

Filters

- Since Java Servlet Specification 2.3
- Intercept request/response
 - perform operations on incoming requests
 - transform response before rendering
- Usually applied across multiple request paths, e.g. globally or some subset of servlets

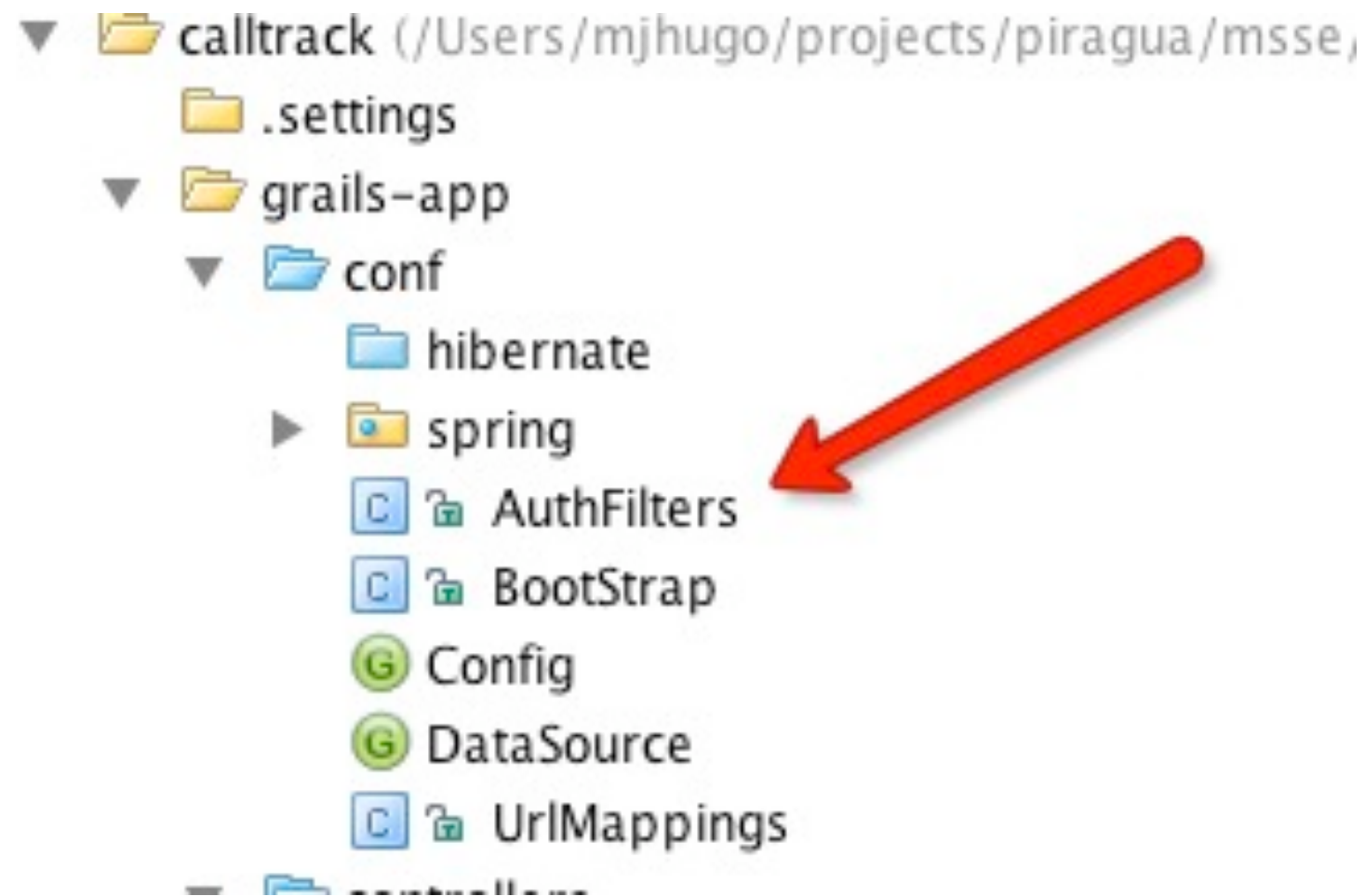
Examples

- Authentication
- Logging / Audit Trail
- Compression
- Localization
- Data Transformation

<http://www.oracle.com/technetwork/java/filters-137243.html>

Filters in Grails

- Name class *Filters.groovy
- place in grails-app/conf



Filters in Java

```
public final class HitCounterFilter implements Filter {
    private FilterConfig filterConfig = null;
    public void init(FilterConfig filterConfig)
        throws ServletException {
        this.filterConfig = filterConfig;
    }
    public void destroy() {
        this.filterConfig = null;
    }
    public void doFilter(ServletRequest request,
        ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        if (filterConfig == null)
            return;
        StringWriter sw = new StringWriter();
        PrintWriter writer = new PrintWriter(sw);
        Counter counter = (Counter)filterConfig.
            getServletContext().
            getAttribute("hitCounter");
        writer.println();
        writer.println("=====");
        writer.println("The number of hits is: " +
            counter.incCounter());
        writer.println("=====");

        // Log the resulting string
        writer.flush();
        filterConfig.getServletContext().
            log(sw.getBuffer().toString());
        ...
        chain.doFilter(request, wrapper);
        ...
    }
}
```

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
Application 2.3//EN" "http://java.sun.com/dtd/web-
app_2_3.dtd">
<web-app>
    <filter>
        <filter-name>HitCounterFilter</filter-name>
        <filter-class>HitCounterFilter</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>HitCounterFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

Filters in Grails

```
1  public class LoggingFilters {
2
3      def filters = {
4
5          logfilter (controller: '*', action: '*') {
6              before = {
7                  log.debug "requesting $controllerName: $actionName"
8              }
9          }
10     }
11 }
```

Filters

- Can be used across the entire application (unlike fine grained Controller Interceptors)
- Filter Types
 - before
 - after
 - afterView

Filter Variables

- **request** - The `HttpServletRequest` object
- **response** - The `HttpServletResponse` object
- **session** - The `HttpSession` object
- **servletContext** - The `ServletContext` object
- **flash** - The flash object
- **params** - The request parameters object
- **actionName** - The action name that is being dispatched to
- **controllerName** - The controller name that is being dispatched to
- **grailsApplication** - The Grails application currently running
- **applicationContext** - The `ApplicationContext` object

Filter Methods

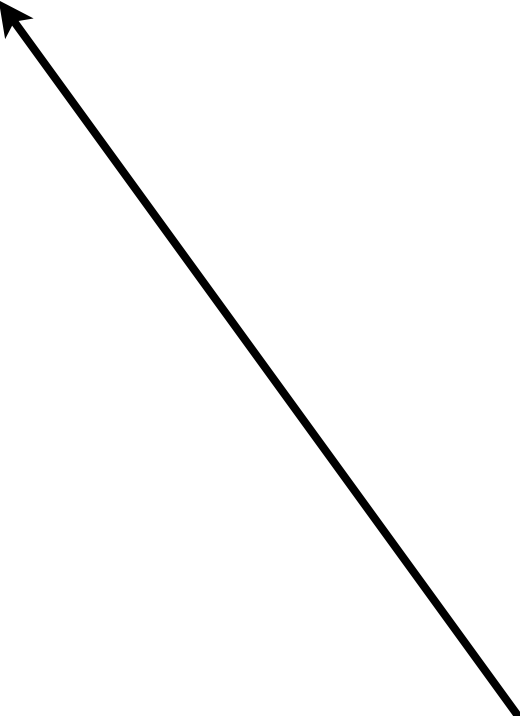
- redirect - For redirects to other controllers and actions
- render - For rendering custom responses / views

Applying Filters

- **controller** - controller matching pattern, by default * is replaced with .* and a regex is compiled
- **action** - action matching pattern, by default * is replaced with .* and a regex is compiled
- **regex** (true/false) - use regex syntax (don't replace '*' with '.*')
- **uri** - a uri to match, expressed with as Ant style path (e.g. /book/**)
- **find** (true/false) - rule matches with partial match (see `java.util.regex.Matcher.find()`)
- **invert** (true/false) - invert the rule (NOT rule)

All Controllers

```
all(controller: '*', action: '*') {  
  
}
```



arbitrary name for the filter, has no
impact on the filter logic

Just Book controller

```
book(controller: 'book', action: '*') {  
  
}
```

Not Book Controller

```
notBook(controller:'book', invert:true) {  
  
}
```

Regular Expression

```
saveInActionName (action: 'save', find: true) {  
  
}
```

Testing Filters

- use `@Mock(FilterName)` annotation in a controller test
- `withFilter()` method provided for you

<http://grails.org/doc/latest/guide/testing.html#unitTestingFilters>

Testing Filters

```
class CancellingFilters {  
  def filters = {  
    all(controller:"simple", action:"list") {  
      before = {  
        redirect(controller:"book")  
      }  
    }  
  }  
}
```

<http://grails.org/doc/latest/guide/testing.html#unitTestingFilters>

Testing Filters

```
@TestFor(SimpleController)
@Mock(CancellingFilters)
class SimpleControllerTests {
    void testInvocationOfListActionIsFiltered() {
        withFilters(action:"list") {
            controller.list()
        }
        assert response.redirectedUrl == '/book'
    }
}
```

<http://grails.org/doc/latest/guide/testing.html#unitTestingFilters>