Red Hat JBoss Developer Studio 11.2

Release Notes and Known Issues

Highlighted features in 11.2

Last Updated: 2018-01-31

Red Hat JBoss Developer Studio 11.2 Release Notes and Known Issues

Highlighted features in 11.2

Misha Husnain Ali
mhusnain@redhat.com

Supriya Takkhi
sbharadw@redhat.com

Legal Notice

Abstract

This document lists and briefly describes new and improved features of Red Hat JBoss Developer Studio 11.2.

Table of Contents

CHAPTER 1. INTRODUCTION TO RED HAT JBOSS DEVELOPER STUDIO

1.1. ABOUT RED HAT JBOSS DEVELOPER STUDIO

JBoss Developer Studio is a set of Eclipse-based development tools. It contains plug-ins that integrate with Eclipse to extend the existing functionality of the integrated development environment (IDE).

JBoss Developer Studio is designed to increase your productivity when developing applications. You can focus on building, testing, and deploying your applications because JBoss application development tools are integrated in one IDE. JBoss Developer Studio can also assist your application development with its unique features in the following ways:

- Develop new applications using the wizards and project examples of Red Hat Central
- Add powerful functionality to applications with minimal effort using Forge Tools
- Build web interfaces with ease using the visual editing and drag-and-drop utilities of Visual Web Tools and Mobile Web Tools
- Experience browsers automatically refreshing in response to modified application resources with LiveReload Tools
- Incorporate Hibernate, CDI, JAX-RS, JSF, Seam, and other popular APIs into applications with simplicity using the tool-driven interface
- Preview and test mobile web applications on a variety of simulated mobile devices using BrowserSim
- Create, build, and test Cordova-based hybrid mobile applications for iOS and Android platforms using Hybrid Mobile Tools and CordovaSim
- Deploy applications to JBoss runtime servers and the cloud using JBoss Server Tools and OpenShift Tools

JBoss Developer Studio is built around Eclipse and packaged with all the necessary dependencies and third-party plug-ins for simplified installing. For developers already running Eclipse,
JBoss Developer Studio can also be installed through Eclipse Marketplace. Installing
JBoss Developer Studio in an existing Eclipse installation is referred to as BYOE (Bring Your Own Eclipse).

1.2. USE CASES OF JBOSS DEVELOPER STUDIO

JBoss Developer Studio assists Java EE developers by integrating JBoss technology and APIs in a single development environment. Here are a few ways that JBoss Developer Studio helps make development easier:

1.2.1. Web Applications

Red Hat Central provides wizards that generate skeletons and sample projects, enabling you to focus on developing the functionality of your applications. The wizards create web applications based on different APIs and technologies, showing the usage and advantages of each. JBoss Developer Studio also offers project file templates in a range of popular programming languages, including HTML, XHTML, and JSF.

Palettes in JBoss Developer Studio give access to the core elements of the JSF, RichFaces and Seam APIs, for use in developing the user interfaces of your applications. Elements of these APIs can be dragged and dropped directly into your project so that you can create richer user interfaces quickly. Visual Web Tools offers graphical and source viewing of files and defaults to dedicated editors for different file types. JBoss Developer Studio supports the Java EE specification and provides tools for JAX-RS, Hibernate, and CDI APIs so you can develop the server-side components of your applications effortlessly.

LiveReload Tools automatically refreshes browsers of local or deployed applications as you modify project resources to avoid needing to manually refresh. You can experience automatic refreshing when viewing applications in browsers on external and mobile devices, with application web addresses easy to navigate to with QR codes.

1.2.2. Web Applications Optimized for Mobile Devices

Mobile Web Tools provides support for HTML5 and jQuery Mobile to enable you to create web applications optimized across desktop and mobile clients. The HTML5 Project wizard in Red Hat Central generates a sample application using HTML5 and jQuery Mobile technologies and, together with HTML5 and jQuery Mobile project file templates, helps you to get up and running with these APIs and technologies quickly. HTML5 and jQuery Mobile widgets can be dragged from the jQuery Mobile palette into your project files and, in conjunction with the widget wizards, enable you to effortlessly develop customized user interfaces for your mobile web applications.

BrowserSim allows you to view your web applications on a variety of simulated mobile devices so that you can ensure they will be correctly formatted. LiveReload also extends to BrowserSim allowing you to experience automatic browser refreshing as you develop your mobile web applications. The integration of Firebug Lite and Weinre capabilities with BrowserSim assists you to inspect the page source of web pages with familiar tools.

1.2.3. Hybrid Mobile Applications

Hybrid Mobile Tools provides support for developing and building Cordova-based hybrid mobile applications for iOS and Android platforms. The Hybrid Mobile application wizard assists you to quickly generate new projects, while the Cordova Configuration Editor and Cordova Plug-in Discovery wizard help you to efficiently modify the capabilities of your projects. Hybrid Mobile Tools provides actions that simplify your workflow, for example calling your system installed Android and iOS SDKs from within the IDE to emulate or run your hybrid mobile applications. With wizards to export workspace projects to Cordova-enabled native projects or ready-to-sign applications, you can quickly be ready to share your hybrid mobile projects and applications.

CordovaSim enables you to view and test your hybrid mobile applications on Android and iOS simulated mobile devices so that you can ensure they look and work as expected. You can interact with your mobile applications through BrowserSim and use the device input panel to provide sample data to your applications for device functions like cameras and accelerometers. An advantage of CordovaSim is that it does not require native SDKs to be installed on your system, unlike native SDK emulators. Additionally, by teaming the device control panel with BrowserSim, you get all of the great functionality of BrowserSim, such as skins and LiveReload, while simulating your hybrid mobile applications.

1.2.4. Applications for Cloud Deployment

OpenShift Tools deploys your applications directly to the cloud on the Red Hat OpenShift platform. You can create and manage your OpenShift account and manage the deployment of applications to OpenShift within the IDE. In addition to using the OpenShift Application wizard to create and deploy new OpenShift applications, OpenShift Tools can import applications already deployed on OpenShift so that you can further develop them and manage their deployment from the comfort of the IDE.

CHAPTER 2. ABOUT THIS RELEASE

Red Hat JBoss Developer Studio 11.2 is an update of Red Hat JBoss Developer Studio 11.1 and it has the following features:

- It includes Eclipse Oxygen.
- It requires a minimum of Java 8 to run.
- It introduces new features, which are outlined in the New Features section.
- It contains new features for the existing tools.
- It resolves issues identified in earlier versions of JBoss Developer Studio.

For more information about operating systems, chip architectures and Java developer kits supported by this release, see Supported Configurations and Components page on the Red Hat Customer Portal.

CHAPTER 3. ECLIPSE AUTOMATED REPORTING INTERFACE (AERI)

To contribute to JBoss Tools, we recommend you to enable the Eclipse Automated Reporting Interface (AERI) in JBoss Tools. To read about configuring error reporting in JBoss Tools, see: http://tools.jboss.org/usage/#error-reporting.

CHAPTER 4. NEW FEATURES AND ENHANCEMENTS

4.1. OPENSHIFT 3

4.1.1. Enhanced Command to Delete Resource(s)

Earlier, you could delete OpenShift resources in one of the following two ways:

- Individually delete each resource. However, this would create a problem because some resources are hidden by the OpenShift Explorer.
- Delete the containing OpenShift project. But, in this case you may delete more resources than required.

A new enhanced command to delete resources is available at the OpenShift project level. It lists all the available OpenShift resources for the selected OpenShift project. You can then select the resources that you want to delete. You can also filter the list using a filter that is applied to the labels for each retrieved OpenShift resource.

For example, if you have two different deployments in a single OpenShift project (if you arev using OpenShift Online Starter) or if you have different kind of resources in a single deployment, you can now distinct them.

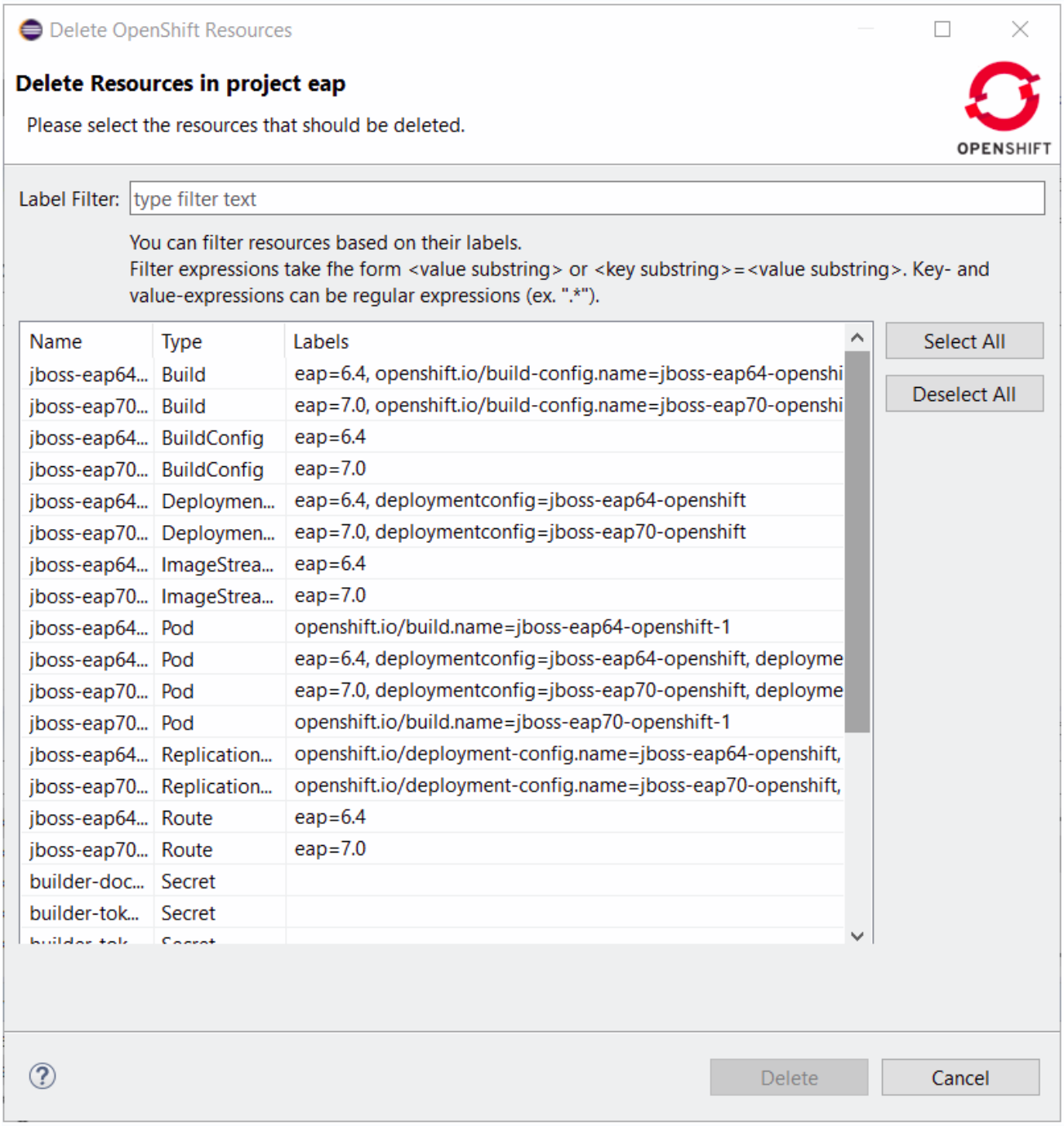
The following images show the workflow of the preceding example.

In this example, we have deployed an EAP6.4 based application and an EAP7.0 based application. The following image shows the **OpenShift Explorer** view.

Figure 4.1. Application as Shown in the OpenShift Explorer

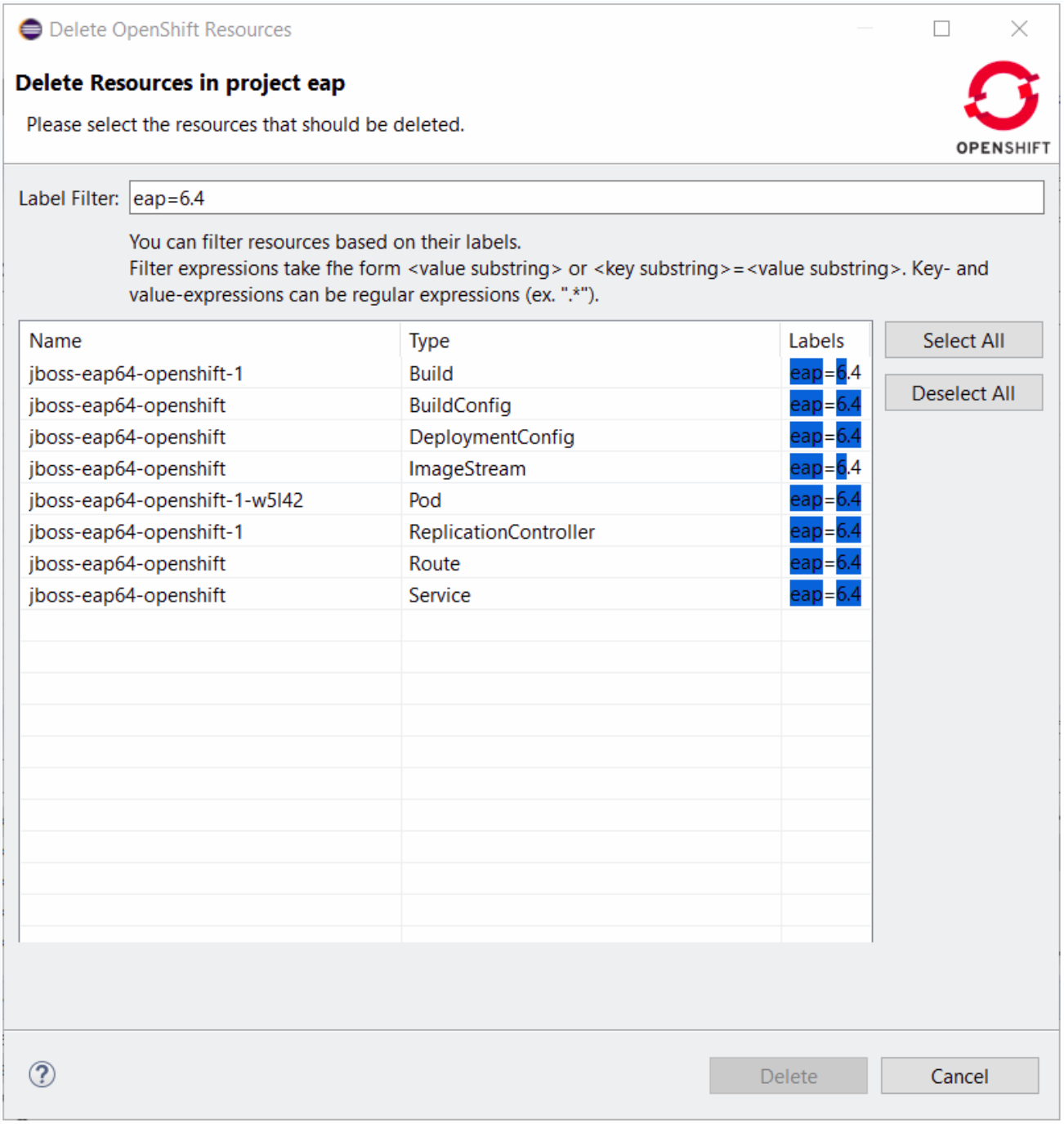
To invoke the new delete command on the EAP OpenShift project, right-click the OpenShift project and click **Delete Resources**.

Figure 4.2. Deleting a Resource in an OpenShift Project



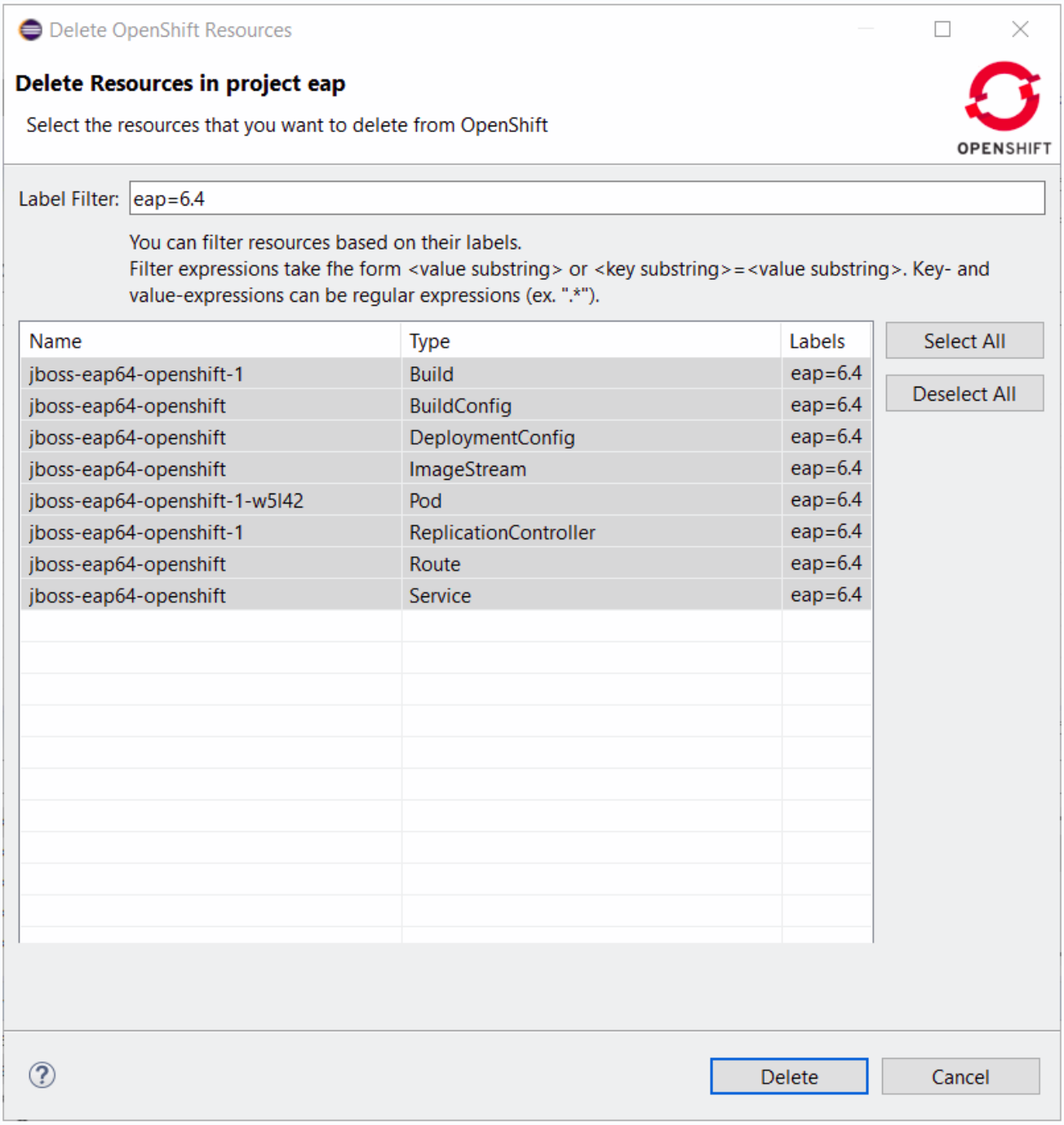
For this example, we will consider deleting the EAP6.4 deployement. In the **Label filter** field, type *eap=6.4*.

Figure 4.3. Selecting a Deployment for Deletion



Click **`Select All`**.

Figure 4.4. Clicking the Select All Button



Click **OK** to close the **Delete Resources in Project *{project name}*** window. The resources are deleted and the **OpenShift Explorer** view is updated accordingly.

Figure 4.5. OpenShift Explorer View Updated after Resource Deletion

Related JIRA: JBIDE-25111

4.1.2. Support for Route Timeouts and Liveness Probe for OpenShift Server Adapter Debugging Configurations

While debugging your OpenShift deployment, you may face two different issues:

- If you launch your test through a Web browser, then it's likely that you will access your OpenShift deployment through an OpenShift route. The problem is that, by default, OpenShift routes have a 30 seconds timeout for each request. So if you are stepping through one of your breakpoints, you will get a timeout error message in the browser window even if you can still debug your OpenShift deployment. And you will be unable to navigate through your OpenShift application.

- If your OpenShift deployment has a liveness probe configured, depending on your virtual machine capabilities or how your debugger is configured, if you are stepping into one of your breakpoints, the liveness probe may fail; so, OpenShift will restart your container and your debugging session will be destroyed.

So, now, when the OpenShift server adapter is started in debug mode, the following action will be performed:

- If an OpenShift route is found that is linked to the OpenShift deployment you want to debug, the route timeout will be set or increased to one hour. The original or default value will be restored when the OpenShift server adapter will be restarted in run mode.

- If your OpenShift deployment has a liveness probe configured, the **initialDelay** field will be increased to one hour if the defined value for this field is lower than one hour. If the value of this field is defined to a value value that is greater than an hour, it is left intact. The original value will be restored when the OpenShift server adapter will be restarted in run mode

Related JIRA: JBIDE-24868

4.1.3. Spring Boot applications support in OpenShift server adapter

The OpenShift server adapter allowed hotdeploy and debugging for JEE and NodeJS based applications. It now supports Spring Boot applications with the following limitations: the Spring Boot devtools module must be added to your application as it monitors code changes and as the application must be launched in exploded mode, you must use the upstream image (docker.io/fabric8/s2i-java) rather than the downstream image builder fis-java-openshift.

As an example, we have provided the following OpenShift template that will create an OpenShift application based on the upstream application and a Git repository that added the Spring Boot devtools to the Fabric8 Spring Boot quickstart.

```
{
 +  "apiVersion": "v1",
 +  "kind": "Template",
 +  "metadata": {
 +    "annotations": {
 +      "description": "Spring-Boot and CXF JAXRS QuickStart. This example
demonstrates how you can use Apache CXF JAXRS with Spring Boot on
Openshift. The quickstart uses Spring Boot to configure a little
application that includes a CXF JAXRS endpoint with Swagger enabled.",
 +      "tags": "quickstart,java,springboot,fis",
 +      "iconClass": "icon-jboss",
 +      "version": "2.0"
 +    },
 +    "name": "s2i-spring-boot-cxf-jaxrs"
 +  },
 +  "labels": {
 +    "template": "s2i-spring-boot-cxf-jaxrs"
 +  },
 +  "parameters": [
```

```
 +    {
 +      "name": "APP_NAME",
 +      "displayName": "Application Name",
 +      "required": true,
 +      "value": "s2i-spring-boot-cxf-jaxrs",
 +      "description": "The name assigned to the application."
 +    },
 +    {
 +      "name": "GIT_REPO",
 +      "displayName": "Git Repository URL",
 +      "required": true,
 +      "value": "https://github.com/jeffmaury/spring-boot-cxf-jaxrs.git",
 +      "description": "The URL of the repository with your application
source code."
 +    },
 +    {
 +      "name": "GIT_REF",
 +      "displayName": "Git Reference",
 +      "value": "hotdeploy",
 +      "description": "Set this to a branch name, tag or other ref of
your repository if you are not using the default branch."
 +    },
 +    {
 +      "name": "SERVICE_NAME",
 +      "displayName": "Service Name",
 +      "value": "cxf-jaxrs",
 +      "description": "Exposed service name."
 +    },
 +    {
 +      "name": "BUILDER_VERSION",
 +      "displayName": "Builder version",
 +      "value": "2.0",
 +      "description": "The version of the FIS S2I builder image to use."
 +    },
 +    {
 +      "name": "APP_VERSION",
 +      "displayName": "Application Version",
 +      "value": "1.0.0.redhat-000014",
 +      "description": "The application version."
 +    },
 +    {
 +      "name": "MAVEN_ARGS",
 +      "displayName": "Maven Arguments",
 +      "value": "package -DskipTests -Dfabric8.skip -e -B",
 +      "description": "Arguments passed to mvn in the build."
 +    },
 +    {
 +      "name": "MAVEN_ARGS_APPEND",
 +      "displayName": "Extra Maven Arguments",
 +      "description": "Extra arguments passed to mvn, e.g. for multi-
module builds."
 +    },
 +    {
 +      "name": "ARTIFACT_DIR",
 +      "displayName": "Maven build directory",
 +      "description": "Directory of the artifact to be built, e.g. for
```

```
multi-module builds."
 +    },
 +    {
 +      "name": "IMAGE_STREAM_NAMESPACE",
 +      "displayName": "Image Stream Namespace",
 +      "value": "openshift",
 +      "required": true,
 +      "description": "Namespace in which the Fuse ImageStreams are
installed. These ImageStreams are normally installed in the openshift
namespace. You should only need to modify this if you've installed the
ImageStreams in a different namespace/project."
 +    },
 +    {
 +      "name": "BUILD_SECRET",
 +      "displayName": "Git Build Secret",
 +      "generate": "expression",
 +      "description": "The secret needed to trigger a build.",
 +      "from": "[a-zA-Z0-9]{40}"
 +    },
 +    {
 +      "name": "CPU_REQUEST",
 +      "displayName": "CPU request",
 +      "value": "0.2",
 +      "required": true,
 +      "description": "The amount of CPU to requests."
 +    },
 +    {
 +      "name": "CPU_LIMIT",
 +      "displayName": "CPU limit",
 +      "value": "1.0",
 +      "required": true,
 +      "description": "The amount of CPU the container is limited to
use."
 +    }
 +  ],
 +  "objects": [
 +    {
 +      "apiVersion": "v1",
 +      "kind": "Route",
 +      "metadata": {
 +        "labels": {
 +          "component": "${APP_NAME}",
 +          "provider": "s2i",
 +          "project": "${APP_NAME}",
 +          "version": "${APP_VERSION}",
 +          "group": "quickstarts"
 +        },
 +        "name": "${SERVICE_NAME}-route"
 +      },
 +      "spec": {
 +        "to": {
 +          "kind": "Service",
 +          "name": "${SERVICE_NAME}"
 +        }
 +      }
 +    },
```

```
+    {
+      "apiVersion": "v1",
+      "kind": "Service",
+      "metadata": {
+        "annotations": {
+        },
+        "labels": {
+          "component": "${APP_NAME}",
+          "provider": "s2i",
+          "project": "${APP_NAME}",
+          "version": "${APP_VERSION}",
+          "group": "quickstarts"
+        },
+        "name": "${SERVICE_NAME}"
+      },
+      "spec": {
+        "clusterIP": "None",
+        "deprecatedPublicIPs": [],
+        "ports": [
+          {
+            "port": 9413,
+            "protocol": "TCP",
+            "targetPort": 8080
+          }
+        ],
+        "selector": {
+          "project": "${APP_NAME}",
+          "component": "${APP_NAME}",
+          "provider": "s2i",
+          "group": "quickstarts"
+        }
+      }
+    },
+    {
+      "kind": "ImageStream",
+      "apiVersion": "v1",
+      "metadata": {
+        "name": "${APP_NAME}",
+        "creationTimestamp": null,
+        "labels": {
+          "component": "${APP_NAME}",
+          "group": "quickstarts",
+          "project": "${APP_NAME}",
+          "provider": "s2i",
+          "version": "${APP_VERSION}"
+        }
+      },
+      "spec": {},
+      "status": {
+        "dockerImageRepository": ""
+      }
+    },
+    {
+      "kind": "BuildConfig",
+      "apiVersion": "v1",
+      "metadata": {
```

```
+        "name": "${APP_NAME}",
+        "creationTimestamp": null,
+        "labels": {
+          "component": "${APP_NAME}",
+          "group": "quickstarts",
+          "project": "${APP_NAME}",
+          "provider": "s2i",
+          "version": "${APP_VERSION}"
+        }
+      },
+      "spec": {
+        "triggers": [
+          {
+            "type": "GitHub",
+            "github": {
+              "secret": "${BUILD_SECRET}"
+            }
+          },
+          {
+            "type": "Generic",
+            "generic": {
+              "secret": "${BUILD_SECRET}"
+            }
+          },
+          {
+            "type": "ConfigChange"
+          },
+          {
+            "type": "ImageChange",
+            "imageChange": {}
+          }
+        ],
+        "source": {
+          "type": "Git",
+          "git": {
+            "uri": "${GIT_REPO}",
+            "ref": "${GIT_REF}"
+          }
+        },
+        "strategy": {
+          "type": "Source",
+          "sourceStrategy": {
+            "from": {
+              "kind": "DockerImage",
+              "name": "fabric8/s2i-java:${BUILDER_VERSION}"
+            },
+            "forcePull": true,
+            "incremental": true,
+            "env": [
+              {
+                "name": "BUILD_LOGLEVEL",
+                "value": "5"
+              },
+              {
+                "name": "ARTIFACT_DIR",
+                "value": "${ARTIFACT_DIR}"
```

```
+              },
+              {
+                "name": "MAVEN_ARGS",
+                "value": "${MAVEN_ARGS}"
+              },
+              {
+                "name": "MAVEN_ARGS_APPEND",
+                "value": "${MAVEN_ARGS_APPEND}"
+              }
+            ]
+          }
+        },
+        "output": {
+          "to": {
+            "kind": "ImageStreamTag",
+            "name": "${APP_NAME}:latest"
+          }
+        },
+        "resources": {}
+      },
+      "status": {
+        "lastVersion": 0
+      }
+    },
+    {
+      "kind": "DeploymentConfig",
+      "apiVersion": "v1",
+      "metadata": {
+        "name": "${APP_NAME}",
+        "creationTimestamp": null,
+        "labels": {
+          "component": "${APP_NAME}",
+          "group": "quickstarts",
+          "project": "${APP_NAME}",
+          "provider": "s2i",
+          "version": "${APP_VERSION}"
+        }
+      },
+      "spec": {
+        "strategy": {
+          "resources": {}
+        },
+        "triggers": [
+          {
+            "type": "ConfigChange"
+          },
+          {
+            "type": "ImageChange",
+            "imageChangeParams": {
+              "automatic": true,
+              "containerNames": [
+                "${APP_NAME}"
+              ],
+              "from": {
+                "kind": "ImageStreamTag",
+                "name": "${APP_NAME}:latest"
```

```
+              }
+            }
+          }
+        ],
+        "replicas": 1,
+        "selector": {
+          "component": "${APP_NAME}",
+          "deploymentconfig": "${APP_NAME}",
+          "group": "quickstarts",
+          "project": "${APP_NAME}",
+          "provider": "s2i",
+          "version": "${APP_VERSION}"
+        },
+        "template": {
+          "metadata": {
+            "creationTimestamp": null,
+            "labels": {
+              "component": "${APP_NAME}",
+              "deploymentconfig": "${APP_NAME}",
+              "group": "quickstarts",
+              "project": "${APP_NAME}",
+              "provider": "s2i",
+              "version": "${APP_VERSION}"
+            }
+          },
+          "spec": {
+            "containers": [
+              {
+                "name": "${APP_NAME}",
+                "image": "library/${APP_NAME}:latest",
+                "readinessProbe" : {
+                  "httpGet" : {
+                    "path" : "/health",
+                    "port" : 8081
+                  },
+                  "initialDelaySeconds" : 10
+                },
+                "livenessProbe" : {
+                  "httpGet" : {
+                    "path" : "/health",
+                    "port" : 8081
+                  },
+                  "initialDelaySeconds" : 180
+                },
+                "ports": [
+                  {
+                    "containerPort": 8778,
+                    "name": "jolokia"
+                  }
+                ],
+                "env" : [ {
+                  "name" : "KUBERNETES_NAMESPACE",
+                  "valueFrom" : {
+                    "fieldRef" : {
+                      "fieldPath" : "metadata.namespace"
+                    }
```

```
+                  }
+                } ],
+                "resources": {
+                  "requests": {
+                    "cpu": "${CPU_REQUEST}"
+                  },
+                  "limits": {
+                    "cpu": "${CPU_LIMIT}"
+                  }
+                }
+              }
+            ]
+          }
+        },
+        "status": {}
+    }
+  ]
+}
```

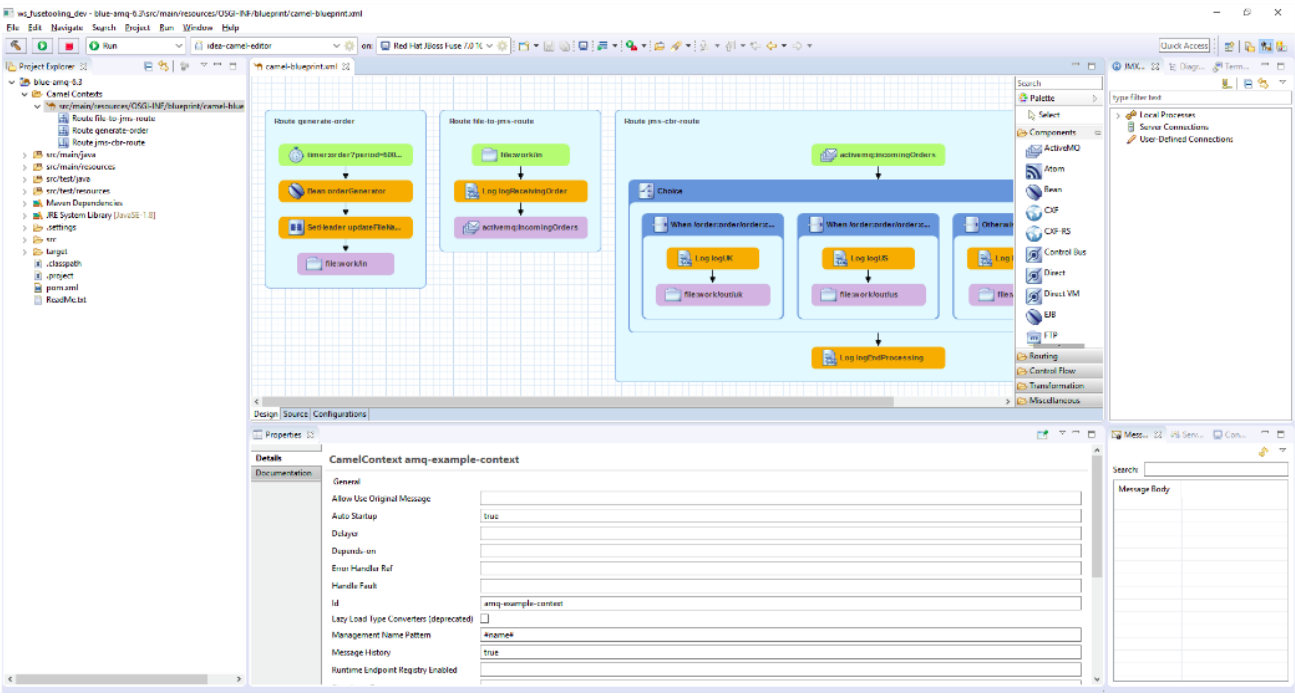
Related JIRA: JBIDE-25303

4.2. FUSE TOOLING

4.2.1. Camel Context Parameters Configurable in Properties View for Camel version < 2.18

In the Camel 2.18 and earlier versions, the Camel catalog did not have information about Camel Context. Fuse Tooling now provides this missing piece of information. Like any other component, it allows to edit Camel Context parameters in the **Properties** view. It is activated when there is no element selected on the diagram.

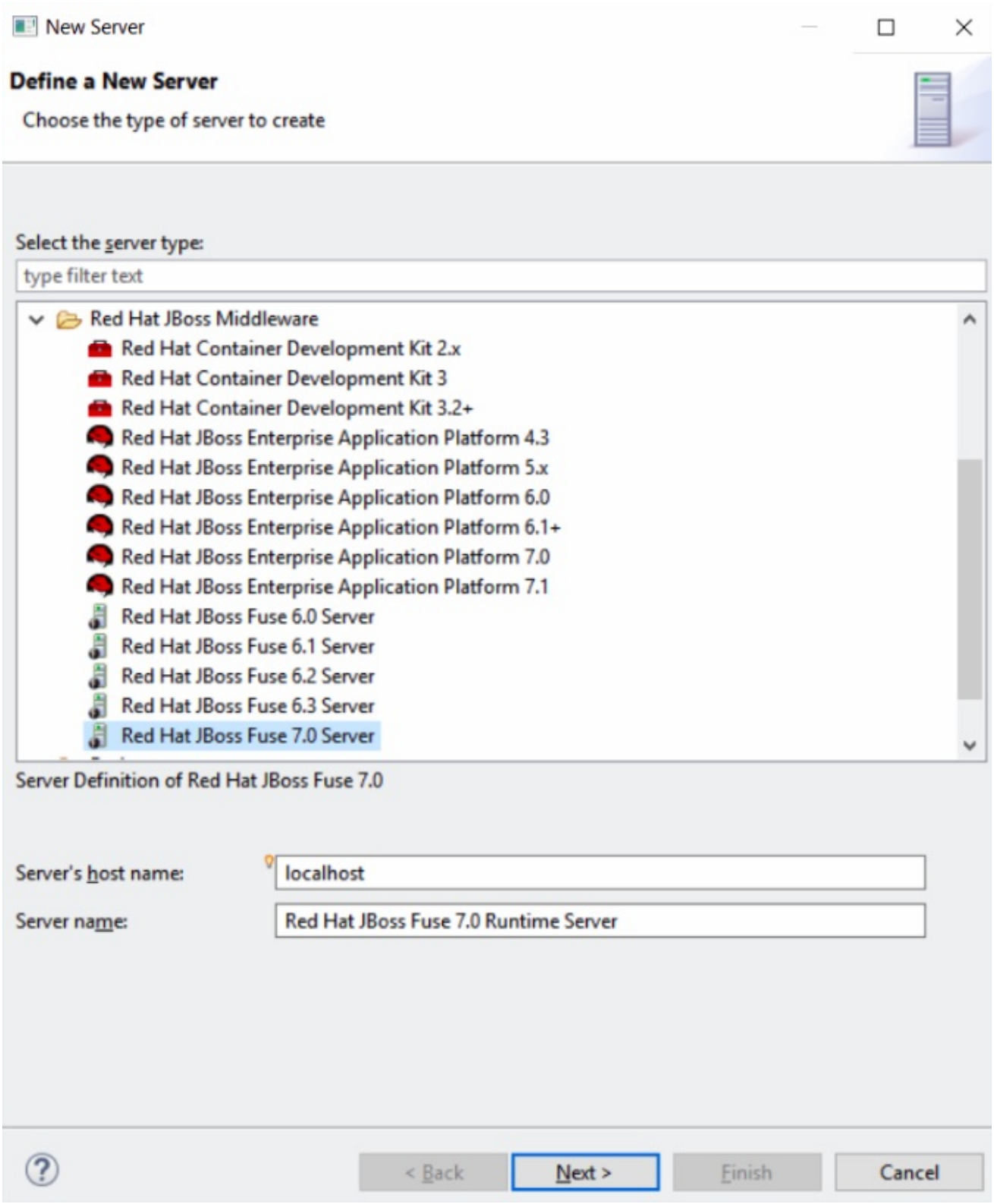
Figure 4.6. Editing the Camel Context Parameters



4.2.2. Fuse 7 Karaf-based Runtime Server Adapter

Fuse 7 is under development and preliminary versions are available on the early-access repository. Fuse Tooling is ready to leverage them so that you can try the upcoming major Fuse version.

Figure 4.7. Using the Upcoming Major Fuse Versions



Classical functionalities with server adapters are available: automatic redeploy, Java debug, Graphical Camel debug through created JMX connection. Note that you can't retrieve the Fuse 7 Runtime directly from Fuse tooling as yet. It is required to download it on your machine and point to it when creating the Server adapter. The provided templates requires some modifications to have them working with Fuse 7, mainly adapting the bom.

Related JIRA: FUSETOOLS-2578

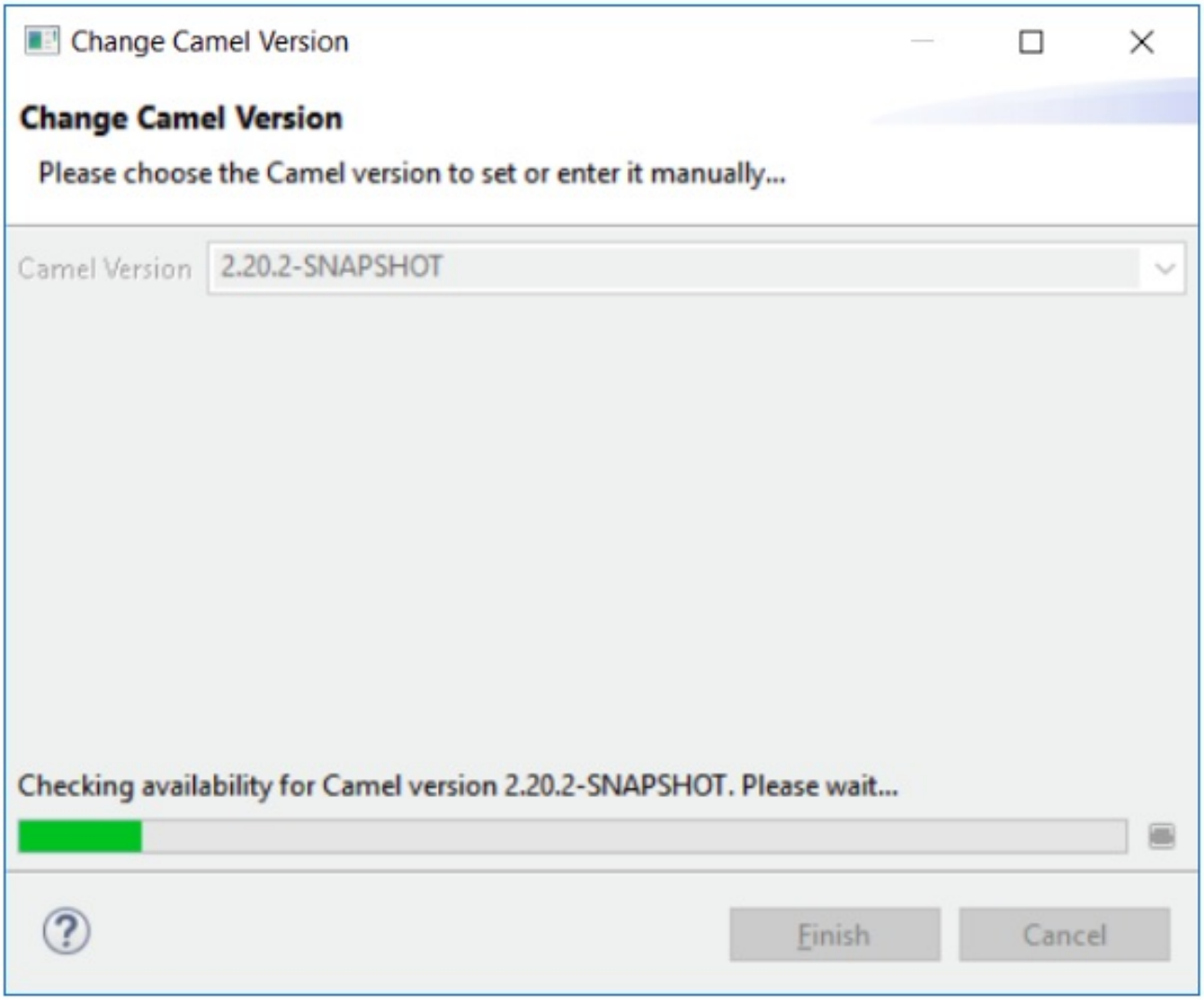
4.2.3. Display routes defined inside "routeContext" in Camel Graphical Editor (Design tab)

The **routeContext** tag is a special tag used in Camel to provide the ability to reuse routes and to split them across different files. This is useful on large projects. For details, see the Camel documentation. Starting with this version, the Design of the routes defined in the **routeContext** tags are displayed.

4.2.4. Usability improvement: Progress bar when "Changing the Camel version"

Starting with Fuse Tooling 10.1.0, it is possible to change the Camel version. In case the Camel version was not cached locally and for slow internet connections, this operation can take some time. There is now a progress bar to see the progress.

Figure 4.8. Progress Bar Showing Progress of Camel Version being Changed



CHAPTER 5. RESOLVED ISSUES

To view information about resolved issues in this release of JBoss Developer Studio, see the Resolved Issues.

CHAPTER 6. KNOWN ISSUES

To view information about known issues in this release of JBoss Developer Studio, see the Known Issues.

The following known issues are highlighted:

- JBIDE-20983: cannot use oracle service name in datasource creation
- JBIDE-19633: Not able to create 'non-bare' repository in JBDS 8.1.
- JBIDE-17176: Unable to browse and select PortletBridge runtime libraries in JPP 6
- JBIDE-12957: Xhtml files appear garbled when it’s reopened in the JBDS editor
- JBDS-3645: Installation of JBoss Developer Studio to a network drive fails
- JBDS-3470: Toolbars + Icons unusable on UHD screens
- JBDS-3069: Ungraceful shutdown results in multiple errors on startup
- JBDS-4442: Central page does not work on Fedora 26 if package webkitgtk3 is not installed
- JBIDE-25146: Eclipse annotation processing not enabled by default can result in errors
- JBDS-4617: NoClassDefFoundError: org/apache/http/client/cache/HttpCacheStorage

CHAPTER 7. APPLY THIS RELEASE

JBoss Developer Studio 11.2 is available from a number of sources:

- To install JBoss Developer Studio 11.2, use the universal installer available from the Red Hat Customer Portal.
- To install JBoss Developer Studio BYOE 11.2 in Eclipse Oxygen, use Eclipse Marketplace, the JBoss Developer Studio update site or the update *.zip* file available from the Red Hat Customer Portal.

In all cases, for more information, see the Red Hat JBoss Developer Studio Installation Guide at the JBoss Developer Studio Documentation page.