

Code Inspection

Filippo Agalbato
850481

Andrea Cannizzo
790469

December 31, 2015

Contents

1	Overview	2
1.1	Assignment	2
1.2	Functional roles	2
1.3	Code fragments proper	2
2	Code issues	8
2.1	Other problems	8
A	Document and work information	9
A.1	Revisions	9
A.2	Tools used	9
A.3	Overall time spent	9

Chapter 1

Overview

1.1 Assignment

The only class that was assigned is `ResourcesDeployer.java`, found in `appserver/-resources/resources-runtime/src/main/java/org/glassfish/resources/-module/`. From this class, the following methods were assigned for inspection:

- `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)`

1.2 Functional roles

—————todo————— Functional role of assigned set of classes: elaborate on the functional role you have identified for the class cluster that was assigned to you, also, elaborate on how you managed to understand this role and provide the necessary evidence, e.g., javadoc, diagrams, etc. —————

1.3 Code fragments proper

Follow here, for the sake of completeness, the actual code fragments that were assigned.

```
246     private void processArchive(DeploymentContext dc) {
        try {
            ReadableArchive archive = dc.getSource();
249             if (ResourceUtil.hasResourcesXML(archive, locator)) {
```

```

252      Map<String, Map<String, List>> appScopedResources =
          new HashMap<String, Map<String, List>>>();
      Map<String, String> fileNames = new HashMap<String,
          String>();

255      String appName = getAppNameFromDeployCmdParams(dc);
      //using appName as it is possible that "deploy --
          name=APPNAME" will
      //be different than the archive name.
258      retrieveAllResourcesXMLs(fileNames, archive,
          appName);

      for (Map.Entry<String, String> entry: fileNames.
          entrySet()) {
261          String moduleName = entry.getKey();
          String fileName = entry.getValue();
          debug("Sun_Resources_XML:_ " + fileName);

264          moduleName = org.glassfish.resourcebase.
              resources.util.ResourceUtil.
                  getActualModuleNameWithExtension(moduleName
                      );
          String scope ;
          if(appName.equals(moduleName)){
267              scope = JAVA_APP_SCOPE_PREFIX;
          }else{
270              scope = JAVA_MODULE_SCOPE_PREFIX;
          }

273          File file = new File(fileName);
          ResourcesXMLParser parser = new
              ResourcesXMLParser(file, scope);

276          validateResourcesXML(file, parser);

          List list = parser.getResourcesList();

279          Map<String, List> resourcesList = new HashMap<
              String, List>();
          List<org.glassfish.resources.api.Resource>
              nonConnectorResources =
282              ResourcesXMLParser.
                  getNonConnectorResourcesList(list,
                      false, true);
          resourcesList.put(NON_CONNECTOR_RESOURCES,
              nonConnectorResources);

285          List<org.glassfish.resources.api.Resource>
              connectorResources =
              ResourcesXMLParser.
                  getConnectorResourcesList(list,
                      false, true);
          resourcesList.put(CONNECTOR_RESOURCES,
              connectorResources);

288          appScopedResources.put(moduleName,
              resourcesList);
      }
291      dc.addTransientAppMetaData(APP_SCOPED_RESOURCES_MAP
          , appScopedResources);
      ApplicationInfo appInfo = appRegistry.get(appName);

```

```

        if(appInfo != null){
            Application app = dc.getTransientAppMetaData(
                ServerTags.APPLICATION, Application.class);
            appInfo.addTransientAppMetaData(ServerTags.
                APPLICATION, app);
        }
    } catch (Exception e) {
        // only DeploymentExceptions 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);
    }
}

/**
 * retain old resource configuration for the new archive being
 * deployed.
 * @param dc DeploymentContext
 * @param allResources all resources (app scoped, module scoped
 * ) of old application
 * @throws Exception when unable to retain old resource
 * configuration.
 */
public void retainResourceConfig(DeploymentContext dc, Map<
    String, Resources> allResources) throws Exception {
    String appName = getAppNameFromDeployCmdParams(dc);
    Application application = dc.getTransientAppMetaData(
        ServerTags.APPLICATION, Application.class);
    Resources appScopedResources = allResources.get(appName);

    if(appScopedResources != null){
        application.setResources(appScopedResources);
    }

    if(DeploymentUtils.isArchiveOfType(dc.getSource(), DOLUtils
        .earType(), locator)){
        List<Module> modules = application.getModule();
        if(modules != null){
            for(Module module : modules){
                Resources moduleScopedResources = allResources.
                    get(module.getName());
                if(moduleScopedResources != null){
                    module.setResources(moduleScopedResources);
                }
            }
        }
    }
}

/**
 * During "load()" event (eg: app/app-ref enable, server start)
 * ,
 * populate resource-config in app-info so that it can be used
 * for
 * constructing connector-classloader for the application.
 * @param dc DeploymentContext
 */
public void populateResourceConfigInAppInfo(DeploymentContext
    dc){
    String appName = getAppNameFromDeployCmdParams(dc);

```

```

        Application application = applications.getApplication(
            appName);
        ApplicationInfo appInfo = appRegistry.get(appName);
351     if(application != null && appInfo != null){
            Resources appScopedResources = application.getResources
                ();
            if(appScopedResources != null){
354                 appInfo.addTransientAppMetaData(ServerTags.
                    APPLICATION, application);
                appInfo.addTransientAppMetaData(application.getName
                    ()+"-resources", appScopedResources);
            }
357
            List<Module> modules = application.getModule();
            if(modules != null){
360                 for(Module module : modules){
                    Resources moduleScopedResources = module.
                        getResources();
                    if(moduleScopedResources != null){
363                        appInfo.addTransientAppMetaData(module.
                            getName()+"-resources",
                                moduleScopedResources);
                    }
                }
366            }
        }
    }

    public void createResources(DeploymentContext dc, boolean
        embedded, boolean deployResources) throws ResourceException
    {
        String appName = getAppNameFromDeployCmdParams(dc);
        Application app = dc.getTransientAppMetaData(ServerTags.
372            APPLICATION, Application.class);
        Map<String, Map<String, List>> resourcesList =
            (Map<String, Map<String, List>>)dc.
                getTransientAppMetadata().get(
                    APP_SCOPED_RESOURCES_MAP);
375
        if (resourcesList != null) {
            Map<String, List> appLevelResources = resourcesList.get
                (appName);
            if (appLevelResources != null) {
378                List<org.glassfish.resources.api.Resource>
                    connectorResources =
                        appLevelResources.get(CONNECTOR_RESOURCES);
381
                createAppScopedResources(app, connectorResources,
                    dc, embedded);

                List<org.glassfish.resources.api.Resource>
384                    nonConnectorResources =
                        appLevelResources.get(
                            NON_CONNECTOR_RESOURCES);

                createAppScopedResources(app, nonConnectorResources
387                    , dc, embedded);
            }
390            List<Module> modules = app.getModule();
            if (modules != null) {
                for (Module module : modules) {
393                    String actualModuleName = org.glassfish.

```

```

        resourcebase.resources.util.ResourceUtil.
            getActualModuleNameWithExtension(module.
                getName());
        //create resources for modules, ignore
        //standalone applications where
        //module name will be the same as app name
396     if(!appName.equals(actualModuleName)){
        Map<String, List> moduleResources =
            resourcesList.get(actualModuleName);
        if (moduleResources != null) {
399             List<org.glassfish.resources.api.
                Resource> connectorResources =
                    moduleResources.get(
                        CONNECTOR_RESOURCES);
            createModuleScopedResources(app, module
                , connectorResources, dc, embedded)
                ;

402             List<org.glassfish.resources.api.
                Resource> nonConnectorResources =
                    moduleResources.get(
                        NON_CONNECTOR_RESOURCES);
405             createModuleScopedResources(app, module
                , nonConnectorResources, dc,
                    embedded);
        }
    }
408 }
    }
411 }

private Collection<Resource>
414 createConfig(Resources resources, Collection<org.glassfish.
    resources.api.Resource> resourcesToRegister,
        boolean embedded)
throws ResourceException {
417     List<Resource> resourceConfigs =
        new ArrayList<Resource>();
    for (org.glassfish.resources.api.Resource resource :
        resourcesToRegister) {
420         final HashMap attrList = resource.getAttributes();
        final Properties props = resource.getProperties();
        String desc = resource.getDescription();
423         if (desc != null) {
            attrList.put("description", desc);
        }

426         try {
            final ResourceManager rm = resourceFactory.
                getResourceManager(resource);
            if(embedded && isEmbeddedResource(resource,
429                 resourcesToRegister)){
                Resource configBeanResource =
                    rm.createConfigBean(resources, attrList
                        , props, false);
432                 resources.getResources().add(configBeanResource
                    );
                resourceConfigs.add(configBeanResource);
            }else if(!embedded && !isEmbeddedResource(resource,
                resourcesToRegister)){
435                 com.sun.enterprise.config.serverbeans.Resource
                    configBeanResource =

```

```

        rm.createConfigBean(resources, attrList
            , props, true);
        resources.getResources().add(configBeanResource
            );
438         resourceConfigs.add(configBeanResource);
        }
    } catch (Exception e) {
441         throw new ResourceException(e);
    }
}
444 return resourceConfigs;
}
```


Chapter 2

Code issues

List of issues found by applying the checklist: report the classes/code fragments that do not fulfill some points in the check list. Explain which point is not fulfilled and why.

2.1 Other problems

Any other problem you have highlighted: list here all the parts of code that you think create or may create a bug and explain why.

Appendix A

Document and work information

A.1 Revisions

This is the first version of this document. There are currently no revisions.

A.2 Tools used

TeXworks editor With PDF^ATeX, for composing and editing this document.

A.3 Overall time spent

The authors spent about 10 hours of their time, equally divided among them, working on this document.