Debugging package

The Core module comes equipped with several classes that can be used to troubleshoot and/or debug your Magnolia instance. This page covers each of these classes and how they should be used. Always use caution when placing a filter in the chain manually and keep in mind that position in the chain matters. Depending on what you are trying to debug you may need to use a filter multiple times or even move it to different positions within the chain.

Never place a filter **before** the ContextFilter. Doing so will break your filter chain.

- Call tracking
- Derby test persistence manager
- Dump headers
- Logging response
- Performance test
- Session debugger

Call tracking

class: `info.magnolia.debug.CallTrackingFilter`

This filter logs whatever has been recorded with the `TrackingStatus`.

Deployment

To deploy the filter:

1. Download this XML: config.server.filters.callTrackingFilter.xml
2. Open the Configuration app and click the node: `/server/filters`
3. From the action bar, click "Import"

   The filter will be imported at the bottom of the chain.

4. Drag (or move) the filter into position.
5. Enable the filter by changing the `enabled` property to `true`.

Usage

Sometimes when debugging code you might need to track the number of calls made to method executed very deep down in the hierarchy on a per-request basis. Analyzing this with profiler can be very cumbersome. While it's easy to see the total time required for complete operations, in some cases, you might need to see what each request thread was executing and how long that took. To get that data this filter analyzes the stacktrace and provides aggregated info about the call hierarchy.

Like the other filters in this package you need to enable the CallTrackingFilter in the filter chain as mentioned above. You will also need to call the `track()` method at the place where you want to see aggregated info from all the invocations.

Finally, here is an example output that shows the aggregated stacktrace from all invocations including counts of each execution path. With this information in hand, you can find (and hopefully fix) excessive calls.

```
2013-01-30 15:04:45,416 INFO  info.magnolia.debug:
/magnoliaAuthor/demo-project.html:inits: 54, total calls: 14714, total time: 191947000
   14714;info.magnolia.debug.TrackingStatus.track(TrackingStatus.java:82)
     14714;info.magnolia.cms.util.SimpleUrlPattern.match(SimpleUrlPattern.java:212)
       4;info.magnolia.voting.voters.URIPatternVoter.boolVote(URIPatternVoter.java:60)

4;info.magnolia.cms.beans.config.DefaultVirtualURIMapping.mapURI(DefaultVirtualURIMapping.java:56)
    14706;info.magnolia.cms.security.PermissionImpl.match(PermissionImpl.java:89)
      4;info.magnolia.voting.voters.AbstractBoolVoter.vote(AbstractBoolVoter.java:62)
      4;info.magnolia.cms.beans.config.VirtualURIManager.getURIMapping(VirtualURIManager.java:107)
      14706;info.magnolia.cms.security.AccessManagerImpl.getPermissions(AccessManagerImpl.java:87)
        4;info.magnolia.cms.filters.VirtualUriFilter.getURIMapping(VirtualUriFilter.java:100)
        14706;info.magnolia.cms.security.AccessManagerImpl.isGranted(AccessManagerImpl.java:64)
        4;info.magnolia.voting.DefaultVoting.vote(DefaultVoting.java:56)
          14706;info.magnolia.cms.core.Access.isGranted(Access.java:63)
          4;info.magnolia.cms.filters.AbstractMgnlFilter.bypasses(AbstractMgnlFilter.java:121)
          4;info.magnolia.cms.filters.VirtualUriFilter.doFilter(VirtualUriFilter.java:66)

3742;info.magnolia.cms.core.DefaultHierarchyManager.isGranted(DefaultHierarchyManager.java:422)
    4;info.magnolia.cms.filters.AbstractMgnlFilter.doFilter(AbstractMgnlFilter.java:88)
    47;info.magnolia.cms.core.DefaultHierarchyManager.isExist(DefaultHierarchyManager.java:396)
    4;info.magnolia.cms.filters.AbstractMgnlFilter.matches(AbstractMgnlFilter.java:94)
    10915;info.magnolia.cms.core.DefaultContent.newNodeDataInstance(DefaultContent.java:229)
    2;info.magnolia.cms.security.URISecurityFilter.isAuthorized(URISecurityFilter.java:98)
      2;info.magnolia.cms.security.URISecurityFilter.isAllowed(URISecurityFilter.java:82)
      4;info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:81)
      246;info.magnolia.cms.core.DefaultContent.(DefaultContent.java:126)
      4;info.magnolia.cms.filters.MgnlFilterChain.doFilter(MgnlFilterChain.java:82)
```

Removal

Once you are finished with the filter it's a good idea to remove it or at least turn it off so that it stops logging. Remember, since every request will produce an output, the log can grow quickly.

> The best thing to do is to disable the filter. This way if you need it again in the future you can just turn it back on.

To disable the filter:

1. Change the `enabled` property to `false`.
2. (Optional) Move the filter back down to the bottom of the chain.

Derby test persistence manager

class: `info.magnolia.debug.DerbyTestPersistenceManager`

The persistence manager implementation measures the time spent saving data. You must also enable the `PerformanceTestFilter`.

> While this specific implementation is meant to be used with Derby database it should be quite straightforward to create an implementation for any other database.

Deployment

To deploy this class:

1. Create a new repository configuration which uses the test persistence manager. Here is an example: jackrabbit-bundle-derby-test-search.xml

   ```
   <PersistenceManager class="info.magnolia.debug.DerbyTestPersistenceManager">
     <param name="url" value="jdbc:derby:${wsp.home}/db;create=true" />
     <param name="schemaObjectPrefix" value="${wsp.name}_" />
   </PersistenceManager>
   ```

2. In your properties file update the JR config to point to the test configuration.

```
magnolia.repositories.home=${magnolia.home}/repositories-test
magnolia.repositories.jackrabbit.config=WEB-INF/config/repo-conf/jackrabbit-bundle-derby-test-search.xml
```

3. Start up the instance and let it create a new test repository.
4. Configure the PerformanceTestFilter behind the context filter (see instructions below).

> We need the PerformanceTestFilter because this is the filter that will print the time data to the standard log. The time data is stored in a PerformanceTestStatus object.

Usage

The point of this class is to examine how long it takes to persist (store) data.

> Typically it's not a good idea to use an embedded database in a production environment. The embedded database will have to share memory with the application.

Let's look at an example where we upload a large file (100M+) into the DAM so we can measure the time it takes to persist it. Create a new asset in the DAM and upload the new file in the detail subapp. Once you click save, you should see some information printed to the standard log via the PerformanceTestFilter. Times are in milliseconds.

```
2019-06-14 11:16:45,381 INFO  info.magnolia.debug : /magnoliaAuthor/.magnolia/admincentral/UIDL/ :
performanceTestFilter: 2092, pm-store: 680, pm-blob: 670
```

Removal

Once the test is complete revert back to your repository.

1. In your properties file, update the JR config to point to the original configuration.

```
magnolia.repositories.home=${magnolia.home}/repositories
magnolia.repositories.jackrabbit.config=WEB-INF/config/repo-conf/jackrabbit-bundle-derby-search.xml
```

Dump headers

class: `info.magnolia.debug.DumpHeadersFilter`

This filter dumps the headers for the request and the response.

Deployment

To deploy the filter:

1. Download this XML: config.server.filters.dumpHeaders.xml
2. Open the Configuration app and click the node: `/server/filters`
3. From the action bar, click "Import"

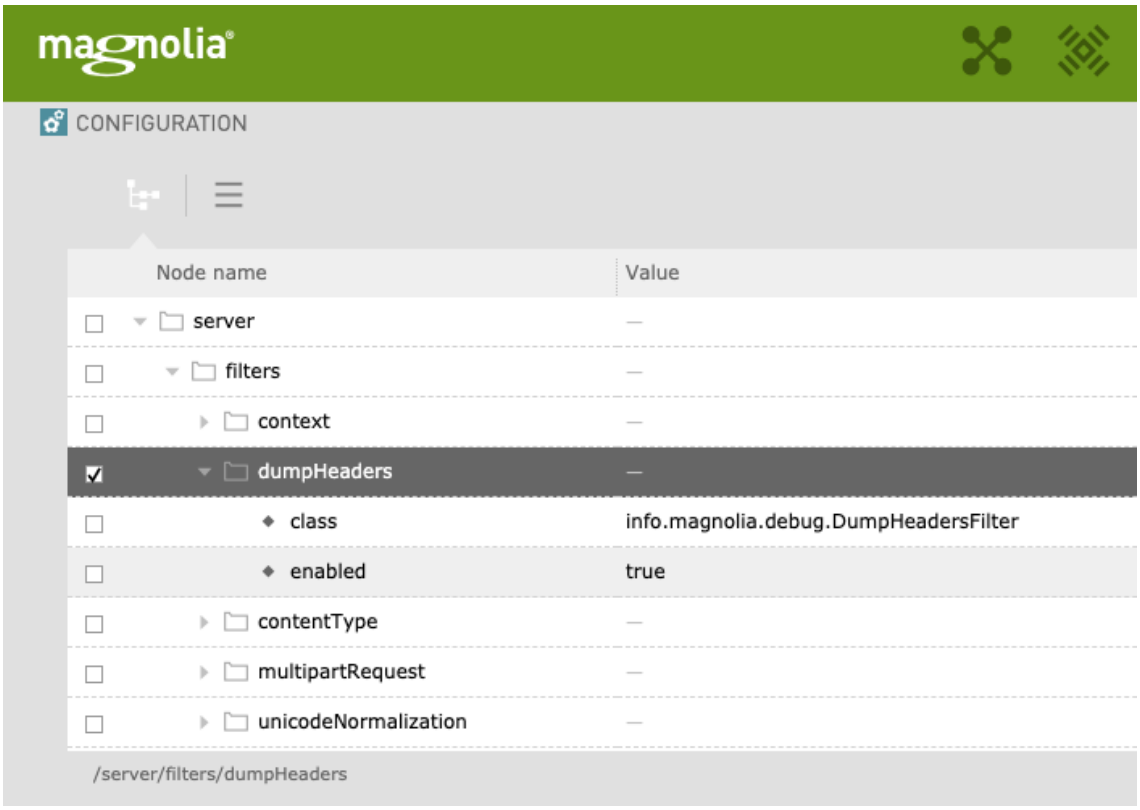
> The filter is imported at the bottom of the chain.

4. Drag (or move) the filter into position.
5. Enable the filter by changing the `enabled` property to `true`.

Usage

The point fo the DumpHeadersFilter is to print header information for each request and response to the standard log. This can be helpful for examining requests and their origin. Especially when the request passes through other servers before arriving in Magnolia. On the response side of things, double check that a specific header is being set before leaving Magnolia.

Let's look at a specific example of deploying to filter at the top of the chain. This is probably the most common use case of the filter. We place the filter near the top of the chain right behind the context filter. Never place the filter above context because this will break the system.



Once the filter is deployed you can begin examining the requests. The request does not need to be for a website. Remember that all requests, even those for Admincentral, go through the filter chain. This means any request could be examined and (with this filter deployed) *all* requests will produce output in the log.

Here we see an example of requesting the travel home page:

Each request will have a number associated with it. This way we can match the request to the response. In the example below, that number is 8.

```
# Request
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 uri: /magnoliaPublic/travel.html
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 session: true
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 parameters:
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 method: GET
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 header: host=localhost:8080
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 header: connection=keep-alive
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 header: cache-control=max-age=0
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 header: upgrade-insecure-requests=1
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 header: user-agent=Mozilla/5.0 (Macintosh; Intel Mac OS X
10_14_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36 OPR/60.0.3255.160
2019-06-12 16:32:03,969 INFO  info.magnolia.debug : 8 header: accept=text/html,application/xhtml+xml,application
/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
2019-06-12 16:32:03,970 INFO  info.magnolia.debug : 8 header: accept-encoding=gzip, deflate, br
2019-06-12 16:32:03,970 INFO  info.magnolia.debug : 8 header: accept-language=en-US,en;q=0.9
2019-06-12 16:32:03,970 INFO  info.magnolia.debug : 8 header:
cookie=JSESSIONID=214520A11DA4335D81735380B0BA06F8; _first_pageview=1; _jsuid=4186500974; no_tracky_100843699=1

# Response
2019-06-12 16:32:04,130 INFO  info.magnolia.debug : 8 response status: 200
2019-06-12 16:32:04,130 INFO  info.magnolia.debug : 8 response length: 5483
2019-06-12 16:32:04,130 INFO  info.magnolia.debug : 8 response mime type: text/html;charset=UTF-8
2019-06-12 16:32:04,130 INFO  info.magnolia.debug : /magnoliaPublic/travel.html response: X-Magnolia-
Registration = Registered
```

Removal

Once you are finished with the filter it's a good idea to remove it or at least turn it off so that it stops logging. Remember, since every request will produce an output, the log can grow quickly.

Probably the best thing to do is simply disable the filter. This way if you need it again in the future you can just turn it back on.

To disable the filter:

1. Change the `enabled` property to `false`.
2. (Optional) Move the filter back down to the bottom of the chain.

Logging response

class: `info.magnolia.debug.LoggingResponse`

This class wraps a response and records the set headers and http status for debugging purposes.

Performance test

class: `info.magnolia.debug.PerformanceTestFilter`

This filter logs whatever has been recorded with the `PerformanceTestStatus` class.

Deployment

To deploy the filter:

1. Download this XML: config.server.filters.performanceTestFilter.xml
2. Open the Configuration app and click the node: `/server/filters`
3. From the action bar, click "Import"

The filter will be imported at the bottom of the chain.

4. Drag (or move) the filter into position.
5. Enable the filter by changing the `enabled` property to `true`.

Usage

The point of this filter is to log the round trip time it takes to fulfil the requests. The time is logged to the standard output in milliseconds.

```
2019-06-13 09:10:23,978 INFO  info.magnolia.debug : /magnoliaPublic/travel.html : performanceTestFilter: 154
```

In this case, the filter was deployed as the second filter in the chain (i.e behind context). So this is approximately the total time to fulfill the request. In some cases, you might want to move this filter down in the chain. For example, right before the cms subchain will tell you how long it takes to render a specific request.

Removal

Once you are finished with the filter it's a good idea to remove it or at least turn it off so that it stops logging. Remember since every request will produce an output the log can grow quickly.

The best thing to do is to disable the filter. This way if you need it again in the future you can just turn it back on.

To disable the filter:

1. Change the `enabled` property to `false`.
2. (Optional) Move the filter back down to the bottom of the chain.

Session debugger

class: `info.magnolia.debug.SessionDebugger`

This is a filter and listener that can help to debug session issues.

Deployment

To deploy the filter:

1. Download this XML: config.server.filters.sessionDebugger.xml
2. Open the Configuration app and click the node: `/server/filters`
3. From the action bar, click "Import"

The filter will be imported at the bottom of the chain.

4. Drag (or move) the filter into position.
5. Enable the filter by changing the `enabled` property to `true`.

Usage

The point of this filter is to log session information. Configure it behind the context filter to see all session data in the log.

```
2019-06-14 14:10:00,272 WARN  info.magnolia.debug                                : -- Session found
-- Session attributes :
    com.vaadin.server.VaadinSession.AdminCentral = com.vaadin.server.VaadinSession@1ea60cc1
    Key[type=info.magnolia.personalization.trait.storage.StorageAwareTraitCollector$SessionScopedTraitStorage,
annotation=[none]] = info.magnolia.personalization.trait.storage.
StorageAwareTraitCollector$SessionScopedTraitStorage@1d4950db
    AdminCentral.lock = java.util.concurrent.locks.ReentrantLock@2f872e69[Unlocked]
    javax.security.auth.Subject = Subject:
        Principal: MgnlUser - superuser [51ae3379-67cf-4994-9e05-f97cb8bc3e4a]
        Principal: info.magnolia.cms.security.Realm$RealmImpl@179a1
        Principal: RoleListImpl[name=roles,list=[workflow-base, rest-admin, publisher, superuser]]
        Principal: GroupListImpl[name=groups,list=[publishers]]
        Principal: PrincipalCollectionImpl[name=PrincipalCollection]

-- Session is new : false
----------
```

Removal

Once you are finished with the filter it's a good idea to remove it or turn it off so that it stops logging. Remember, since every request will produce an output, the log can grow quickly.

> The best thing to do is to disable the filter. This way if you need it again in the future you can just turn it back on.

To disable the filter:

1. Change the `enabled` property to `false`.
2. (Optional) Move the filter back down to the bottom of the chain.