BIO

- 阻塞IO,在read()和write()操作时候，当前线程会阻塞。
- 阻塞IO+多线程 来支持多个客户端连接服务端。（线程创建开销明显，线程数不受控制）
- 阻塞IO+线程池来进行优化（tomcat容器在使用BIO模式即为该模式）

NIO

- Non-block IO
- 在read()和write()操作时候不会阻塞。而是通过在seletor上注册监听事件，阻塞在seletor上。
- 用主线程（一个线程或者是 CPU 个数的线程）保持住所有的连接，管理和读取客户端连接的数据，将读取的数据交给后面的线程池处理，线程池处理完业务逻辑后，将结果交给主线程发送响应给客户端，少量的线程就可以处理大量连接的请求。

servlet3.0之前

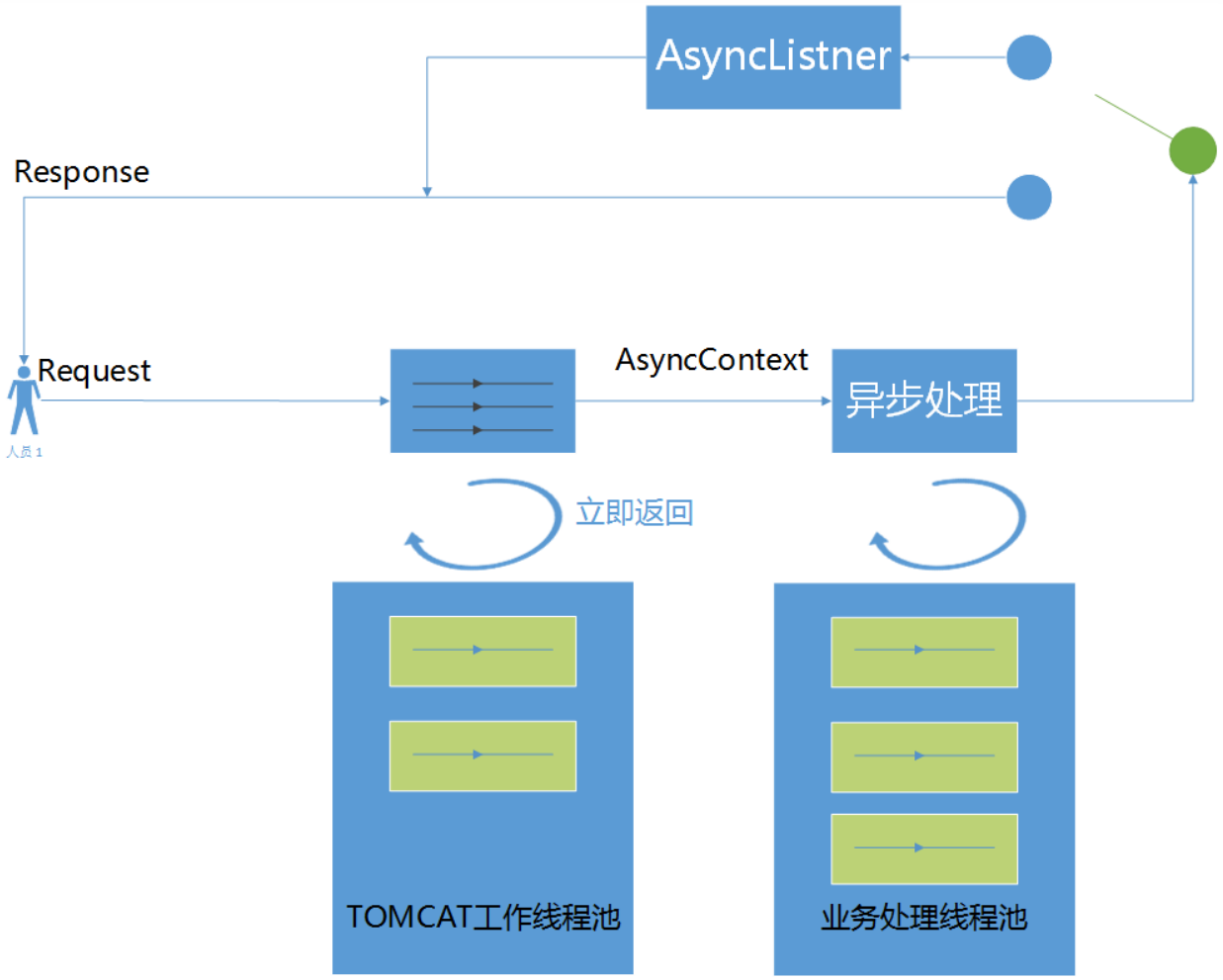
- 单例多线程
- 模式通常是 获取request数据 —> 业务逻辑操作（业务逻辑判断，数据库等操作）—> 写入response数据
- 假设servletA 访问耗时20s, servletB 访问耗时200ms,在1s内有大量请求，系统很快就会被servletA hold住，servletB并不能提供服务。

servlet3.0

- 新增了AsyncContext，提供异步线程处理。
- 将耗时的操作交给worker线程去做，request线程及时收回request线程池，继续能去处理其他请求。
- 接收到request请求之后，由tomcat工作线程从HttpServletRequest中获得一个异步上下文AsyncContext对象，然后由tomcat工作线程把AsyncContext对象传递给业务处理线程，同时tomcat工作线程归还到工作线程池，这一步就是异步开始。在业务处理线程中完成业务逻辑的处理，生成response返回给客户端。

servlet3.0 + BIO

在Servlet3.0中虽然处理请求可以实现异步，但是InputStream和OutputStream的IO操作还是阻塞的，当数据量大的request body 或者 response body的时候，就会导致不必要的等待。



servlet3.1 + NIO

从Servlet3.1以后增加了非阻塞IO，需要tomcat8.x支持