



AEM Rules for SonarQube



AEM Rules

for SonarQube

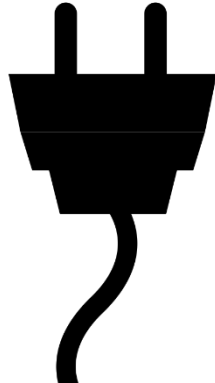
Michał Chudy



@michal_chudy

michal@chudy.co

sonarqube



code smells



Severity

❗ Blocker	0
🔴 Critical	2
🔴 Major	40
🟢 Minor	20
🟢 Info	166

Rule

🟢 Prefer cleaner @SlingServlet annotation.	18
🟢 Useless imports should be removed	2

»

```
15 import org.apache.sling.commons.json.JSONException;
16
17 @SlingServlet
18 @Properties({
19     @Property(name = "sling.servlet.extensions", value = { "json" }),
20     @Property(name = "sling.servlet.selectors", value = { "paragraphs" }),
21     @Property(name = "sling.servlet.resourceTypes", value = {
22         path/to/some/resource/type}},
23     @Property(name = "sling.servlet.methods", value = { HttpConstants.METHOD_GET }) })
24 public class SampleServlet extends SlingSafeMethodsServlet {
25
```

🟢 Property "sling.servlet.extensions" can be handled by @SlingServlet annotation.

[Comment](#) [Open](#) [Confirm](#) [Resolve](#) [False Positive](#) [Assign \[to me\]](#) [Plan](#) [Change Severity](#)

🟢 Property "sling.servlet.selectors" can be handled by @SlingServlet annotation.

[Comment](#) [Open](#) [Confirm](#) [Resolve](#) [False Positive](#) [Assign \[to me\]](#) [Plan](#) [Change Severity](#)

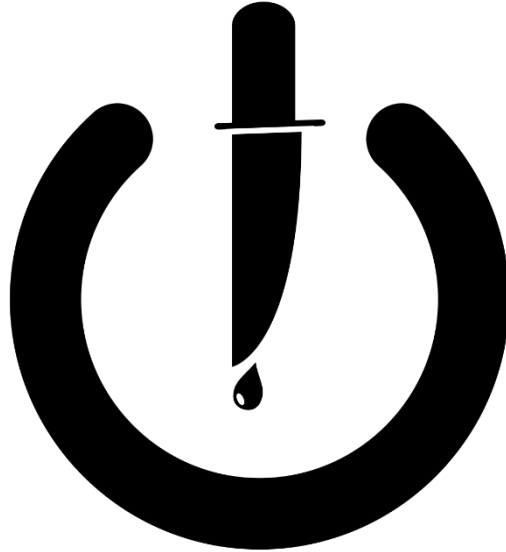
🟢 Property "sling.servlet.resourceTypes" can be handled by @SlingServlet annotation.

[Comment](#) [Open](#) [Confirm](#) [Resolve](#) [False Positive](#) [Assign \[to me\]](#) [Plan](#) [Change Severity](#)



**Session
should be
logged out.**

How many sessions it takes to kill AEM?



How many sessions it takes to kill AEM?

125024*



**Assuming we open
3 sessions per minute
on average. It would take
1 month
to take down CQ/AEM.**


Do

```
Session session = null;
try {
    session = repo.loginService(null, null);
    // do stuff
} catch (RepositoryException e) {
    LOGGER.error("Unable to create session", e);
} finally {
    if (session != null && session.isLive()) {
        session.logout();
    }
}
```



Don't

```
Session session = null;
try {
    session = repo.loginService(null, null);
    // do stuff
} catch (RepositoryException e) {
    LOGGER.error("Unable to create session", e);
}
```



**ResourceResolver
should be **closed** in
finally block.**

Do

```
ResourceResolver resourceResolver = null;
try {
    resourceResolver =
    factory.getServiceResourceResolver(null);
    // do stuff
} catch (LoginException e) {
    LOGGER.error("Unable to get service
                ResourceResolver.", e);
} finally {
    if (resourceResolver != null) {
        resourceResolver.close();
    }
}
```



Don't

```
ResourceResolver resourceResolver = null;
try {
    resourceResolver =
    factory.getServiceResourceResolver(null);
    // do stuff
} catch (LoginException e) {
    LOGGER.error("Unable to get service
                ResourceResolver.", e);
}
```



3 Injector should
be **closed** in
finally block.

Do

```
InjectorWithContext injector = null;
try {
    injector = InjectorUtil.getInjector(INJECTOR, resourceResolver)
    ModelProvider modelProvider = injector.getInstance(ModelProvider.class);
    // do stuff
} finally {
    if (injector != null) {
        injector.close();
    }
}
```



Don't

```
InjectorWithContext injector = injector = InjectorUtil.getInjector(INJECTOR, resourceResolver)
ModelProvider modelProvider = injector.getInstance(ModelProvider.class);
// do stuff
```



**Injector can be closed
using `try-with-resources`
Java 7 feature.**

Good

```
InjectorWithContext injector = null;
try {
    injector = InjectorUtil.getInjector(INJECTOR, resourceResolver)
    ModelProvider modelProvider = injector.getInstance(ModelProvider.class);
    // do stuff
} finally {
    if (injector != null) {
        injector.close();
    }
}
```

Better

```
try (InjectorWithContext injector = InjectorUtil.getInjector(INJECTOR, resourceResolver)) {
    ModelProvider modelProvider = injector.getInstance(ModelProvider.class);
    // do stuff
}
```




Sling / SLING-4798

ResourceResolver should extend java.io.Closeable

Comment

Agile Board

More ▾

Export ▾

Details

Type:	Improvement	Status:	RESOLVED
Priority:	Major	Resolution:	Duplicate
Affects Version/s:	Resource Resolver 1.2.4	Fix Version/s:	None
Component/s:	ResourceResolver		
Labels:	None		

Description

As discussed at [dev@sling](#) : Make ResourceResolver implement java.io.Closeable. This will allow its usage in the compatibility with older versions (java.io.Closeable was

Issue Links

⋮ duplicates

[SLING-4895](#) ResourceResolver should extend Closeable

People

Assignee:

Unassigned

Reporter:

Robert Munteanu

Votes:

0 Vote for this issue

Watchers:

2 Stop watching this issue

Dates

Created:

11/Jun/15 10:29

Updated:

13/Jun/15 17:31

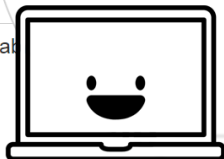
Resolved:

13/Jun/15 17:31

Updated:
13/Jun/15 17:31

Resolved:

13/Jun/15 17:31





5 Prefer **cleaner**
@SlingServlet
annotation.

Prefer...

```
@SlingServlet(  
    resourceTypes = "sling/servlet/default",  
    selectors = "selector",  
    extensions = "tab",  
    methods = HttpConstants.METHOD_GET  
)
```

Over...

```
@Component  
@Service(value = javax.servlet.Servlet.class)  
@Properties({ @Property(name = "sling.servlet.resourceTypes", value = { "sling/servlet/default" }),  
    @Property(name = "sling.servlet.selectors", value = { "selector" }),  
    @Property(name = "sling.servlet.extensions", value = { "tab" }),  
    @Property(name = "sling.servlet.methods", value = { HttpConstants.METHOD_GET }) })
```

Do

```
@SlingServlet(  
    resourceTypes = "sling/servlet/default", selectors = "selector",  
    extensions = "tab", methods = HttpConstants.METHOD_GET  
)  
@Properties({  
    @Property(name = Constants.SERVICE_VENDOR, value = "Cognifide"),  
    @Property(name = Constants.SERVICE_DESCRIPTION, value = "Some description")  
})
```

Don't

```
@SlingServlet(methods = "GET")  
@Properties({  
    @Property(name = Constants.SERVICE_VENDOR, value = "Cognifide"),  
    @Property(name = Constants.SERVICE_DESCRIPTION, value = "Some description"),  
    @Property(name = "sling.servlet.selectors", value = "selector"),  
    @Property(name = "sling.servlet.extensions", value = "tab"),  
    @Property(name = "sling.servlet.resourceTypes", value = { "sling/servlet/default" })  
})
```



Do

```
@SlingServlet(  
    resourceTypes = "sling/servlet/default", selectors = "selector",  
    extensions = "tab", methods = HttpConstants.METHOD_GET  
)
```

Don't

```
⊗ @Service  
⊗ @Component  
@SlingServlet(  
    resourceTypes = "sling/servlet/default", selectors = "selector",  
    extensions = "tab", methods = HttpConstants.METHOD_GET  
)
```




**Use predefined
constant instead of
hardcoded value.**

Do

```
private boolean isVisible(NavigationNode<Page> item) {  
    ValueMap valueMap = item.getData().getContentResource().adaptTo(ValueMap.class);  
    String template = valueMap.get(NameConstants.NN_TEMPLATE, String.class);  
    boolean hideInNav = BooleanUtils.toBoolean(valueMap.get(NameConstants.PN_HIDE_IN_NAV,  
        String.class));  
    return !hideInNav && template.equals(PAGE_TEMPLATE);  
}
```

Don't

```
private boolean isVisible(NavigationNode<Page> item) {  
    ValueMap valueMap = item.getData().getContentResource().adaptTo(ValueMap.class);  
    String template = valueMap.get("cq:template", String.class);  
    boolean hideInNav = BooleanUtils.toBoolean(valueMap.get("hideInNav", String.class));  
    return !hideInNav && template.equals(PAGE_TEMPLATE);  
}
```



7 Use predefined **constant** instead of hardcoded value in annotation.

Do

```
@Properties({
    @Property(name = EventConstants.EVENT_FILTER, value = "(path=/content/*)",
    @Property(name = EventConstants.EVENT_TOPIC, value = {
        SlingConstants.TOPIC_RESOURCE_ADDED,
        SlingConstants.TOPIC_RESOURCE_CHANGED,
        SlingConstants.TOPIC_RESOURCE_REMOVED
    })
})
```

Don't

```
@Properties({
    @Property(name = "event.filter", value = "(path=/content/*)",
    @Property(name = "event.topics", value = {
        "org/apache/sling/api/resource/Resource/ADDED",
        "org/apache/sling/api/resource/Resource/CHANGED",
        "org/apache/sling/api/resource/Resource/REMOVED"
    })
})
```

5 Reasons to use AEM constants

1. Consistency accross whole codebase.

2. Broader context of a value.

```
DamConstants.DC_TITLE = "dc:title"
```

3. Deprecation

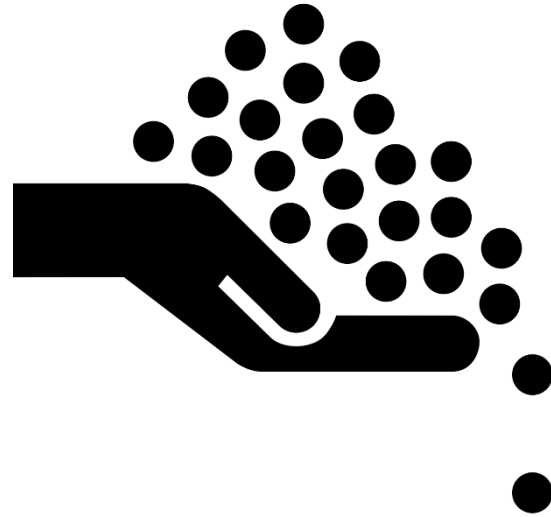
```
@Deprecated "filter.scope" VS "sling.filter.scope"
```

4. Extra documentation

```
SlingPostConstants.RD_NODE_NAME
```

- Optional request parameter specifying a node name for a newly created node (value is ":name").

5. Hey, someone made it a constant for a reason!



201 constants

159 @annotation constants



**Non-thread safe
object used as a
field of
Servlet/Filter etc.**

Don't

```
private Session session;
```

```
@Override
```

```
protected void doPost(SlingHttpServletRequest request, SlingHttpServletResponse response)
    throws ServletException, IOException {
    session = request.getResourceResolver().adaptTo(Session.class);
    String removeParam = request.getRequestParameter(REMOVE).getString();
    if (ALL.equals(removeParam)) {
        removeAllPagesCreatedByCurrentAuthor();
    } else if (OLD.equals(removeParam)) {
        removeOldPagesCreatedByCurrentAuthor();
    }
}
```

7 easy steps to start

1. Get SonarQube **4.5.4 or later**.
2. Make sure you have **Java 3.3** plugin installed.
3. Go to **github.com/Cognifide/AEM-Rules-for-SonarQube** and download plugin. (or **cognifide.github.io** :-)
4. Paste it into your **`sonarqube/extensions/plugins`** directory.
5. (Re)**start** your SonarQube instance!
6. Log in as admin and **activate** chosen rules in your default profile.
7. Go to your project directory and run
`clean install sonar:sonar`

What's coming next?

(don't read now :-)

Good practices:

1. GET Servlet should not modify repository.
2. Use findResources instead of QueryManager when not necessary.
3. ResourceResolver should be get using getServiceResourceResolver instead of getAdministrativeResourceResolver.
4. Sling Resource API should be used instead of JCR Node API
5. ResourceResolver can be closed using try-with-resources Java 7 feature.
6. Avoid mixture of data access API in the same method. (JCR, Sling, WCM)
7. Prefer cleaner @SlingFilter annotation.

JSP:

1. Is your JSP is compatible with trimWhiteSpaces AEM functionality?

What's coming next?

(don't read now :-)

Slice:

1. Use getListFromResources() instead of iteration.
2. @JcrProperty annotated fields should not be accessed in constructor.
3. @SliceResource annotated object should not keep any session based object.

Security:

1. Unsynchronized modification of a Service field in a method.
2. Creating ResourceResolver with administrative rights when request based is available.
3. Untrusted SlingServlet Parameter
4. Potential XSS in Servlet

Probable bugs:

1. Mutually exclusive settings in SlingServlet annotations.
2. GET Servlet should not modify repository.



AEM Rules for SonarQube



<http://cognifide.github.io/aem-rules-for-sonarqube/>

