Coding Style and Guidelines

Overview

This document covers common coding styles and guidelines for all ForgeRock products.

The ForgeRock Java coding style is loosely based on the Sun Java conventions. We provide a set of Checkstyle rules which can be used to enforce the code style, as well as integration for Maven based projects.

- Copyright notices
- Commit messages
- Security fixes

Copyright notices

All new source files must begin with the following copyright notice, which should be adapted accordingly for non-Java source code (e.g. XML, properties, etc):

```
/*
 * Copyright 2020 ForgeRock AS. All Rights Reserved
 *
 * Use of this code requires a commercial software license with ForgeRock AS.
 * or with one of its affiliates. All use shall be exclusively subject
 * to such license between the licensee and ForgeRock AS.
 */
```

All source files should contain copyright attributions for the people or organizations who have contributed the code. For new files this should be of the form:

```
Copyright [year] [owner]
```

The attribute should be prefixed with "Portions" for non-ForgeRock changes to existing ForgeRock-created files or for any changes to existing non-ForgeRock-created files:

```
Portions copyright [year] [owner]
```

When multiple contributions have been made in different years by the same contributor a year range should be used and kept up to date, for example:

```
Copyright 2010 Acme, Inc.
Portions copyright 2011-2016 ForgeRock AS.
```

or

```
Copyright 2011-2016 ForgeRock AS.
```

Note that copyright attributions DO NOT need to include a (c) symbol nor the phrase "All rights reserved". In addition, projects MUST include the CDDL license in its entirety in the project relative location `legal/CDDLv1.0.txt`.

Commit messages

When committing code it is essential that others can quickly get an idea of what the commit relates to, and also find more information on the issue and review. A message must always be provided when committing to trunk, sustaining or release branches/tags and we have some simple guidelines for writing them.

1. Start with the JIRA issue ID for the story or bug
2. State in up to 50 chars how this commit changes the product. Begin with a capital letter and don't end with a full stop. Write as if completing the sentence *"If applied, this commit will..."*
3. If you really need to provide further info in the commit message (info about the fix should be captured in the JIRA issue), then leave a blank line below the summary before adding the details.

Examples:

AME-9876 CR-1234 Add new authentication module for device auth

AME-9876 Add new authentication module for device auth

Security fixes

Ideally the commit message for a security fix should **only** contain the JIRA issue ID. You may also optionally provide a simple description of the general area of the fix, but you **must not mention any details of the vulnerability.**

Good examples:

OPENAM-12345

OPENAM-12345 Fix email service.

OPENAM-12345 Adjust LDAP connection settings.

Bad examples:

OPENAM-12345 Eliminate XSS in /json/sessions endpoint *- mentions a specific vulnerability and/or endpoint*

OPENAM-12345 Fix issue reported by customer *- customers often report security issues, so this is a red flag*

OPENAM-12345 JWT validation *- NB even something as simple as this should be avoided as bugs in validation are almost always security issues*

If in doubt, leave it out!