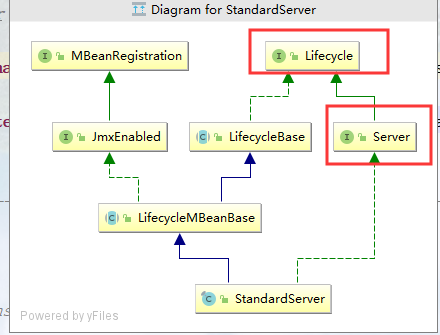
**Tomcat**

# StandardServer

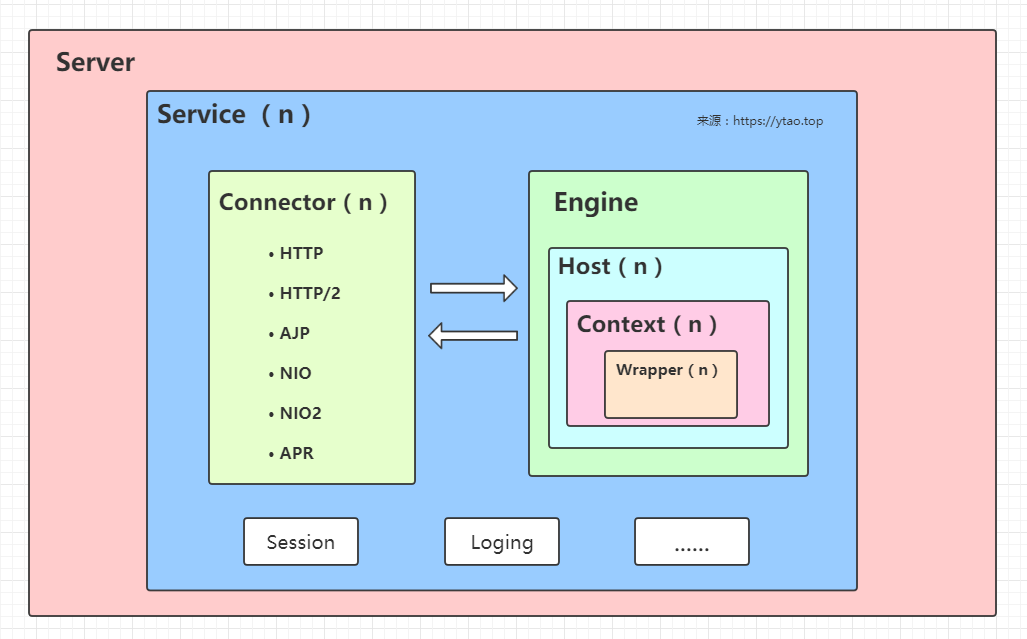
## 类图



## 初始化

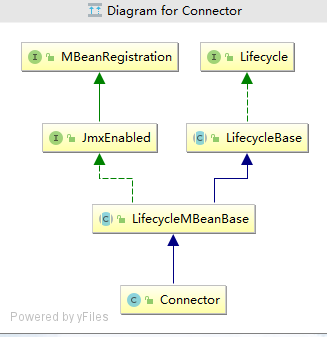
由Catalina初始化的

# 架构图



# Connector

## 类图



## 初始化

是在StandarServer中初始化的

# Bootstrap

## 作用

启动tomcat

## Main方法

|  |
| --- |
| ***/\*\*  \* Main method and entry point when starting Tomcat via the provided  \* scripts.  \*  \* @param args Command line arguments to be processed  \*/* public static void main(String args[]) {   if (*daemon* == null) {  *// Don't set daemon until init() has completed* Bootstrap bootstrap = new Bootstrap();  try {**  **// 初始化  bootstrap.init();  } catch (Throwable t) {  *handleThrowable*(t);  t.printStackTrace();  return;  }  *daemon* = bootstrap;  } else {  *// When running as a service the call to stop will be on a new  // thread so make sure the correct class loader is used to prevent  // a range of class not found exceptions.* Thread.*currentThread*().setContextClassLoader(*daemon*.catalinaLoader);  }   try {  String command = "start";  if (args.length > 0) {  command = args[args.length - 1];  }   if (command.equals("startd")) {  args[args.length - 1] = "start";  *daemon*.load(args);  *daemon*.start();  } else if (command.equals("stopd")) {  args[args.length - 1] = "stop";  *daemon*.stop();  } else if (command.equals("start")) {  *daemon*.setAwait(true);  *daemon*.load(args);  *daemon*.start();  if (null == *daemon*.getServer()) {  System.*exit*(1);  }  } else if (command.equals("stop")) {  *daemon*.stopServer(args);  } else if (command.equals("configtest")) {  *daemon*.load(args);  if (null == *daemon*.getServer()) {  System.*exit*(1);  }  System.*exit*(0);  } else {  *log*.warn("Bootstrap: command \"" + command + "\" does not exist.");  }  } catch (Throwable t) {  *// Unwrap the Exception for clearer error reporting* if (t instanceof InvocationTargetException &&  t.getCause() != null) {  t = t.getCause();  }  *handleThrowable*(t);  t.printStackTrace();  System.*exit*(1);  }  }** |

# Catalina

## 作用

启动/终止shell程序的

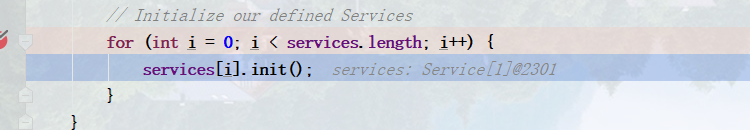
## 启动新的server实例

|  |
| --- |
| ***/\*\*  \* Start a new server instance.  \*/* public void load() {   if (loaded) {  return;  }  loaded = true;   long t1 = System.*nanoTime*();   initDirs();   *// Before digester - it may be needed* initNaming();   *// Create and execute our Digester* Digester digester = createStartDigester();   InputSource inputSource = null;  InputStream inputStream = null;  File file = null;  try {  try {  file = configFile();  inputStream = new FileInputStream(file);  inputSource = new InputSource(file.toURI().toURL().toString());  } catch (Exception e) {  if (*log*.isDebugEnabled()) {  *log*.debug(*sm*.getString("catalina.configFail", file), e);  }  }  if (inputStream == null) {  try {**  **// 配置文件为 server.xml  inputStream = getClass().getClassLoader()  .getResourceAsStream(getConfigFile());  inputSource = new InputSource  (getClass().getClassLoader()  .getResource(getConfigFile()).toString());  } catch (Exception e) {  if (*log*.isDebugEnabled()) {  *log*.debug(*sm*.getString("catalina.configFail",  getConfigFile()), e);  }  }  }   *// This should be included in catalina.jar  // Alternative: don't bother with xml, just create it manually.* if (inputStream == null) {  try {  inputStream = getClass().getClassLoader()  .getResourceAsStream("server-embed.xml");  inputSource = new InputSource  (getClass().getClassLoader()  .getResource("server-embed.xml").toString());  } catch (Exception e) {  if (*log*.isDebugEnabled()) {  *log*.debug(*sm*.getString("catalina.configFail",  "server-embed.xml"), e);  }  }  }    if (inputStream == null || inputSource == null) {  if (file == null) {  *log*.warn(*sm*.getString("catalina.configFail",  getConfigFile() + "] or [server-embed.xml]"));  } else {  *log*.warn(*sm*.getString("catalina.configFail",  file.getAbsolutePath()));  if (file.exists() && !file.canRead()) {  *log*.warn("Permissions incorrect, read permission is not allowed on the file.");  }  }  return;  }   try {  inputSource.setByteStream(inputStream);  digester.push(this);**  **// 解析server.xml  digester.parse(inputSource);  } catch (SAXParseException spe) {  *log*.warn("Catalina.start using " + getConfigFile() + ": " +  spe.getMessage());  return;  } catch (Exception e) {  *log*.warn("Catalina.start using " + getConfigFile() + ": " , e);  return;  }  } finally {  if (inputStream != null) {  try {  inputStream.close();  } catch (IOException e) {  *// Ignore* }  }  }   getServer().setCatalina(this);  getServer().setCatalinaHome(Bootstrap.*getCatalinaHomeFile*());  getServer().setCatalinaBase(Bootstrap.*getCatalinaBaseFile*());   *// Stream redirection* initStreams();   *// Start the new server* try {**  **// 初始化server  getServer().init();  } catch (LifecycleException e) {  if (Boolean.*getBoolean*("org.apache.catalina.startup.EXIT\_ON\_INIT\_FAILURE")) {  throw new java.lang.Error(e);  } else {  *log*.error("Catalina.start", e);  }  }   long t2 = System.*nanoTime*();  if(*log*.isInfoEnabled()) {  *log*.info("Initialization processed in " + ((t2 - t1) / 1000000) + " ms");  } }** |

# StandardService

## 初始化

是由StandardServer初始化的



# StandardEngine

# CoyoteAdapter

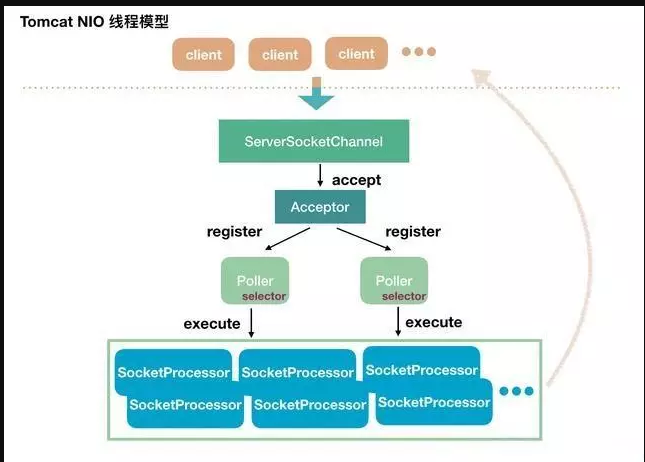
# ProtocolHandler

## Http11NioProtocol

### 是么时候创建的

# NioEndpoint

## 线程模型



## 作用

非阻塞 IO 来进行处理 HTTP/1.1 协议的请求。

## 启动

|  |
| --- |
| ***/\*\*  \* Start the NIO endpoint, creating acceptor, poller threads.  \*/* @Override public void startInternal() throws Exception {   if (!running) {  running = true;  paused = false;   processorCache = new SynchronizedStack<>(SynchronizedStack.*DEFAULT\_SIZE*,  socketProperties.getProcessorCache());  eventCache = new SynchronizedStack<>(SynchronizedStack.*DEFAULT\_SIZE*,  socketProperties.getEventCache());  nioChannels = new SynchronizedStack<>(SynchronizedStack.*DEFAULT\_SIZE*,  socketProperties.getBufferPool());   *// Create worker collection* if ( getExecutor() == null ) {  createExecutor();  }   initializeConnectionLatch();   *// Start poller threads* pollers = new Poller[getPollerThreadCount()];  for (int i=0; i<pollers.length; i++) {  pollers[i] = new Poller();  Thread pollerThread = new Thread(pollers[i], getName() + "-ClientPoller-"+i);  pollerThread.setPriority(threadPriority);  pollerThread.setDaemon(true);  pollerThread.start();  }  // 启动Accopetor线程，监听客户端的请求  startAcceptorThreads();  } }** |