# SimpleTomcat各个主题汇总

我们把SimpleTomcat待讨论的各个主题汇总在这里。

# 主题1-如何阅读各个版本的tomcat

参考： 主题1-如何阅读各个版本的tomcat

# 主题2-pipeline主题

# 备选主题

## Tomcat整体架构是什么样的？

我们简单在《Tomcat整体架构.uxf》中总结了tomcat的整体架构

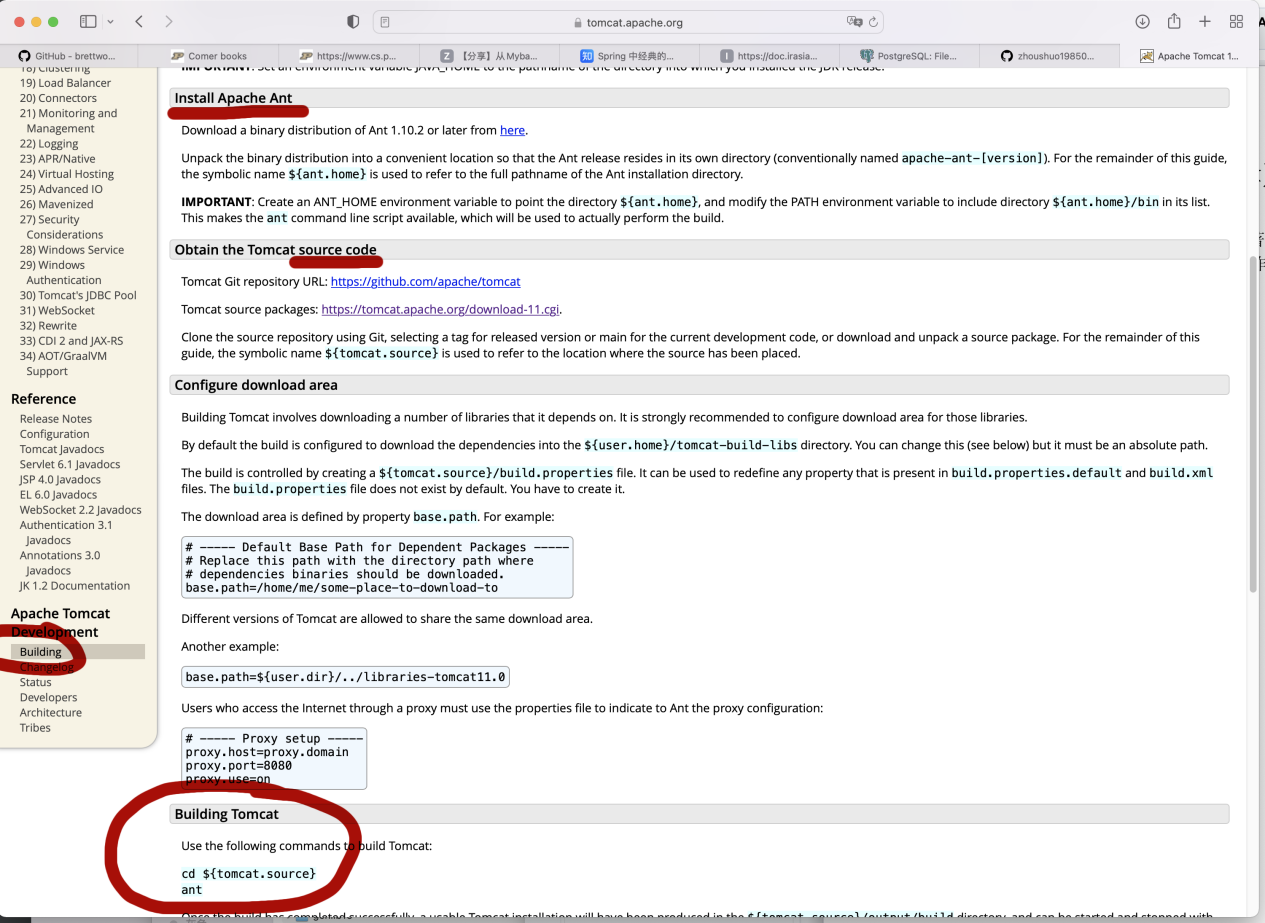
## 2.一个http请求过来，tomcat是如何处理这个请求的？

## 3.如何编译tomcat

tomcat官网上有对tomcat的编译方法进行说明。大致步骤就是：

1. 下载合适的JDK版本；
2. 安装ant
3. 下载tomcat源码
4. 利用ant编译tomcat源码

后续参考这个文档实践一下。



## 2.如何给tomcat贡献源码

## 3.tomcat和JDK/Servlet版本有什么联系

|  |
| --- |
| tomcat各个版本  Tomcat4 JDK1.4 servlet 2.3  Tomcat5  JDK1.5起 servlet 2.4  Tomcat6 JDK1.5起 servlet 2.5  Tomcat7 JDK1.6 servet3.0  Tomcat8 JDK 1.7 servlet3.1  Tomcat9 JDK 1.8 servlet4.0 |

## 4.Springboot是如何集成tomcat的？

## 5.How to warmup？

我们碰到一个很大的问题，就是一段时间没有学tomcat源码，要重新拾起来就比较麻烦，如何能够快速入门，需要专题讨论一下。

是UML图？写各个测试案例？写文档？

目标是尽快热身熟悉代码结构，上手开发新的功能。

## 6.tomcat是如何处理keepalive的？

我们之前在一篇网文中，学习过如何通过调整tomcat运行时代码参数，改变tomcat keepalive行为，最终实现优雅f5切换的目标。

在这里主题中，我们参考这篇文章，系统学习一下tomat具体是如何实现keepalive的。

keepalive涉及tomcat遵循http协议处理http servlet request，这就涉及到tomcat专业的connector实现类：coyote。

我们随便浏览一下coyote的代码实现，比如我们找了tomcat9的源码：：

|  |
| --- |
| // 处理Http1.1协议的coyote Processor  Http11Processor.java  // 是否需要keepalive的参数  private volatile boolean keepAlive = true; |

有很多涉及keepalive的片段：

|  |
| --- |
| // 代码片段1 根据实际情况数结合阈值，判断是否需要keepalive  int maxKeepAliveRequests = protocol.getMaxKeepAliveRequests(); if (maxKeepAliveRequests == 1) {  keepAlive = false; } else if (maxKeepAliveRequests > 0 &&  socketWrapper.decrementKeepAlive() <= 0) {  keepAlive = false; } |

|  |
| --- |
| // 代码片段2 根据HTTP header判断是否需要keepalive  MessageBytes connectionValueMB = headers.getValue(Constants.CONNECTION); if (connectionValueMB != null && !connectionValueMB.isNull()) {  Set<String> tokens = new HashSet<>();  TokenList.parseTokenList(headers.values(Constants.CONNECTION), tokens);  if (tokens.contains(Constants.CLOSE)) {  keepAlive = false;  } else if (tokens.contains(Constants.KEEP\_ALIVE\_HEADER\_VALUE\_TOKEN)) {  keepAlive = true;  } } |

## 7.tomcat是如何实现NIO的

我们通过学习netty，接触到了NIO的相关功能，NIO的好处是很多的，比如一个节点可以接收很多很多的并发连接，那么tomcat是从哪个版本开始引入NIO的，NIO的相关实现逻辑是怎么样的，和Netty相比，有何异同。