Documentation
- Main
- Getting Started
- The Sling Engine
- Development
- Bundles
- Tutorials & How-Tos
- Maven Plugins
- Configuration

API Docs
- Sling 11
- Sling 10
- Sling 9
- All versions

Support
- Wiki
- FAQ
- Site Map

Project Info
- Downloads
- License
- News
- Releases
- Issue Tracker
- Links
- Contributing
- Project Information
- Security

Source
- GitHub
- Git at Apache

Apache Software Foundation
- Thanks!
- Become a Sponsor
- Buy Stuff

Servlet Filter Support

Sling supports filter processing by applying filter chains to the requests before actually dispatching to the servlet or script for processing. Filters to be used in such filter processing are plain OSGi services of type javax.servlet.Filter which of course means that the services implement this interface.

> See SLING-1213, SLING-1734, and Registering filters with Sling for more details. The NoPropertyFilter from our integration tests shows an example Sling Filter.

For Sling to pick up a javax.servlet.Filter service for filter processing two service registration properties are inspected:

Property	Type	Default Value	Valid Values	Description
sling.filter.scope	String, String[] or Vector<String>	request	REQUEST, INCLUDE, FORWARD, ERROR, COMPONENT	Indication of which chain the filter should be added to. This property is required. If it is missing from the service, the service is ignored because it is assumed another consumer will be interested in using the service. Any unknown values of this property are also ignored causing the service to be completely ignored if none of the values provided by the property are valid. See below for the description of the filter chains.
service.ranking	Integer	0	Any Integer value	Indication of where to place the filter in the filter chain. The higher the number the earlier in the filter chain. This value may span the whole range of integer values. Two filters with equal service.ranking property value (explicitly set or default value of zero) will be ordered according to their service.id service property as described in section 5.2.5, Service Properties, of the OSGi Core Specification R 4.2.
sling.filter.pattern	String	(-)	Any String value	Restrict the filter to paths that match the supplied regular expression. Requires Sling Engine 2.4.0.
sling.filter.suffix.pattern	String	(-)	Any String value	Restrict the filter to requests with suffix that match the supplied regular expression. Requires Sling Engine 2.6.14.
sling.filter.selectors	String[]	(-)	Any String value	Restrict the filter to requests whose selectors match one or more of the provided ones. Requires Sling Engine 2.6.14.
sling.filter.methods	String[]	(-)	Any String value	Restrict the filter to requests whose methods match one or more of the provided ones. Requires Sling Engine 2.6.14.
sling.filter.resourceTypes	String[]	(-)	Any String value	Restrict the filter to requests whose resource type match one of the provided ones. Requires Sling Engine 2.6.14.

Property	Type	Default Value	Valid Values	Description
sling.filter.extensions	String[]	(-)	Any String value	Restrict the filter to requests whose extension matches one of the provided ones. Requires Sling Engine 2.6.14.

SlingServletFilter Annotation

Coding the above being a bit tedious, Apache Sling Servlets Annotations 1.1.0 provides handy `SlingServletFilter annotation to set those values:

```
...
  import org.apache.sling.servlets.annotations.SlingServletFilter;
  import org.apache.sling.servlets.annotations.SlingServletFilterScope;
  import org.osgi.service.component.annotations.Component;
...
  @Component
  @SlingServletFilter(scope = {SlingServletFilterScope.REQUEST},
                      suffix_pattern = "/suffix/foo",
                      resourceTypes = {"foo/bar"},
                      pattern = "/content/.*",
                      extensions = {"txt","json"},
                      selectors = {"foo","bar"},
                      methods = {"GET","HEAD"})
  public class FooBarFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {

    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOExcep
      SlingHttpServletResponse slingResponse = (SlingHttpServletResponse)response;
      //will only be run on (GET|HEAD) /content/.*.foo|bar.txt|json/suffix/foo requests
      //code here can be reduced to what should actually be done in that case
      //for other requests, this filter will not be in the call stack
      slingResponse.addHeader("foobared", "true");
      chain.doFilter(request, slingResponse);
    }

    @Override
    public void destroy() {

    }
  }
```

Filter Chains

Sling maintains five filter chains: request level, component level, include filters, forward filters and error filters. Except for the component level filter these filter chains correspond to the filter <dispatcher> configurations as defined for Servlet API 2.5 web applications (see section SRV.6.2.5 Filters and the RequestDispatcher).

The following table summarizes when each of the filter chains is called and what value must be defined in the sling.filter.scope property to have a filter added to the respective chain:

sling.filter.scope	Servlet API Correspondence	Description
REQUEST	REQUEST	Filters are called once per request hitting Sling from the outside. These filters are called after the resource addressed by the request URL and the Servlet or script to process the request has been resolved before the COMPONENT filters (if any) and the Servlet or script are called.
INCLUDE	INCLUDE	Filters are called upon calling the RequestDispatcher.include method after the included resource and the Servlet or script to process the include have been resolved before the Servlet or script is called.
FORWARD	FORWARD	Filters are called upon calling the RequestDispatcher.forward method after the included resource and the Servlet or script to process the include

sling.filter.scope	Servlet API Correspondence	Description
		have been resolved before the Servlet or script is called.
ERROR	ERROR	Filters are called upon HttpServletResponse.sendError or any uncaught Throwable before resolving the error handler Servlet or script.
COMPONENT	REQUEST,INCLUDE,FORWARD	The COMPONENT scoped filters are present for backwards compatibility with earlier Sling Engine releases. These filters will be called among the INCLUDE and FORWARD filters upon RequestDispatcher.include or RequestDispatcher.forward as well as before calling the request level Servlet or script after the REQUEST filters.

Note on INCLUDE and FORWARD with respect to JSP tags: These filters are also called if the respective including (e.g. <jsp:include> or <sling:include>) or forwarding (e.g. <jsp:forward> or <sling:forward>) ultimately calls the RequestDispatcher.

Filter Processing

Filter processing is part of the Sling request processing, which may be sketched as follows:

- *Request Level*:
  - Authentication
  - Resource Resolution
  - Servlet/Script Resolution
  - Request Level Filter Processing

The first step of request processing is the *Request Level* processing which is concerned with resolving the resource, finding the appropriate servlet and calling into the request level filter chain. The next step is the *Component Level* processing, calling into the component level filters before finally calling the servlet or script:

- *Component Level*:
  - Component Level Filter Processing
  - Call Servlet or Script

When a servlet or script is including or forwarding to another resource for processing through the RequestDispatcher (or any JSP tag or other language feature ultimately using a RequestDispatcher) the following *Dispatch* processing takes place:

- *Dispatch*:
  - Resolve the resource to dispatch to if not already defined when getting the RequestDispatcher
  - Servlet/Script resolution
  - Call include or forward filters depending on the kind of dispatch
  - Call Servlet or Script

As a consequence, request level filters will be called at most once during request processing (they may not be called at all if a filter earlier in the filter chain decides to terminate the request) while the component level, include, and forward filters may be called multiple times while processing a request.

Disabling Filters

As hinted earlier in the page, a filter is ignored if you set an invalid sling.filter.scope. To disable a specific filter, you can deploy an OSGi config setting an invalid sling.filter.scope, for instance *disabled*.

Troubleshooting

Apart form the logs which tell you when filters are executed, two Sling plugins provide information about filters in the OSGi console.

Recent Requests plugin

The request traces provided at /system/console/requests contain information about filter execution, as in this example:

```
0 (2010-09-08 15:22:38) TIMER_START{Request Processing}
...
0 (2010-09-08 15:22:38) LOG Method=GET, PathInfo=/some/path.html
3 (2010-09-08 15:22:38) LOG Applying request filters
3 (2010-09-08 15:22:38) LOG Calling filter: org.apache.sling.bgservlets.impl.BackgroundServletStarterFilter
3 (2010-09-08 15:22:38) LOG Calling filter: org.apache.sling.portal.container.internal.request.PortalFilter
3 (2010-09-08 15:22:38) LOG Calling filter: org.apache.sling.rewriter.impl.RewriterFilter
3 (2010-09-08 15:22:38) LOG Calling filter: org.apache.sling.i18n.impl.I18NFilter
3 (2010-09-08 15:22:38) LOG Calling filter: org.apache.sling.engine.impl.debug.RequestProgressTrackerLogFilter
3 (2010-09-08 15:22:38) LOG Applying inner filters
3 (2010-09-08 15:22:38) TIMER_START{/some/script.jsp#0}
...
8 (2010-09-08 15:22:38) TIMER_END{8,Request Processing} Request Processing
```

Config Status plugin

The configuration status page at /system/console/config includes the current list of active filters in its *Servlet Filters* category, as in this example:

Current Apache Sling Servlet Filter Configuration

Request Filters:
-2147483648 : **class org**.apache.sling.bgservlets.impl.BackgroundServletStarterFilter (2547)
-3000 : **class org**.apache.sling.portal.container.internal.request.PortalFilter (2562)
-2500 : **class org**.apache.sling.rewriter.impl.RewriterFilter (3365)
-700 : **class org**.apache.sling.i18n.impl.I18NFilter (2334)
0 : **class org**.apache.sling.engine.impl.debug.RequestProgressTrackerLogFilter (2402)

Error Filters:

Include Filters:

Forward Filters:
1000 : **class some**.package.DebugFilter (2449)

Component Filters:
-200 : **class some**.package.SomeComponentFilter (2583)

The first numbers on those lines are the filter priorities, and the last number in parentheses is the OSGi service ID.

Support in Sling Engine 2.1.0

Up to and including Sling Engine 2.1.0 support for Servlet Filters has been as follows:

- Any javax.servlet.Filter service is accepted as a filter for Sling unless the pattern property used by the Apache Felix HttpService whiteboard support is set in the service registration properties.
- The filter.scope property is optional and supports the case-sensitive values request and component.
- Filter ordering is defined by the filter.order property whose default value is Integer.MAX_VALUE where smaller values have higher priority over higher values.

Last modified by Nicolas Peltier on Mon Sep 17 15:01:50 2018 +0200