



# How GlassFish v3 build on top of Grizzly HTTP Extension Points

Jeanfrancois Arcand

Oleksiy Stashok

Sun Microsystems



# Agenda

- What is project Grizzly
- Grizzly's Extensions point
- Grizzly in v3
  - > What the monster is capable of
  - > Configuration
  - > Runtime
- Conclusion

# What is Project Grizzly

- Open Source Project <https://grizzly.dev.java.net>
- Open Sourced under CDDL/LGPL license.
- Very open community policy.
  - > All project communications are done on Grizzly mailing list. No internal, off mailing list conversations a-la-Sailfin.
  - > Project meetings open to anyone
- Project decisions are made by project member votes. Code contributions frequent.
- Approx ~35 000 download for the last 3 months
- Mailing list quite busy some day!

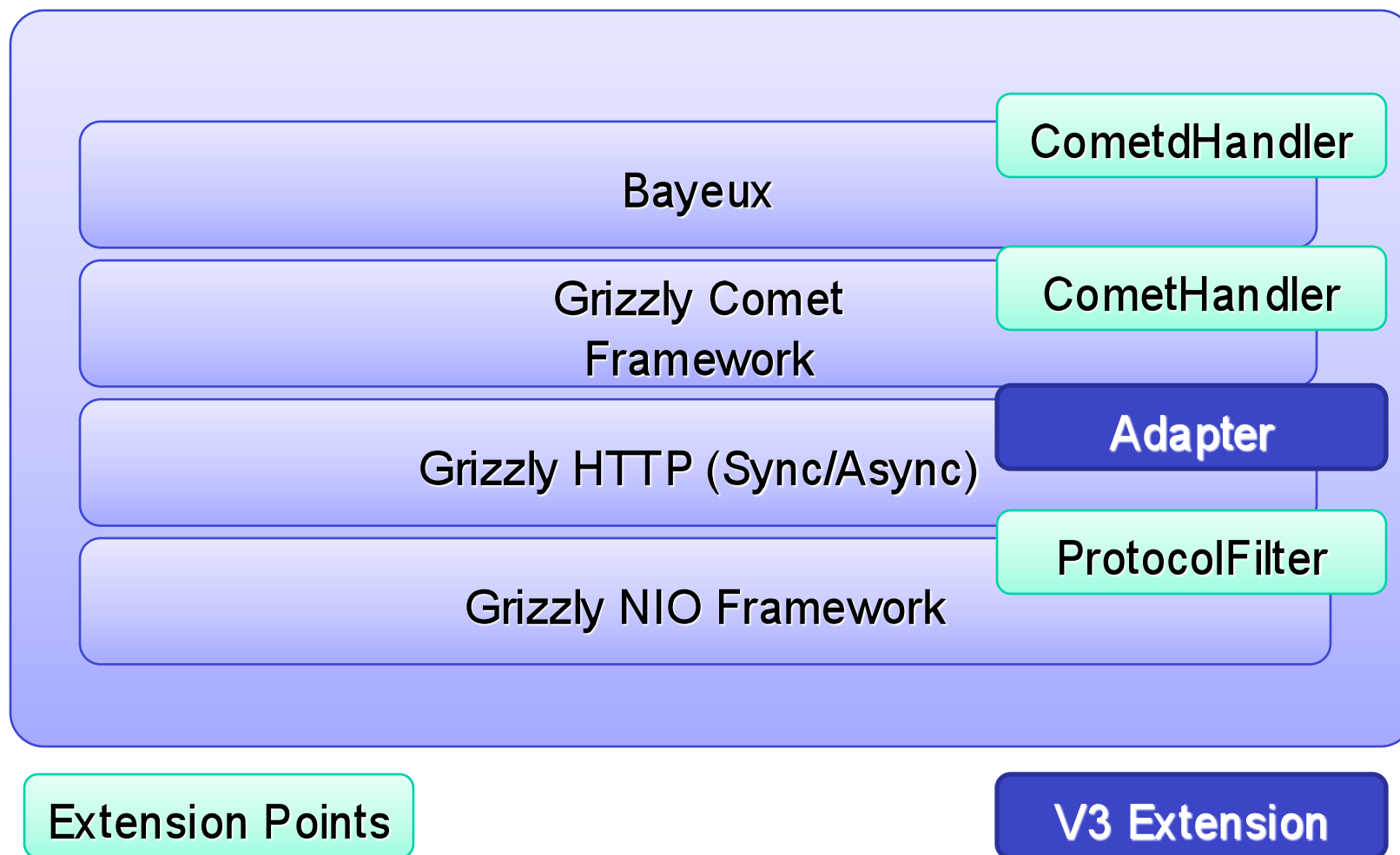
# What is project Grizzly

- Committers
  - > Oleksiy Stashok (Sun) – Lead (*all modules*)
  - > Jean-Francois Arcand (Sun) | (*all modules*)
  - > Shing Wai Chan (Sun) | (*http, comet, bayeux*)
  - > Ken Cavanaugh (Sun) | (*framework*)
  - > Charlie Hunt (Sun) | (*framework*)
  - > **John Vieten (Conclude GmbH)** | (*tutorial, framework*)
  - > **Sebastien Dionne (Consultant)** | (*tutorial, framework*)
  - > **Takai Naoto (ITOCHU Techno-Solution)** | (*jruby, bayeux*)
- Active users list...the community answer for us!

# What is Project Grizzly

- Uses Java NIO primitives and hides the complexity programming with Java NIO.
- Started in 2004 (Grizzly 1.0). Own project February 2007 (Grizzly 1.5)
- 1.0 represented extracted network layer of Glassfish
- Easy-to-use high performance APIs for TCP, UDP and SSL communications.
- Utilizes high performance buffers and buffer management.
- Choice of several different high performance thread pools.
- Ship with an HTTP module which is really easy to embed.

# Grizzly Extension Points



## Grizzly in v3 - Features

- Grizzly is a master piece of GlassFish v3. The implementation is called **grizzly-kernel**.
- Like in v2, handles all HTTP requests to the WebContainer.
- If the WebContainer hasn't been deployed/installed, serves directly static resources.
- If no Container deployed and jsp, jruby, groovy pages are dropped under domains/domain1/docroot/, automatically loads the associated WebContainer

## Grizzly in v3 – Features (Cont')

- Unlike in v2, handles all HTTP requests to other Containers:
  - > **\*ALL\*** Scripting extension are build on top of Grizzly's Extension Point (JRuby, Groovy, etc.)
  - > Admin CLI support is implemented as a Grizzly's Extension Point
  - > On-demand installation of Admin-Gui is also implemented as a Grizzly's Extension Point.
  - > Other Container (like Jetty) are also implemented using the same Extension Points.
- A lot of work for a little monster!

# Grizzly in v3 – Extension Points

- Grizzly is embedded in v3 by extending:
  - > **SelectorThread**: Main class in Grizzly to construct an HTTP/S based WebServer (was Grizzly11Protocol in v2)
  - > **Adapter**: All Extension in v3 (JRuby, Admin, etc.) implements that interface or its base implementation called **GrizzlyAdapter**
  - > **GrizzlyRequest/GrizzlyResponse**: Used to manipulate the request and the response
    - Ex: for the WebContainer, those objects are wrapped and exposed under HttpServletRequest/Response
  - > **Pipeline**: The thread pool used by Grizzly.
  - > **Mapper**: The object that maps requests to “something”.

Remember...

A bug in grizzly-  
kernel **IS NOT** a  
bug in Grizzly  
automatically

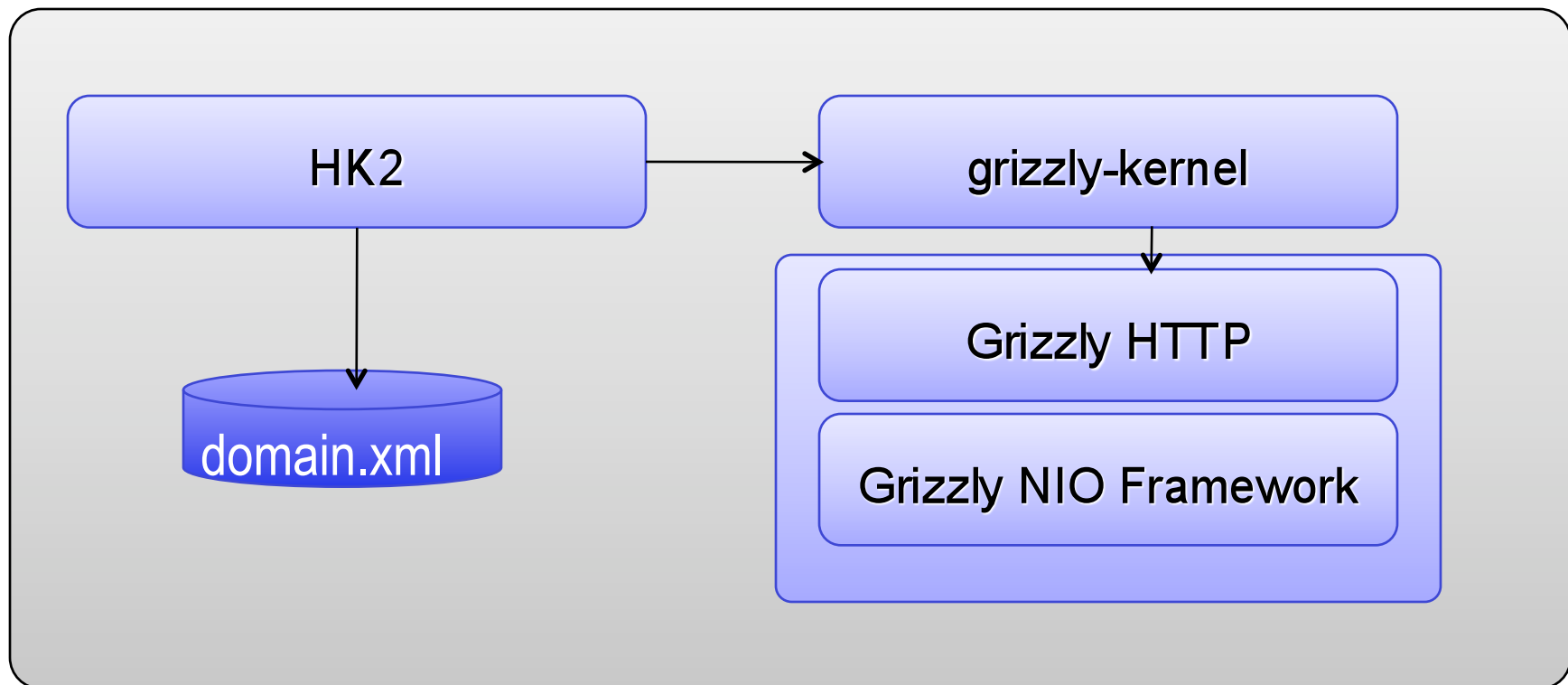
## grizzly-kernel: info

- Based on Grizzly 1.8.6.2.
- Grizzly is rebundled and re-OSGi-zed in v3. End modules located under
  - > modules/grizzly-module.jar (grizzly-framework/http)
  - > modules/grizzly-optionals.jar (grizzly-comet/bayeux)
- Code located under [v3/core/kernel/src/main/java/com/sun/enterprise/v3/services/impl](#)
- Handle ALL requests to GlassFish:
  - > Currently only http
  - > FCS will add IIOP, Soap over TCP, etc.

## grizzly-kernel: Main Classes

- > **GrizzlyServices**: Invoked at startup by hk2. Used to configure Grizzly using the domain.xml's http-service element (will be changed in v3 FCS for its own element)
- > **GrizzlyProxy**: One per http-listener. Represent an instance of a Grizzly Listener.
- > **ContainerMapper**: map request to the proper Container.  
**The most important class.**
- > **V3Mapper**: Contains information about which extension maps to which Container. Also contains all the information the WebContainer needs to associate a request with a host/servlet (hence shared object)
- > **GrizzlyProbePipeline**: v3 Thread pool, which extends Grizzly default thread pool by adding monitoring support.

# grizzly-kernel Configuration



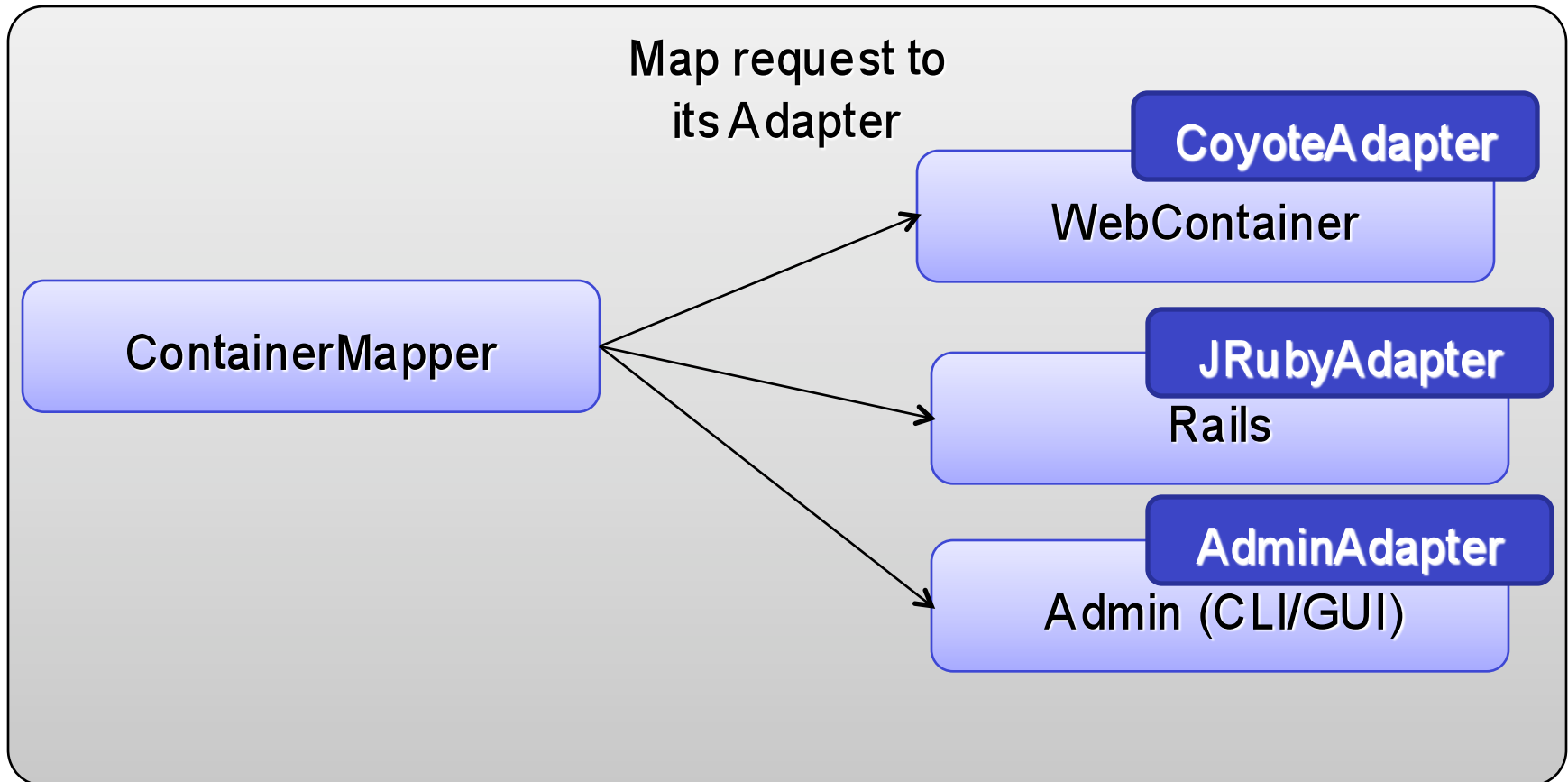
## At startup...

- When HK2/Felix starts, GrizzlyService is invoked as it use the @Service annotation
- From GrizzlyService, a GrizzlyProxy is created for each http-listener element in domain.xml.
- V3Mapper are also created and configured with:
  - > Adapter and its associated supported extension  
*ex: \*.jsp maps to then CoyoteAdapter (WebContainer)*  
*Everything on port 4848 maps to AdminAdapter*
- The ContainerMapper class is created with ready-to-use V3Mapper.

## At startup...

- If the WebContainer needs to be started, the V3Mapper is shared using the @Inject annotation.
- PEWebContainer (in web/web-glue module) will later grasp the V3Mapper and configure the WebContainer using the same instance that its associated GrizzlyProxy/ContainerMapper
- The WebContainer will use the V3Mapper to register all its deployed applications. The Container will then be able to map requests to the WebContainer.
- If Comet is enabled, a Grizzly AsyncFilter is added to its list of ProtocolFilter.

# grizzly-kernel Runtime



## At runtime...

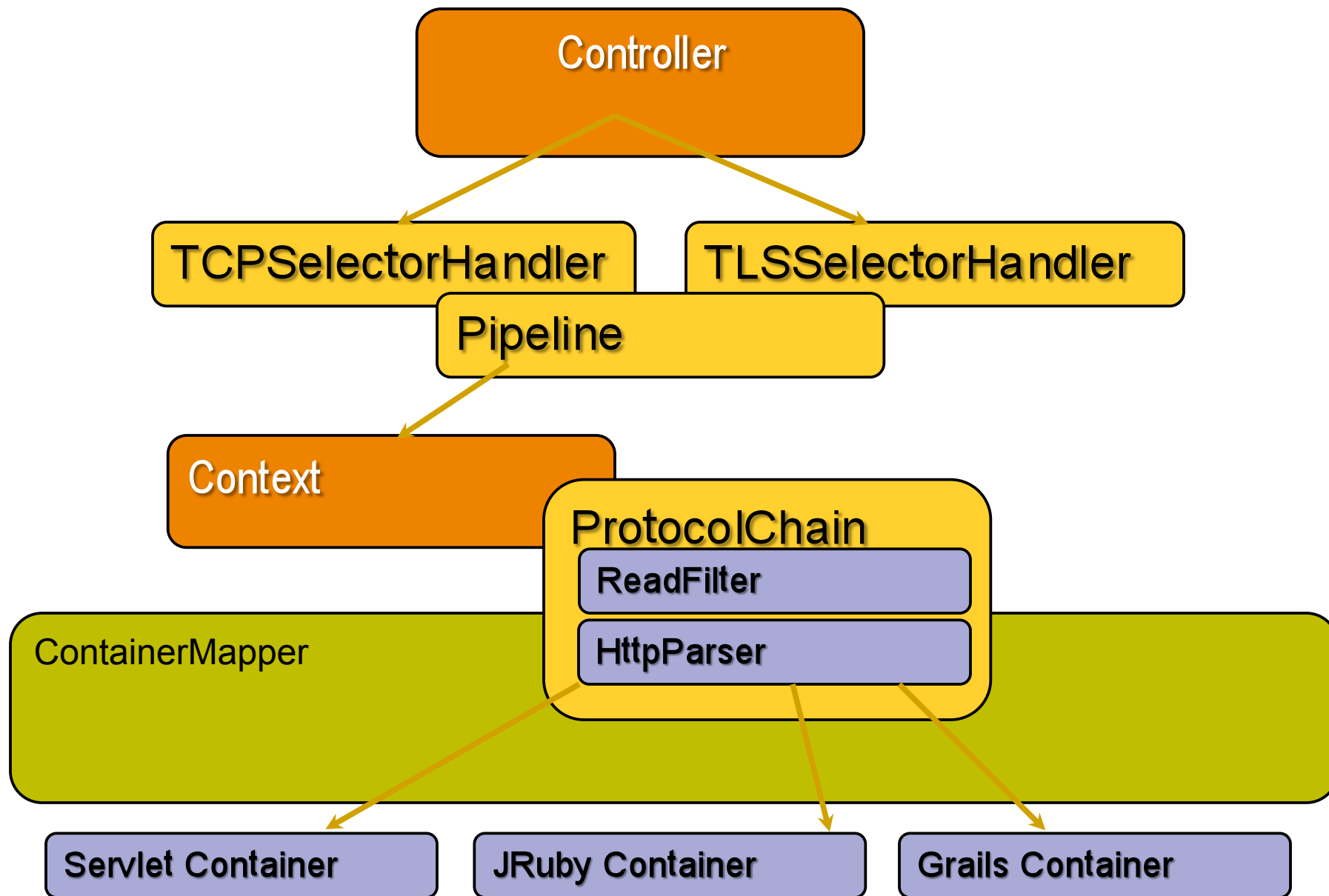
- When a request comes, Grizzly first creates the Request/Response and then invoke:

### **ContainerMapper.service (Request,Response)**

- This method is the where **everything** happens.
  1. First, decode the Request bytes (Prevent XSS/Dos/ Security attack)
  2. Try to map the \*decoded\* request to its associated Adapter.
  3. If **found**, invoke the Adapter.service() (Delegate the request/response processing to the mapped Container)

## At runtime...

- If **no mapping** has been found (no Adapter)
  1. Load all SnifferAdapter. SnifferAdapter are Sniffer extension that can be used to automatically starts an installed Container that hasn't been started yet.
  2. All SnifferAdapter are added to the V3Mapper.
  3. Try to map again. If a SnifferAdapter is found, invoke it `SnifferAdapter.service(Request,Response)`
  4. The SnifferAdapter will in turn start its associated Container, and register its Adapter to the V3Mapper
  5. Try to map the request again. If mapped, invoke the `Adapter.service(Request,Response)`
  6. **If not mapped, fall back to Grizzly's StaticResourcesAdapter.**



Remember...

A stack trace  
containing grizzly  
classes **IS NOT** a bug  
in Grizzly/Grizzly-  
Kernel automatically

# Conclusion

- Grizzly in v3 is a central piece, where everything happens
  - > *We no longer supports Tomcat's http listener*
  - > *We no longer supports Grizzly blocking mode*
- If we found a bug, we NO LONGER have workaround like in v1/v2. So we must release a patch as soon as possible.
- Every returned http status 404 are are suspicious. Always starts debugging `ContainerMapper.service()`.
  - > Might means no Adapter where mapped.

# For More Information

- Grizzly Active's Bloggers:
  - > Alexey: <http://blogs.sun.com/oleksiys/>
  - > Shing Wai: <http://blogs.sun.com/swchan/>
  - > John: <http://weblogs.java.net/blog/johnmann/>
  - > Sebastien: <http://www2.sebastiendionne.ca>
  - > Jeanfrancois: <http://weblogs.java.net/jfarcand>
- News:
  - > Every week or so, a list of interesting blogs talking about us  
<http://www.nabble.com/forum/Search.jtp?forum=23249&local=y&query=News>
- Project Grizzly mailing lists: [users/dev@grizzly.dev.java.net](mailto:users/dev@grizzly.dev.java.net)
- Download the JavaOne 2008 presentation for a technical overview of Grizzly Framework and Grizzly HTTP.