RPM Packaging

We'll be packaging an RPM for Java Mission Control.
We will adhere to Fedora Project guidelines to ensure we can easily release our package in Fedora software repositories.

This tutorial assumes basic understanding of the following:

- how to build JMC from source [see: http://hg.openjdk.java.net/jmc/jmc/file/tip/README.md#l177]
- high-level understanding of Eclipse Rich Client Platform (RCP) [see: https://wiki.eclipse.org/Rich_Client_Platform]

Install required packages

```
sudo dnf install gcc rpm-build rpm-devel rpmlint make python bash coreutils
diffutils patch rpmdevtools
```

Setup folders for RPM

```
rpmdev-setuptree
```

This will create some folders in your home directory to be used for RPM related files and build products.

Verify dependencies are available

If the package has any dependencies (build-time or run-time), they must be available in the software repositories beforehand or else you will need to package them as well. JMC does have run-time dependencies. Fortunately, most of them are already packaged by the Eclipse foundation. Some will need to be packaged. (We will also need to package the dependencies of JMC's dependencies and so on...)
We've already packaged our required dependencies (and all of their dependencies):

- JMC Core
- JavaMail
- JavaBeans Activation Framework
- Fix Info Plist Maven Plugin
- Directory Maven Plugin

Get tarball for source code

Fetch source tarball and place it in sources directory

```
wget http://hg.openjdk.java.net/jmc/jmc/archive/1ddf3baa4e26.tar.gz
mv 1ddf3baa4e26.tar.gz ~/rpmbuild/SOURCES/
```

Create SPEC file

```
cd ~/rpmbuild/SPECS
rpmdev-newspec jmc
```

Open the SPEC file in your text editor of choice and fill in the necessary directives and fields.
For reference, see: https://rpm-packaging-guide.github.io/#working-with-spec-files

Some points to note about filling in the SPEC file for JMC:

- since we are downloading a snapshot release from the upstream JMC repository, we will use snapshot versioning for the package that is built
    - see: https://docs.fedoraproject.org/en-US/packaging-guidelines/Versioning/#_snapshots
- software must be under licenses in the Fedora licensing list
- JMC is arch-specific because the build target contains files that are arch-specific (e.g. the JMC launcher)
- Fedora Project does not allow bundled libraries so we need to make sure that Eclipse libraries are excluded from the `Provides:`.
    - this is why we need to symlink to libraries that are already on the system
    - we also need to explicitly exclude `org.eclipse.*jars` from `Provides:` [see: https://docs.fedoraproject.org/en-US/packaging-guidelines/AutoProvidesAndRequiresFiltering/]
- use URLs when possible for sources (easier to verify sources)
- never use modified sources (always use directly from upstream)
    - if changes are required, use patches [see: https://rpm-packaging-guide.github.io/#patching-software]
- list all dependencies under `BuildRequres:` (for build-time dependencies) or `Requires:` (for run-time dependencies)
    - if dependencies are missing, package will fail to build or fail to run properly
    - most `Requires:` are auto-generated by the build system but you should verify that list (especially if the `%build` and/or `%intall` scripts are doing something to complicate the auto-generation)
- a `.desktop` file is required for JMC because it is a GUI application
    - see: https://docs.fedoraproject.org/en-US/packaging-guidelines/#_desktop_files
- a manpage should be included since our package will install an executable
    - see: https://docs.fedoraproject.org/en-US/packaging-guidelines/#_manpages
- in the `%files` section, list all files and directories that are owned by this package
    - there are macros to identify license and documentation files
    - in general, files that are installed by this package, should be owned by this package
- document changes between builds in the `%changelog` section with date-stamped entries

    - log packaging changes, not software changes
- add comments to the SPEC file to explain anything non-trivial

We have created a SPEC file for JMC here: https://pagure.io/jmc-rpm/raw/master/f/jmc/jmc.spec
You can also download the sources and patches for JMC from here:
https://pagure.io/jmc-rpm/blob/master/f/jmc

Build SRPM

Once all our sources and patches are ready, we can build the RPM. There are 2 types of RPMs that can be built: Source RPM (SRPM) and Binary RPM (RPM). SRPMs preserve the exact source of a certain RPM (source code, all relevant patches and exact SPEC file). SRPMs can be used to build a binary RPM for a different architecture.
For more information about source and binary RPMS, see:
https://rpm-packaging-guide.github.io/#building-rpms

We are going to build a SRPM:

```
rpmbuild -bs jmc.spec
```

Build RPM in Mock

When we build our binary RPM, we want to be certain that our build environment is clean and so we know our build is reporducible.
For more info about mock, see: https://github.com/rpm-software-management/mock/wiki

We can start by initializing a chroot in mock:

```
mock -r fedora-rawhide-x86_64 --init
```

JMC has dependencies that are not yet availabe in the DNF software repositories. We will need to install those RPMs in the chroot ourself, otherwise our build will fail. [See: https://fedoraproject.org/wiki/Using_Mock_to_test_package_builds#Building_packages_that_depend_on_packages_not_in_a_repository]

We will need to install the RPMs for JMC's dependencies in our chroot. Download the following files into `~/rpmbuild/RPMS/noarch/:

- https://copr-be.cloud.fedoraproject.org/results/sasiddiq/missioncontrol/fedora-rawhide-x86_64/00833776-dd-plist/dd-plist-1.21-1.fc30.noarch.rpm
- https://copr-be.cloud.fedoraproject.org/results/sasiddiq/missioncontrol/fedora-rawhide-x86_64/00833777-directory-maven-plugin/directory-maven-plugin-0.3.1-1.fc30.noarch.rpm
- https://copr-be.cloud.fedoraproject.org/results/sasiddiq/missioncontrol/fedora-rawhide-x86_64/00833779-owasp-java-encoder/owasp-java-encoder-1.2.2-1.fc30.noarch.rpm

- https://copr-be.cloud.fedoraproject.org/results/sasiddiq/missioncontrol/fedora-rawhide-x86_64/00833780-fix-info-plist-maven-plugin/fix-info-plist-maven-plugin-1.2-1.fc30.noarch.rpm
- https://copr-be.cloud.fedoraproject.org/results/sasiddiq/missioncontrol/fedora-rawhide-x86_64/00833781-javamail/javamail-1.6.3-2.fc30.noarch.rpm
- https://copr-be.cloud.fedoraproject.org/results/sasiddiq/missioncontrol/fedora-rawhide-x86_64/00833782-jaf/jaf-1.2.1-2.fc30.noarch.rpm
- https://copr-be.cloud.fedoraproject.org/results/sasiddiq/missioncontrol/fedora-rawhide-x86_64/00833783-jmc-core/jmc-core-7.0.0-1.20181122hg9aa7085f938b.fc30.noarch.rpm

Install the dependency RPMs in the chroot:

```
mock -r fedora-rawhide-x86_64 --install ~/rpmbuild/RPMS/noarch/*
```

Now, rebuild the SRPM in the chroot:

```
mock -r fedora-rawhide-x86_64 --rpmbuild-opts=--noclean --no-clean --no-cleanup-
after ~/rpmbuild/SRPMS/jmc-7.0.0-0.20181130hg1ddf3baa4e26.fc30.src.rpm
```

The `--no-clean` option is required so that the RPMs we manually installed earlier don't get erased. The `--rpmbuild-opts=--noclean` and `--no-cleanup-after` options preserve the `BUILDROOT` after a successful build. This may be useful in helping to diagnose rpmbuild issues we may encounter.

Lint RPMs

`rpmlint` is a tool to check SPEC files, SRPMs and Binary RPMs for common errors.
Run `rpmlint` on the SRPM and RPM for JMC:

```
rpmlint /var/lib/mock/fedora-rawhide-x86_64/result/*.rpm
```

Resolve errors and warnings from `rpmlint`.
Also check the package against this checklist for Fedora package reviewers:
https://docs.fedoraproject.org/en-US/packaging-guidelines/ReviewGuidelines/#_things_to_check_on_review

Submitting package for review

Make sure your SRPM and SPEC files are publicly available online. You may use the Copr build service for this.

Initiate the package review process by creating a BugZilla ticket:
https://bugzilla.redhat.com/bugzilla/enter_bug.cgi?product=Fedora&format=fedora-review

- initially, the `fedora-review` flag will be blank

- a reviewer will set the `fedora-review` flag to `?` if they've taken on the task of reviewing your package
- once you've fixed any issues identified by the reviewer, they will set the `fedora-review` flag to `+` if they are satisfied with the packaging

Congrats! You're package has passed review.
At this point, if you are still not sponsored into the packager group, read here for more info:
https://fedoraproject.org/wiki/How_to_get_sponsored_into_the_packager_group

fedpkg

The next step is to request a repository for your package using the package name and BugZilla ID of the review request:

```
fedpkg request-repo jmc 1649552
```

Once the repository is created, clone it locally, import SRPM, commit changes to Git, and submit a Koji build for Rawhide:

```
fedpkg clone rpms/jmc
cd jmc
fedpkg import ~/rpmbuild/SRPMS/jmc-7.0.0-0.20181130hg1ddf3baa4e26.fc30.src.rpm
fedpkg commit -m 'Initial import (#1649552)'
fedpkg push
fedpkg build
```

....And that's it! You're package is ready to be installed from the DNF repository in the next release of Fedora.

References

- https://rpm-packaging-guide.github.io/
- https://docs.fedoraproject.org/en-US/packaging-guidelines/
- https://fedoraproject.org/wiki/Package_Review_Process
- https://fedoraproject.org/wiki/Join_the_package_collection_maintainers#Becoming_a_Fedora_Package_Collection_Maintainer