~~Into the wild with~~ Servlet Async IO

Greg Wilkins <gregw@webtide.com>
Founder: Jetty, cometd, Mortbay, Webtide
Standards: Servlet, Websocket, HTTP/2
http://www.webtide.com

Servlet 3.1 Async IO

Why async Servlets?

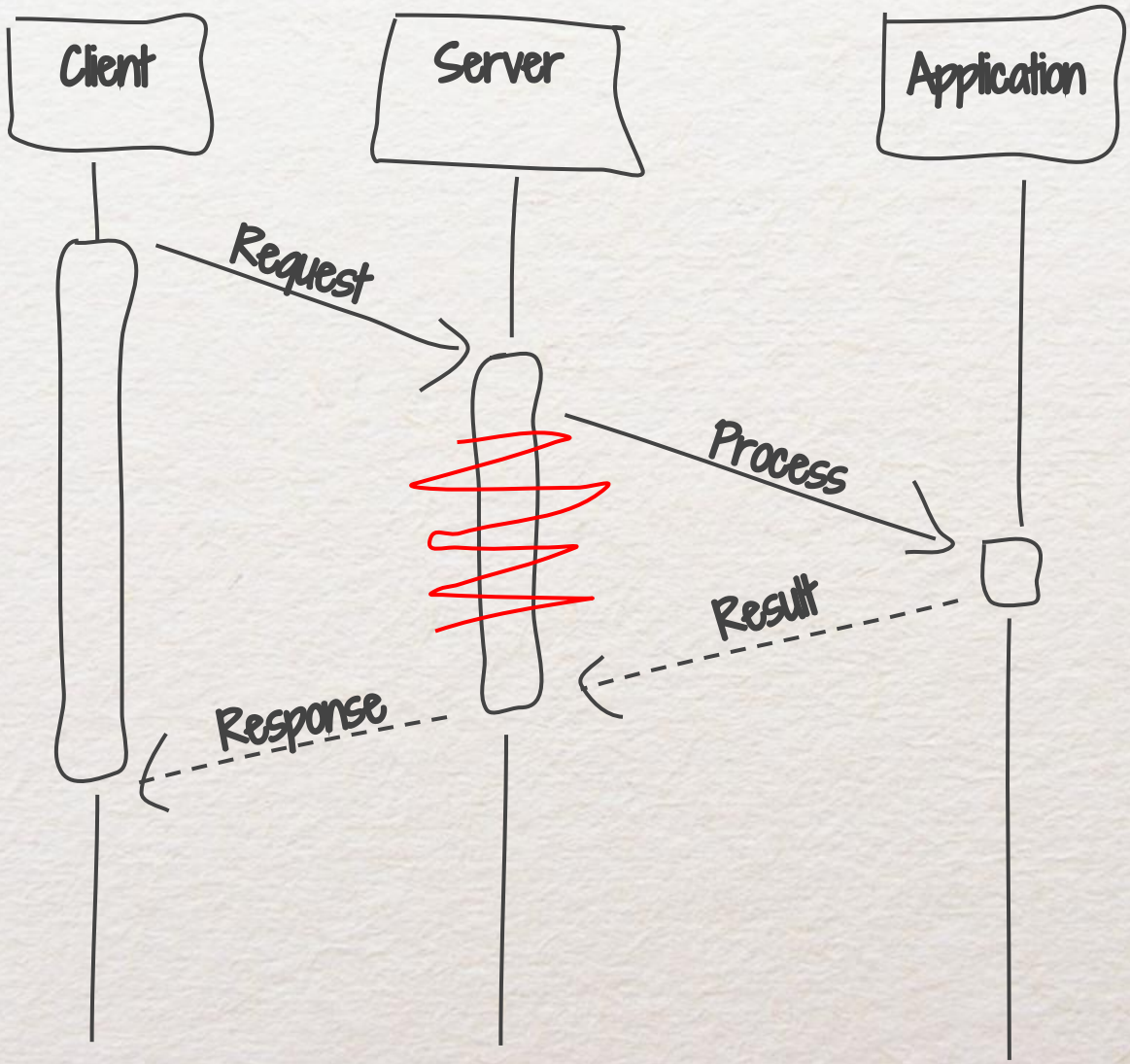
Wasn't Servlet 3.0 Async?

Oh! async IO in Servlet 3.1!

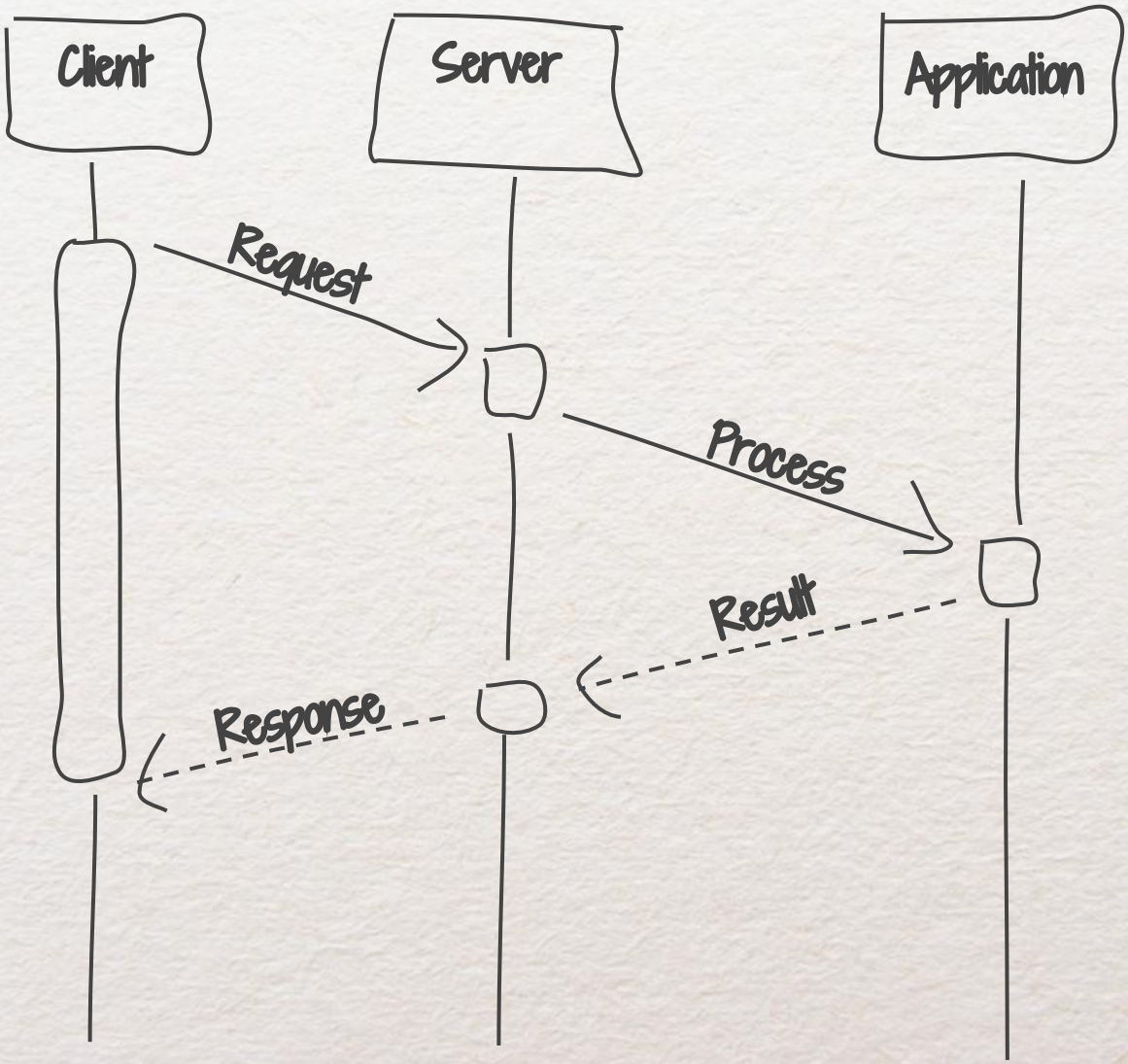
How hard can that be? I know NIO!

Geeeeeze that's hard! Can anybody really use it?

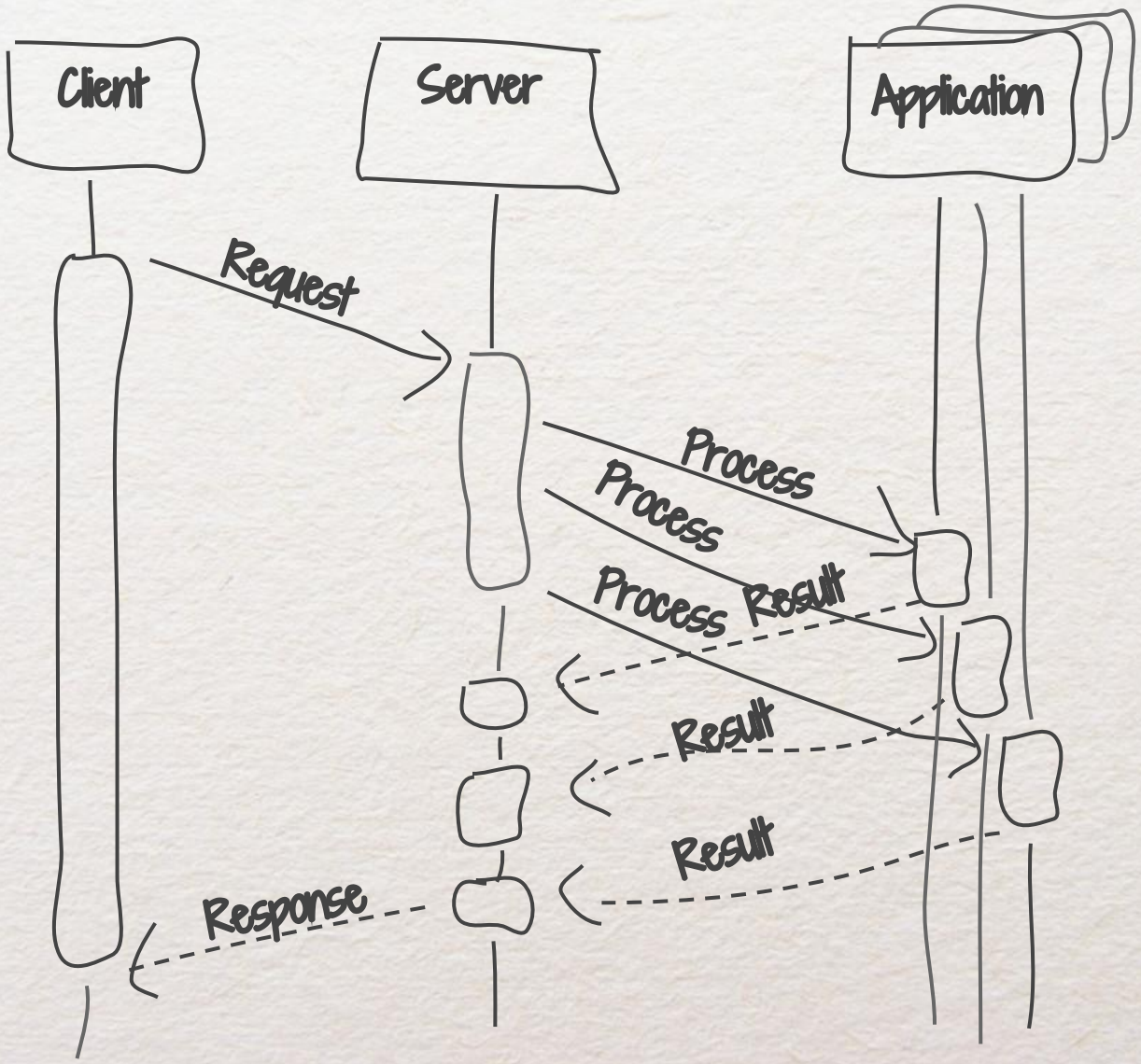
Synchronous Processing



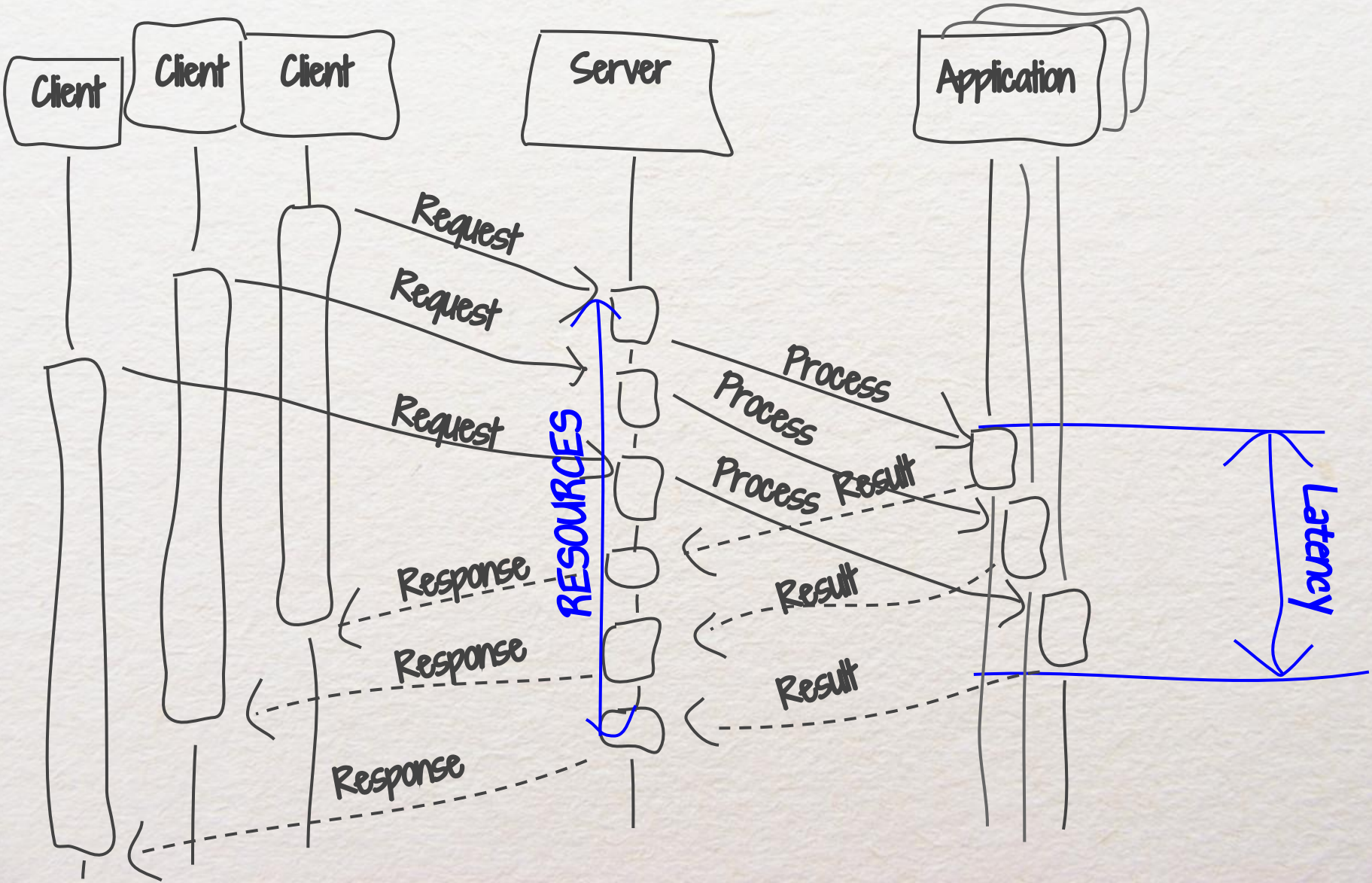
Asynchronous Processing



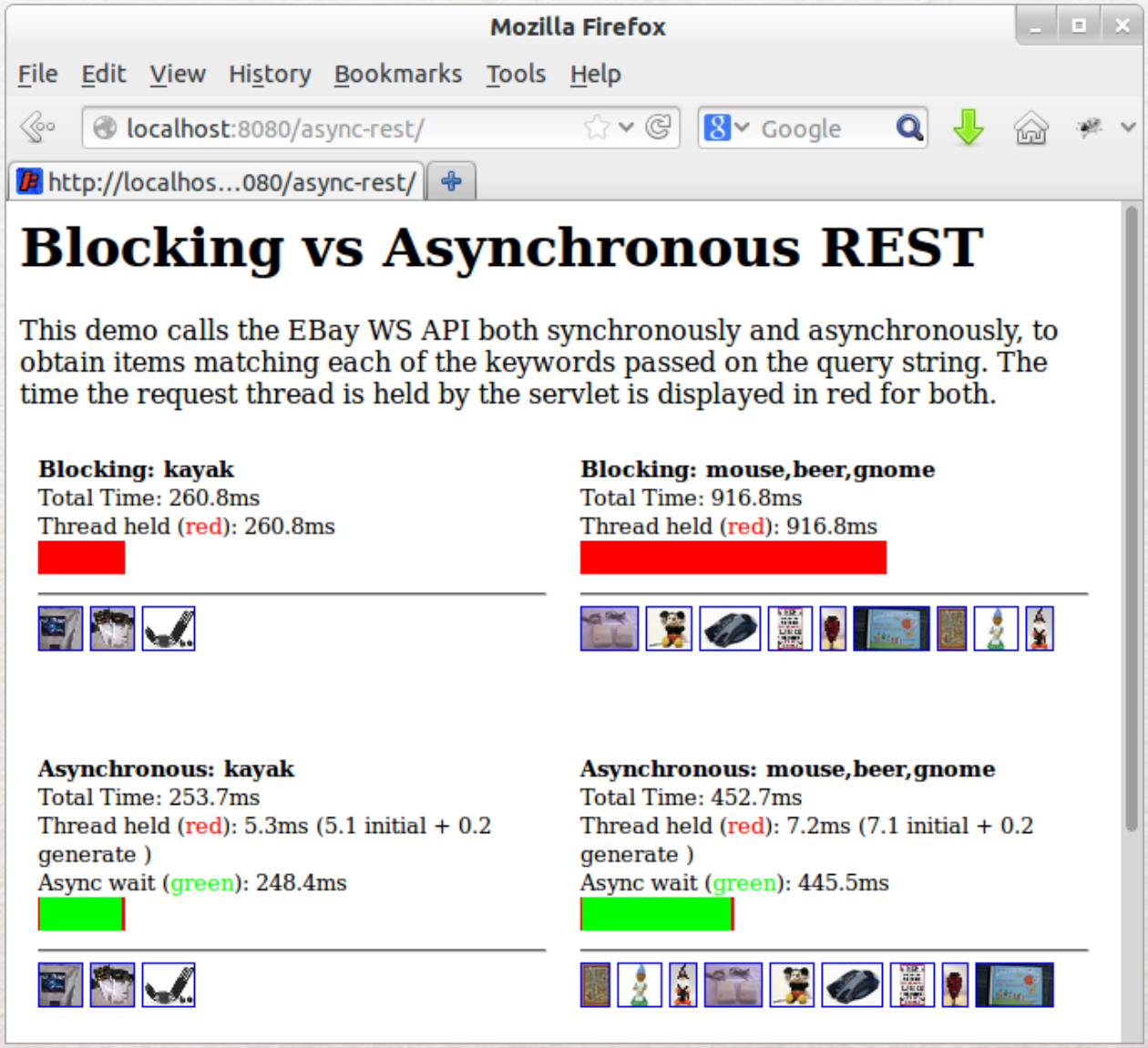
Asynchronous Processing



Asynchronous Processing



Async ReST Demo



An Async Servlet 3.0 Style

```
void doGet(ServletRequest req, final ServletResponse resp)
{
    final AsyncContext ctx=req.startAsync();

    startAsyncProcessing(req, new Callback()
    {
        public void onSuccess(Object result)
        {
            resp.setStatus(200)
            resp.getWriter().print(result.toString());
            ctx.complete();
        }
    });
}
```

It takes two to tango asynchronously!

What if this blocks?

What can an async Servlet do?

- ~~Access database~~ ✗ Few Async clients!
- NIO Calls ✔
- REST / RPC calls ✔ Async clients!
- Talk to equests! ✔
- ~~Read S~~ ~~Body~~
- ~~Write S~~ ~~Body~~

Need Async IO API in Servlet 3.1

No API in servlet 3.0!

NIO? Candidate for Servlet 3.1

```
public interface AsynchronousByteChannel
    extends AsynchronousChannel
{
   Future<Integer> write(ByteBuffer src);

   <A> void write(ByteBuffer src,
                  A attachment,
                  CompletionHandler
                  <Integer,? super A>
      handler);
   ...
}
```

NIO? - No Future for Futures!

```
{
   Future<Integer> future=channel.write(buf);

   // do something in parallel here ???

   int wrote = future.get().intValue();
}
```

This is just delayed blocking!

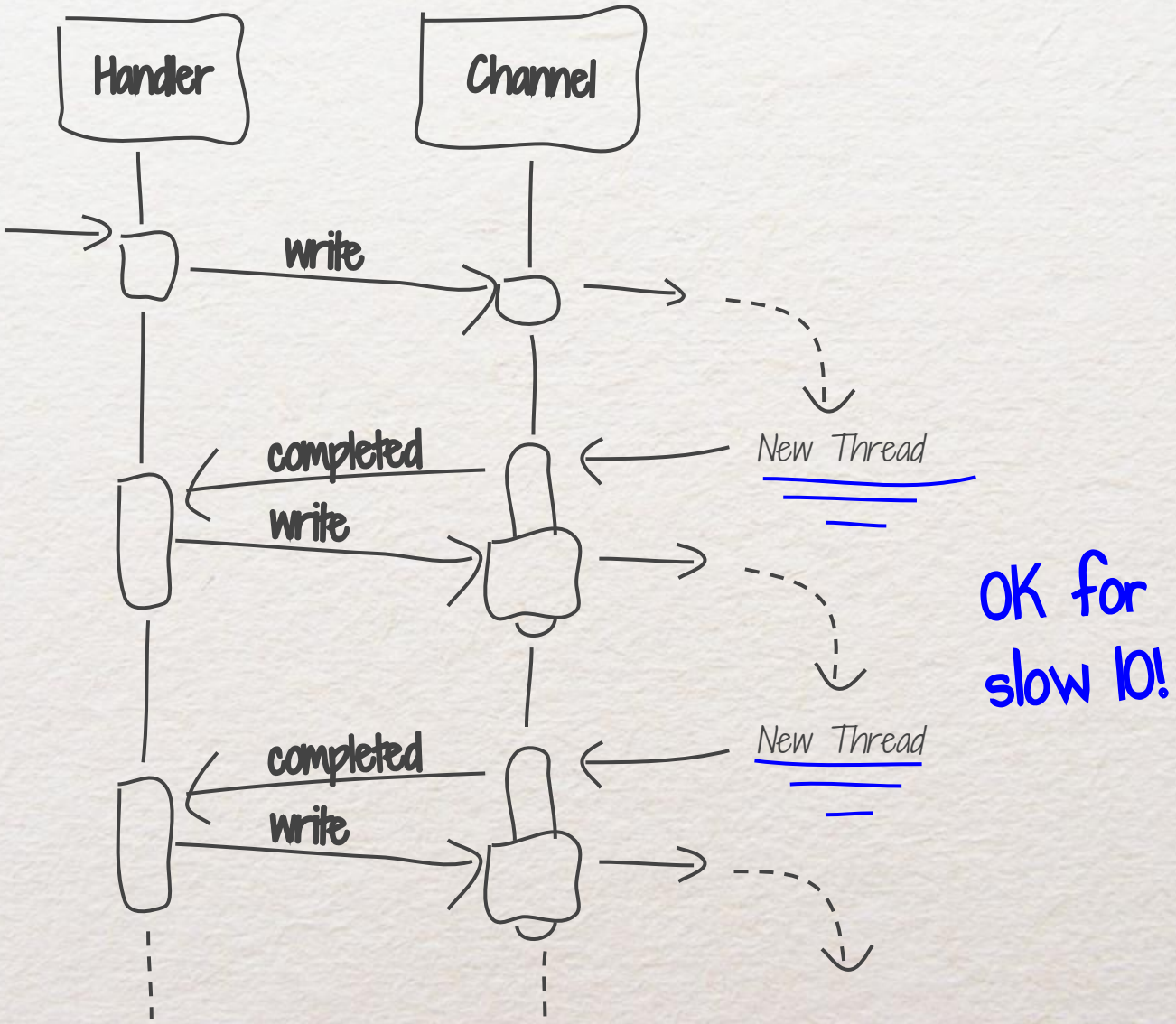
NIO? - Cumbersome Callbacks?

```
void asyncWriteFile(ByteChannel file)
{
   if (file.read(buf)<0)
      return;
   buf.flip();
   channel.write(buf,file,
      new CompletionHandler<Integer,FileChannel>()
      {
         public void completed(Integer w,FileChannel file)
         {
            asyncWriteFile(file);
         }
      }
   );
}
```

Verbose callback
not suitable for lambda :(
can be lots of garbage

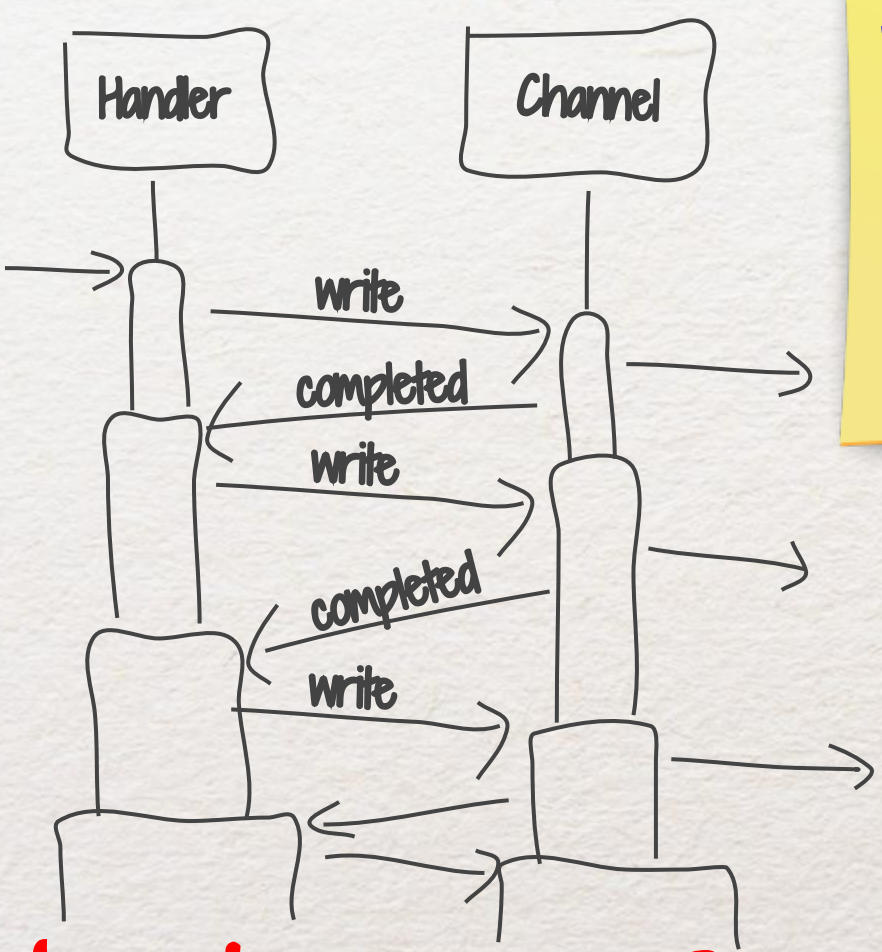
What Thread calls this?

Threadful Callbacks



OK for slow IO!

Threadless Callbacks - fast IO



Need a better paradigm for Async IO API!!

Stack Overflow!!!

Servlet 3.1 Async Output API

```
class ServletOutputStream extends OutputStream
{
   void setWriteListener(WriteListener l);
   boolean isReady();
   ...
}

interface WriteListener extends EventListener
{
   void onWritePossible() throws IOException;
   void onError(Throwable t);
}
```

An Async File Servlet

```
void doGet(ServletRequest req,final ServletResponse resp)
{
   final AsyncContext ctx=req.startAsync();
   final ServletOutputStream out = resp.getOutputStream();
   final FileInputStream file = ...;
   out.setWriteListener(new WriteListener()
   {
      void onWritePossible() throws IOException
      {
         while(out.isReady())
         {
            int len=file.read(buf);
            if (len<0)
            {
               ctx.complete();
               return;
            }
            out.write(buf,0,len);
         }
      }
   });
```

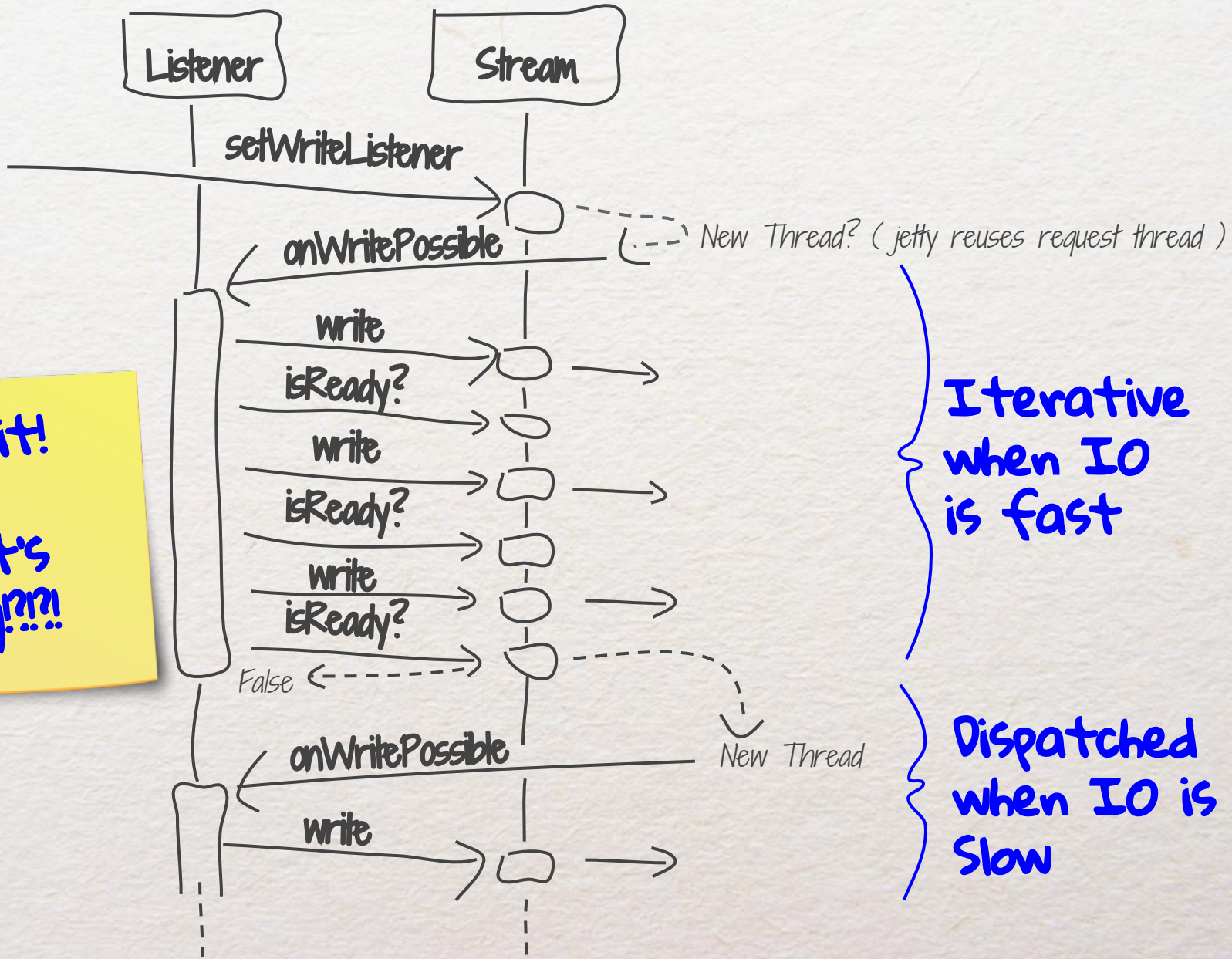
active methods!

may callback onWritePossible

Iterative for fast IO!!

Threaded for slow IO!!

Iterative Servlet IO



Got it!

That's easy!?!?!

Active isReady()

```
void onWritePossible()
{
   out.write(allContent,0,allContent.length);

   LOG.debug("all written {}",out.isReady());
}
```

BUG!!!

false return from isReady()
schedules a callback!

Be careful with isReady()

```
void onWritePossible()
{
   if (resultsAvailable() && out.isReady())
      out.write(getResults());
}
```

Possible BUG!!!

Order of conditionals is significant!

You must use isReady()

```
void onWritePossible()
{
   if (isReady())
   {
      out.write("<H1>Async Hello</H1>");
      out.write("Async Content!");
   }
}
```

BUG!!!

first write may be Pending!

Breaks GzipFilter & Wrappers

```
public void write(byte b[],
{
   if(!gzipping)
   {
      if(buf.space()<len)
      {
         gzipping=true;
         gzipAndWrite(buf,0,buf.remaining());
         gzipAndWrite(b,0,len);
      }
      else
         buffer.append(b,off,len);
   }
   else
      gzipAndWrite(b,0,len);
```

Looks like a Stream
Quacks like a Stream

IS NOT A STREAM!!!!!

BUG!!! isReady() not called between writes

Jetty provides AsyncGzipFilter

Don't misuse isReady()

```
void onWritePossible()
{
   if (isReady())
   {
      out.write("<H1>Async Hello</H1>");

      if (isReady())
         out.write("Async Content!");
   }
}
```

BUG!!!
May be written multiple times!

Can't mix and Match!

```
void doGet(ServletRequest req,final ServletResponse resp)
{
   final AsyncContext ctx=req.startAsync();
   final ServletOutputStream out = resp.getOutputStream();
   final FileInputStream file = ...;
   out.setWriteListener(new WriteListener()
   {
     void onWritePossible() throws IOException
     {
       try
       {
          ...
       }
       catch(Exception e)
       {
          resp.sendError(500,e.toStrin
       }
     }
   });
```

Also avoid:
sendRedirect()
getParameter()
getPart()
etc.

BUG!!!
May do stream output

All your buffers are belong to us

```
void doGet(ServletRequest req,final ServletResponse resp)
{
  out.setWriteListener(new WriteListener()
  {
    void onWritePossible() throws IOException
    {
      if (len==0 && len=file.read(buf)<0)
        {ctx.complete();return;}

      while(out.isReady())
      {
        out.write(buf,0,len);
        if (len=file.read(buf)<0){ctx.complete();return;}
      }
    }
  });
```

BUG!!!

you don't 'own' buffer here

Can't Use ByteBuffers?!?!?

```
RandomAccessFile file = new RandomAccessFile(file,"r"));
ByteBuffer allMyContent =
  raf.getChannel().map(MapMode.READ_ONLY,0,file.length());




void onWritePossible()
{
  if (allMyContent.hasRemaining())
    out.write(allMyContent);
  if (isReady())
    ctx.complete();
}
```

API is only OutputStream with byte[]!!

Can cast to org.eclipse.jetty.server.HttpOutput
Great for sendFile semantic with File mapping

Real uses of Async Servlet IO

org.eclipse.jetty:

- o.e.j.servlets.**DataRateLimitedServlet**
- o.e.j.proxy.**AsyncProxyServlet**
- o.e.j.fcgi.server.proxy.**FastCGIProxyServlet**
- o.e.j.servlets.**AsyncFileUploadServlet**

DataRateLimitedServlet

```
void onWritePossible()
{
   if(out.isReady())
   {
      byte[] buffer = new byte[buffersize];

      int len=out.read(buffer);
      if (len<0)
      {
         out.complete();
         return;
      }

      out.write(buffer,0,len);

      schedule(this,pauseNS,TimeUnit.NANOSECONDS);
   }
}
public void run(){ onWritePossible();}
```

Hold buffer while waiting?

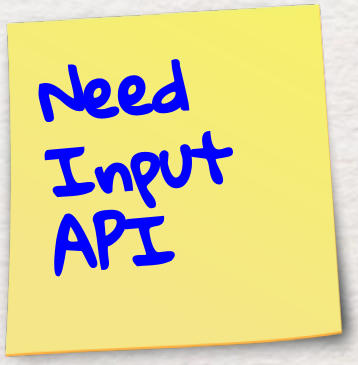
Developed for real time trading site

Protects QoS

Real uses of Async Servlet IO

org.eclipse.jetty:

- o.e.j.servlets.**DataRateLimitedServlet** ✔
- o.e.j.proxy.**AsyncProxyServlet**
- o.e.j.fcgi.server.proxy.**FastCGIProxyServlet**
- o.e.j.servlets.**AsyncFileUploadServlet**



Servlet 3.1 Async Input API

```
class ServletInputStream extends InputStream
{
   void setReadListener(ReadListener l);
   boolean isReady();
   boolean isFinished();
   ...
}

interface ReadListener extends EventListener
{
   void onDataAvailable() throws IOException;
   void onAllDataRead() throws IOException;
   void onError(Throwable t);
}
```

Not active!

AsyncProxyServlet

```
class StreamReader implements ReadListener, Callback
{
  public void onDataAvailable() throws IOException
  {
    ServletInputStream input = request.getInputStream();
    while (input.isReady() && !input.isFinished())
    {
      int len = input.read(buffer);
      if (len > 0)
      {
        proxy.write(ByteBuffer.wrap(buffer,0,len),this);
        break;
      }
    }
  }
  public void succeeded()
  {
    onDataAvailable();
  }
}
```

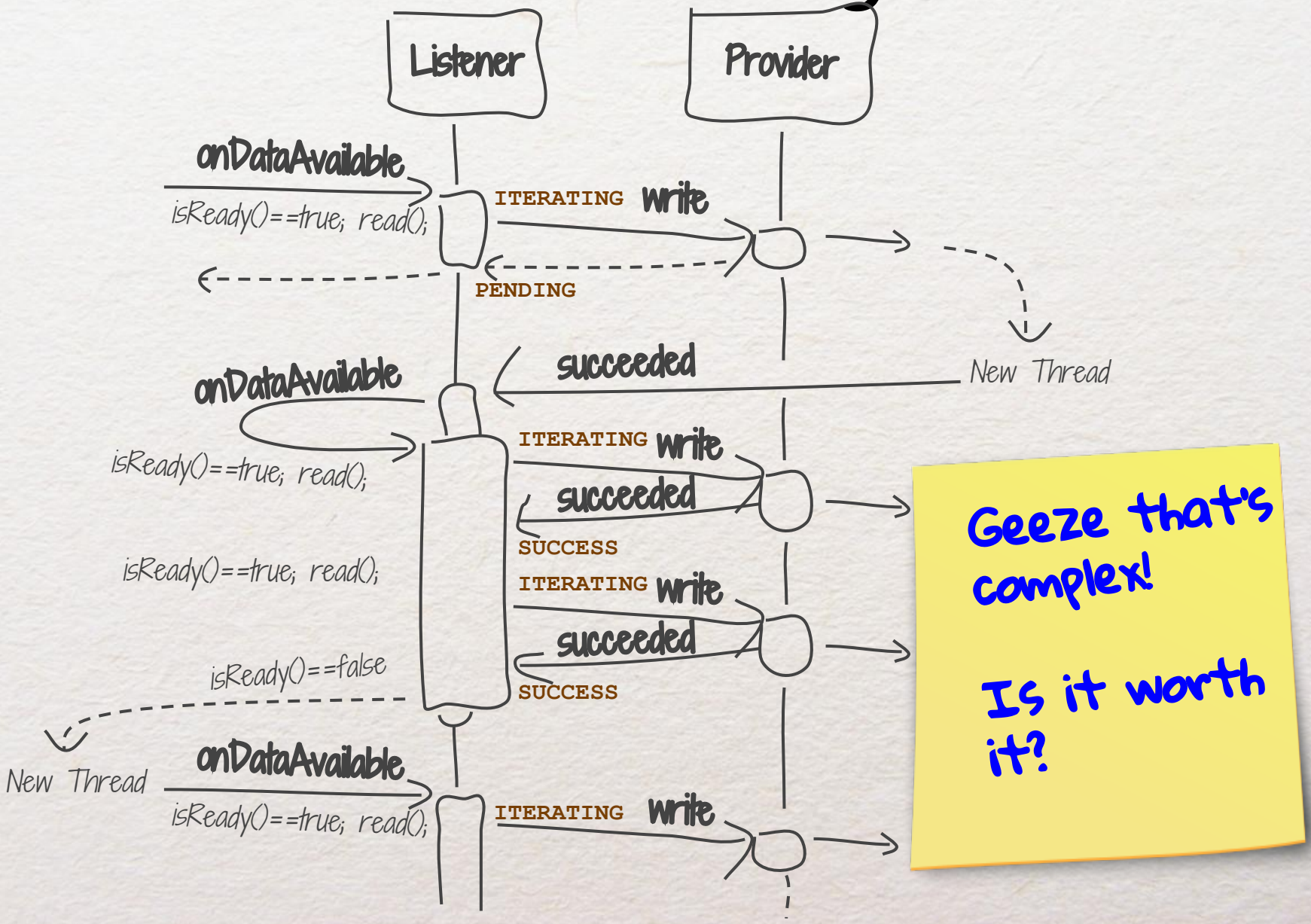
isReady()==true at EOF!
false would cause dispatch!

Stack Overflow!!!!

IteratingCallback

```
class IteratingCallback implements Callback {
  enum State { IDLE, ITERATING, PENDING, SUCCESS, FAILURE };
  final AtomicReference<State> state = new AtomicReference<>();
  public void iterate()
  {
    _state.set(ITERATING);
     while (true)
     {
        doSomethingAsync(this);
        if (_state.compareAndSet(ITERATING,PENDING))
           return;
     }
  }
  public void succeeded()
  {
    loop: while (true)
    {
      switch(state.get())
      {
        case ITERATING: if(!state.compareAndSet(ITERATING,SUCCESS)) continue;
          break loop;
        case PENDING: if (!state.compareAndSet(PENDING,SUCCESS)) continue;
          iterate();
          break loop;
      }
    }
  }
```

Async IO with IteratingCB



Asynchronous Echo Servlet...

```
void doGet(ServletRequest req, final ServletResponse resp)
{
  AsyncContext ctx = req.startAsync();
  Echo echo = new Echoer(ctx);
  req.getInputStream().setReadListener(echo);
  resp.getOutputStream().setWriteListener(echo);
}
class Echo implements ReadListener, WriteListener {
  private final byte[] buffer = new byte[SIZE];
  private final AsyncContext ctx;
  private final ServletInputStream input;
  private final ServletOutputStream output;
  private final AtomicBoolean complete=
    new AtomicBoolean(false);

  Echo(AsyncContext ctx) throws IOException  {
    this.ctx = ctx;
    input = ctx.getRequest().getInputStream();
    output = ctx.getResponse().getOutputStream();
  }
```

... Asynchronous Echo Servlet

```
  public void onWritePossible() throws IOException  {
    if (input.isFinished())
    {
      if (complete.compareAndSet(false,true))
        ctx.complete();
      return;
    }
    while (input.isReady())
    {
      int read = input.read(buffer);
      output.write(buffer, 0, read);
      if (!output.isReady())
        break;
    }
  }
  public void onDataAvailable() throws IOException  {
    if (output.isReady())
      onWritePossible();
  }
  public void onAllDataRead() throws IOException  {
    if (output.isReady() && complete.compareAndSet(false,true))
      ctx.complete();
  }
  public void onError(Throwable failure)  {
    failure.printStackTrace();
  }
}
```

Blocking Echo Servlet

```
void doGet(ServletRequest req, final ServletResponse resp)
{
   byte[] buf = new byte[SIZE];
   while (true)
   {
      int len = req.getInputStream().read(buf,0,SIZE);
      if (len < 0)
        break;
      resp.getOutputStream().write(buf, 0, len);
   }
}
```

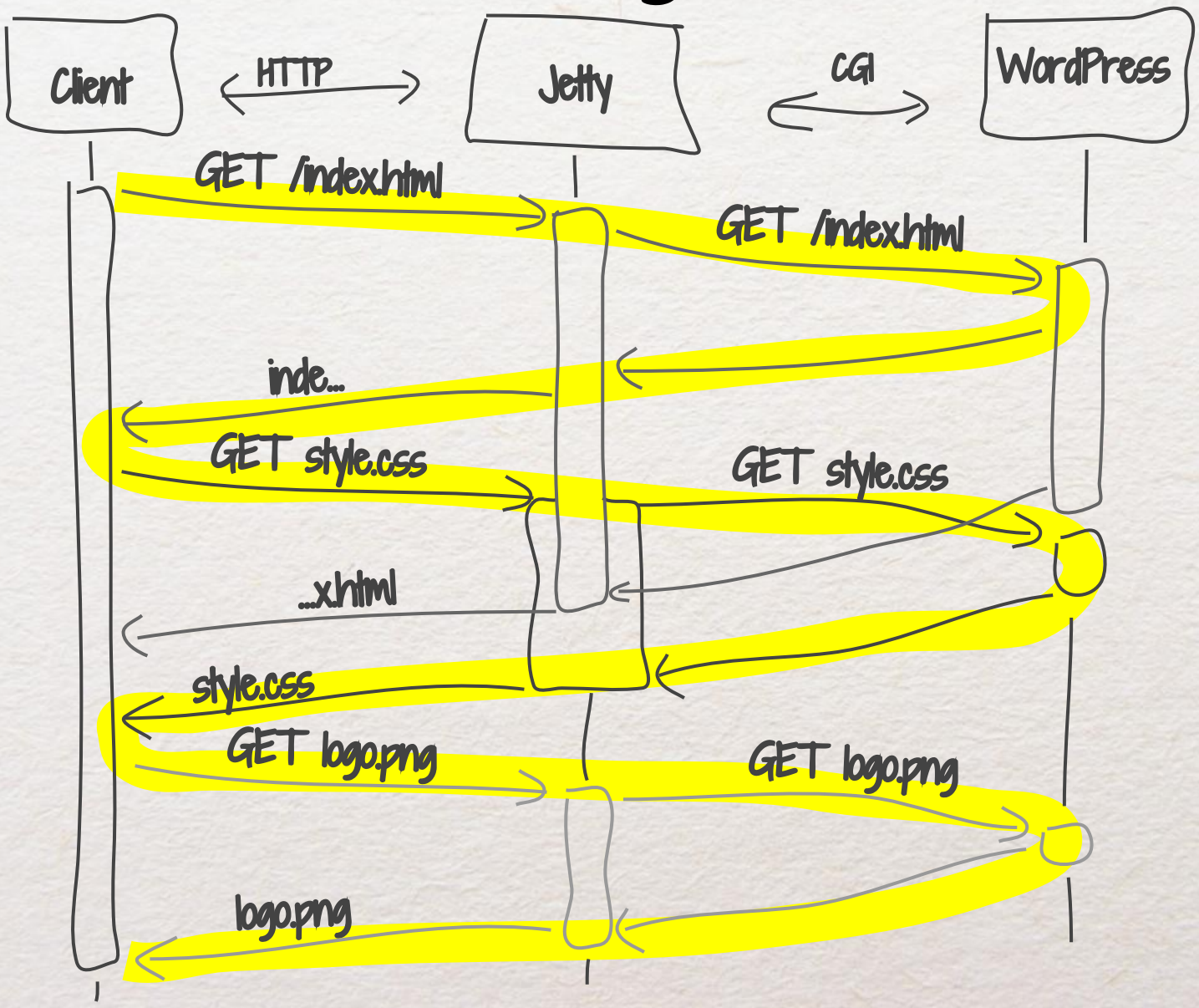
Geeze async is complex!

Is it worth it?

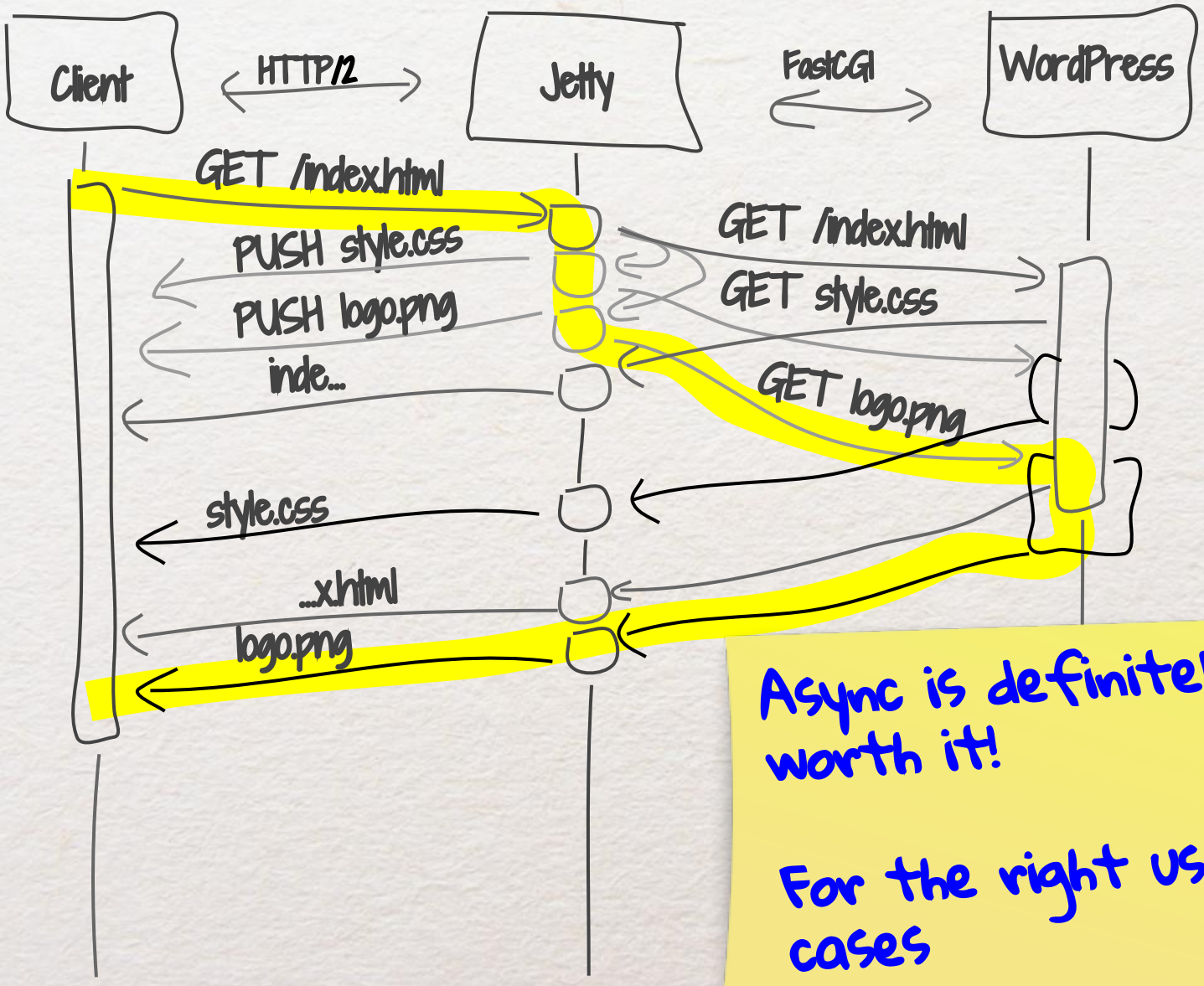
Async IO Demo

http://webtide.com/http2-test/push

HTTP and Blocking CGI



HTTP/2 and Async FastCGI



Async is definitely worth it!

For the right use-cases

Real uses of Async Servlet IO

org.eclipse.jetty:

- o.e.j.servlets.**DataRateLimitedServlet** ✔
- o.e.j.proxy.**AsyncProxyServlet** ✔
- o.e.j.fcgi.server.proxy.**FastCGIProxyServlet** ✔
- ~~o.e.j.servlets.**AsyncFileUploadServlet**~~

Not yet implemented!

Questions?

Greg Wilkins <gregw@webtide.com>
http://www.webtide.com