Code Inspection

On the open source Glassfish 4.1 server

Assignment

- From package `org.glassfish.resources.module`
- Class `ResourcesDeployer.java`
- Five functions:
  - `processArchive(DeploymentContext dc)`
  - `retainResourceConfig(DeploymentContext dc, Map<String, Resources> allResources)`
  - `populateResourceConfigInAppInfo(DeploymentContext dc)`
  - `createResources(DeploymentContext dc, boolean embedded, boolean deployResources)`
  - `createConfig(Resources resources, Collection<org.glassfish.resources.api.Resource> resourcesToRegister, boolean embedded)`

Functional roles

- Handle `glassfish-resources.xml` files
- Loading and processing functionalities
- Such XML files come bundled with application
- This information was extracted from the attached Javadoc
  - And from reading the actual code

Code Issues

- All in all, examined code is compliant with all standards
- Only a handful of nitpicks, none of them really relevant
- Code is clear, very high-quality and well-written
  - At least for what concerns the checklist

Code Issues

- Checklist #10: Selected bracing style is Kerrigan & Ritchie
  - Choice otherwise consistently enforced for all snippets

Code Issues

- Checklist #14: Line length exceeds 120 characters on three lines
  - In two instances it's a method called with a fully qualified class name
    - `org.glassfish.resourcebase.resources.util.ResourceUtil.getActualModuleNameWithExtension(moduleName);`
    - `org.glassfish.resourcebase.resources.util.ResourceUtil.getActualModuleNameWithExtension(module.getName());`
  - The other is a method declaration with many verbose parameters
    - `public void createResources (DeploymentContext dc, boolean embedded, boolean deployResources) throws ResourceException`

Code Issues

- Checklist #18: Two public methods have no Javadoc
  - In methods `createResources()` and `createConfig()`
  - Other private methods have none either
  - While other public methods do
  - Unclear whether this was done on purpose

Code Issues

- Checklist #34: A single case of unused parameter
  - In method `createResources()`
  - Parameter `deployResources` is never used

Code Issues

- Checklist #41: A single case of spelling mistake
  - In method `processArchive()`
  - The thrown Exception string contains a minor mistake
    - `throw new DeploymentException ("Failue while processing glassfish-resources.xml (s) in the archive ", e)`

Code Issues

- Checklist #52: Two instances of general Exceptions caught instead of listing the specific ones
  - In methods `processArchive()` and `createConfig()`
  - Yet these are then bundled into DeploymentExceptions and rethrown
  - First instance is even commented
    - ```
      catch (Exception e) {
      //only Deployment Exceptions are propagated and result in
      deployment failure
      //in the event notification infrastructure
      throw new DeploymentException ("Failue while processing
      glassfish-resources.xml (s) in the archive ", e);
      ```
  - So they cannot really be considered mistakes
      ```

Other problems

- Deeply nested structures are sometimes used
  - In methods `createResources()` and `createConfg()`
  - Made of chains of (mainly) `if` and `try` constructs
  - They could be simplified…
    - Although it's nothing terribly complex

Conclusions

- Inspected code was clean and free of anything major
- We were actually hard-pressed to find anything to report
- The only *real* issue is the unused parameter case
- Everything else is either irrelevant or has a good reason to be like that