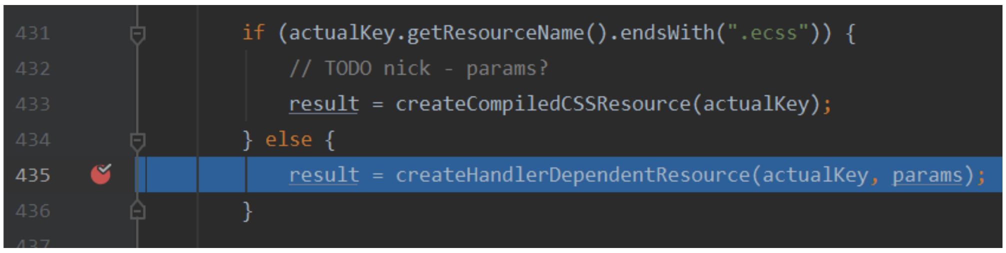
RichFaces's resource handler

As finding how to reach vulnerable entry point of CVE-2013-2165 and CVE-2015-0279, it reveals the way RichFaces processes requested resource.

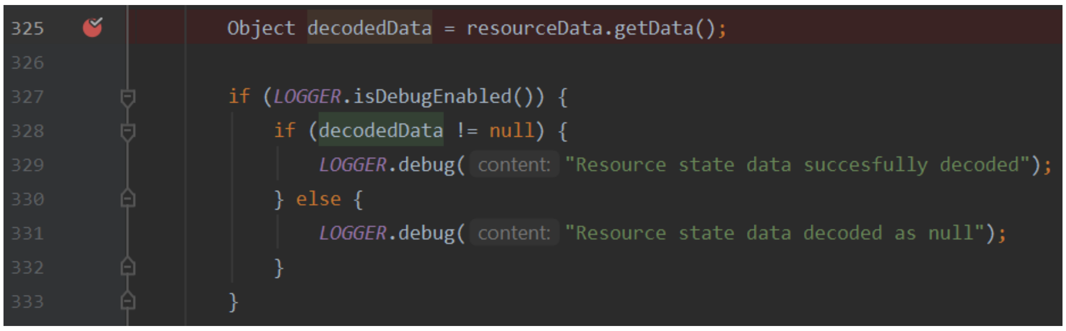
Every resource requests go into 'main resource handler' which is `core.jar::ResourceHandlerImpl`. This class performs this procedure:

1. Get the resource's class name that is used to handle. Two ways to provide:
   - `/rfRes/<resource class name>`. For example: `/rfRes/org.richfaces.resource.MediaOutputResource`
   - For static resource. `/rfRes/<resource name>?ln=<library name>`
2. Get serialized state of the resource via parameter `do`.
3. Create resource object with information in step 1 and restore its state by deserializing information in step 2.
   1. Create resource by `ResourceFactoryImpl` class. There are 2 types of resource
      1. Mapped resource, eg. images: `ResourceFactoryImpl#createResource()`
      2. Dynamic resources, eg. `ResourceFactoryImpl#createHandlerDependentResource()`:



```
            if (actualKey.getResourceName().endsWith(".ecss")) {
                // TODO nick - params?
                result = createCompiledCSSResource(actualKey);
            } else {
                result = createHandlerDependentResource(actualKey, params);
            }
```

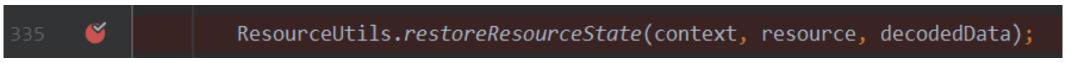
   2. Deserialise the serialized data by call resource's method `getData()`



```
        Object decodedData = resourceData.getData();

        if (LOGGER.isDebugEnabled()) {
            if (decodedData != null) {
                LOGGER.debug( content: "Resource state data succesfully decoded");
            } else {
                LOGGER.debug( content: "Resource state data decoded as null");
            }
        }
```

   3. Restore the resource's state with the deserialized object `decodedData`:



```
        ResourceUtils.restoreResourceState(context, resource, decodedData);
```

4. Process resource object:
   - Store cache
   - Produce corresponding response (images, videos, tables...):

```java
if (resource.userAgentNeedsUpdate(context)) {
    ...
    if (resource instanceof ContentProducerResource) {
        ContentProducerResource contentProducerResource =
(ContentProducerResource) resource;
        contentProducerResource.encode(context);
    } else {
        ...
    }
} else {
    sendNotModified(context);
}
```

The corresponding vulnerabilities:

- CVE-2013-2165: arbitrary deserialization in **step 2**
- CVE-2015-0279: EL injection in **step 4 -> Produce corresponding response -> ContentProduceResource#encode()**
- CVE-2018-12532: bypassing mitigation of CVE-2015-0279