Code Inspection

Filippo Agalbato
850481

Andrea Cannizzo
790469

January 3, 2016

Contents

Chapter 1

Overview

1.1 Assignment

The only class that was assigned is `ResourcesDeployer.java`, from package `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

As the provided Javadoc for the class makes very clear, it is intended to handle `glassfish-resources.xml` files bundled in the application, loading and processing them. The assigned methods quite trivially enforce this declaration of purpose.

1.3 Code fragments

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

```
private void processArchive(DeploymentContext dc) {

    try {
        ReadableArchive archive = dc.getSource();

        if (ResourceUtil.hasResourcesXML(archive, locator)) {
```

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

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

        for (Map.Entry<String, String> entry: fileNames.
            entrySet()) {
            String moduleName = entry.getKey();
            String fileName = entry.getValue();
            debug("Sun␣Resources␣XML␣:␣" + fileName);

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

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

            validateResourcesXML(file, parser);

            List list = parser.getResourcesList();

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

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

            appScopedResources.put(moduleName,
                resourcesList);
        }
        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);
        if(application != null && appInfo != null){
            Resources appScopedResources = application.getResources
                ();
            if(appScopedResources != null){
                appInfo.addTransientAppMetaData(ServerTags.
                    APPLICATION, application);
                appInfo.addTransientAppMetaData(application.getName
                    ()+"-resources", appScopedResources);
            }

            List<Module> modules = application.getModule();
            if(modules != null){
                for(Module module : modules){
                    Resources moduleScopedResources = module.
                        getResources();
                    if(moduleScopedResources != null){
                        appInfo.addTransientAppMetaData(module.
                            getName()+"-resources",
                            moduleScopedResources);
                    }
                }
            }
        }
    }

    public void createResources(DeploymentContext dc, boolean
        embedded, boolean deployResources) throws ResourceException
         {
        String appName = getAppNameFromDeployCmdParams(dc);
        Application app = dc.getTransientAppMetaData(ServerTags.
            APPLICATION, Application.class);
        Map<String, Map<String, List>> resourcesList =
                (Map<String, Map<String, List>>)dc.
                    getTransientAppMetadata().get(
                    APP_SCOPED_RESOURCES_MAP);

        if (resourcesList != null) {
            Map<String, List> appLevelResources = resourcesList.get
                (appName);
            if (appLevelResources != null) {
                List<org.glassfish.resources.api.Resource>
                    connectorResources =
                        appLevelResources.get(CONNECTOR_RESOURCES);

                createAppScopedResources(app, connectorResources,
                    dc, embedded);

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

                createAppScopedResources(app, nonConnectorResources
                    , dc, embedded);

            }
            List<Module> modules = app.getModule();
            if (modules != null) {
                for (Module module : modules) {
                    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
                            if(!appName.equals(actualModuleName)){
                                Map<String, List> moduleResources =
                                    resourcesList.get(actualModuleName);
                                if (moduleResources != null) {
                                    List<org.glassfish.resources.api.
                                        Resource> connectorResources =
                                            moduleResources.get(
                                                CONNECTOR_RESOURCES);
                                    createModuleScopedResources(app, module
                                        , connectorResources, dc, embedded)
                                        ;

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

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

                    try {
                        final ResourceManager rm = resourceFactory.
                            getResourceManager(resource);
                        if(embedded && isEmbeddedResource(resource,
                            resourcesToRegister)){
                            Resource configBeanResource =
                                    rm.createConfigBean(resources, attrList
                                        , props, false);
                            resources.getResources().add(configBeanResource
                                );
                            resourceConfigs.add(configBeanResource);
                        }else if(!embedded && !isEmbeddedResource(resource,
                             resourcesToRegister)){
                            com.sun.enterprise.config.serverbeans.Resource
                                configBeanResource =
                                    rm.createConfigBean(resources, attrList
                                        , props, true);
                            resources.getResources().add(configBeanResource
                                );
                            resourceConfigs.add(configBeanResource);
                        }
                    } catch (Exception e) {
                        throw new ResourceException(e);
                    }
                }
                return resourceConfigs;
            }
```

Chapter 2

Code issues

This chapter makes explicit reference to both the provided checklist and the code fragments (included in section 1.3).

2.1 Main issues

- Item 14 of the checklist is violated on line 265, method `processArchive()`: line length exceeds 120 characters;
- Item 41 of the checklist is violated on line 301, method `processArchive()`: there is a spelling error in the exception string;

2.2 Other problems

No other problems have been found in the assigned fragments.

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 PDFLaTeX, for composing and editing this document.

A.3 Overall time spent

The authors spent about 8 hours of their time, equally divided among them, working on this document.