节块赋值给request,将request交给容器去处理，容器会通过四大组件容器链，找到一个对应的servlet，什么时候创建spring容器的，在servlet初始化的时候Init,他的父类有一个属性叫application，在init方法中就会初始化null的sping容器，然后通过参数配置 得到一个有bean的spring容器即可。