**Coding Guidelines**



**Purpose of this Document**

This document explains the coding standard that should be followed within projects.

**Document Information**

|  |  |
| --- | --- |
| Submitted by | Information |
| Name | Tanumoy Mishra |
| Company Name | Liferay India Pvt. Ltd. |
| E-mail Address | [tanumoy.mishra@liferay.com](mailto:tanumoy.mishra@liferay.com) |

**Document History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 0.1 | 4th April’ 2019 | Mohit Soni | Initial draft |
| 0.2 | 5th April’ 2019 | Tanumoy | Change in format & font |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

Contents

[1.](#_30j0zll) Module Type 3

[2.](#_1fob9te) Module Naming 3

[3.](#_3znysh7) Package Naming 3

[4.](#_2et92p0) Class Naming 4

[5.](#_tyjcwt) Defining Variables 4

[6.](#_3dy6vkm) Code Level Commenting 4

[7.](#_1t3h5sf) Coding Practice 4

[8.](#_4d34og8) Sample Class (For reference) 5

# Module Type

* + **Web Module:** A web module is a module which can be seen by guest user i.e. if module is created for end users then it be called as Web Module
  + **Admin Module**: An Admin module is a module which can be seen by admin user i.e. if module is available under control panel i.e. accessible by admin then it will be called as Admin Module.
  + **Fragment/Overriding Functionalities Module:** If a Module is created to override Liferay functionality then it will be called as Fragment/Overriding Functionalities Module.
  + **Service Builder:** A Module is called a service builder module if it has service.xml to generate DB tables i.e. provides layer to connect to DB tables.

# Module Naming

* + **Name** should always be in small letters.
  + There should not be any space in between module.
  + **Web Module:** should have -web as postfix.
  + **Admin Module:** should have -admin as postfix.
  + **Fragment:** should have -fragment as postfix.
  + **Service Builder:** Should have only functionality name.

**Example:** If we have a Requirement Called **Contact us**, then following will be created.

* **Web Module:** contact-us-web.
* **Admin Module:** contact-us-admin.
* **Service Builder:** contact-us.

# Package Naming

* + **Name** should always be in small letters
  + **Web Module:** com.{projectname}.{functionality-name}. web
  + **Admin Module:** com.{projectname}.{functionality-name}.admin
  + **Service Builder:** com.{projectname}.{functionality-name}

**Example:** If we have a Requirement Called Contact us (For abc Project) then following will be created.

* **Web Module:** com.abc.contactus.web
* **Admin Module:** com.abc.contactus.admin
* **Service Builder**: com.abc. contactus

# Class Naming

* Name should always start with a capital letter

**Example**: contact-us-web Module can have ContactusAction, ContactusUtil, etc.

# Defining Variables

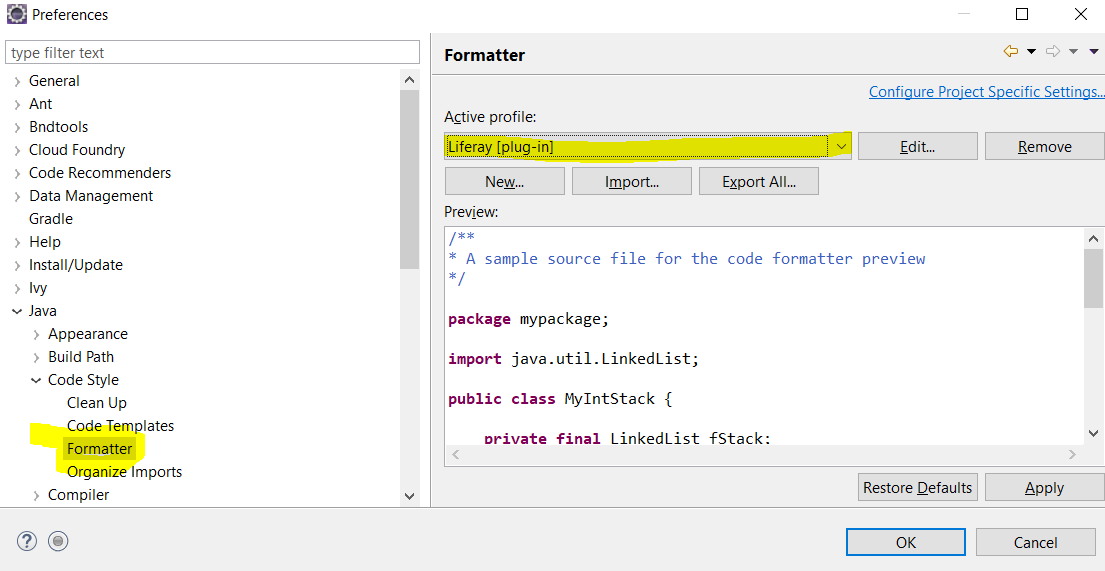
* + Should always starts with small letter.
  + Defining variable is tedious but very important part, so all the variables used in a class/jsp/free marker/script should be meaningful.
  + Never use any numbers in variables like test1, employee2, session1, etc.
  + It should always be meaningful like if you are getting ThemeDisplay object then variable name could be themeDisplay.

# Code Level Commenting

* 1. **Class Level:** Class level commenting should include following:
     + Purpose of the class i.e. usage, working etc.
     + Accessibility: The location, from where this class gets executed.
     + Author: Full name of the person who is creating the class.
  2. **Public Method Level:** Public methodlevel commenting should include following:
     + Purpose of the method, i.e. usage, working etc.
     + Parameters description.
     + Return type description.
  3. **Private Method Level:** Private methodlevel commenting should include following:
     + Purpose of the method i.e. usage, working etc.
     + Parameters description.
     + Return type description.

# Coding Practice

* + Go to eclipse.ini file available in Eclipse home.
  + Add following property:
    - -Duser={Your Full Name}
    - -Duser.name={Your Full Name}
  + Start Eclipse.
  + Click on Windows → Preferences → Java → Code Style → Formatter
  + Make Active profile as Liferay [plug-in].



* + Click on Windows → Preferences → Java → Code Style → Code Templates
  + Download CodeTemplates.xml from <https://github.com/mosoni/guidelines>
  + Click on Import and select the CodeTemplates.xml and apply.
  + Create classes/code/logic etc.
  + Once code is ready to be pushed please make sure:
    - You don’t have unused imports; unused imports can be removed by **Ctrl + Shift + O**
    - Code can be formatted using **Ctrl + Shift + F**
    - To generate commenting please press **Alt + Shift + J**
    - All System.out.println should be removed.
    - Use of logger in proper places.
    - Code should have proper comments of what code is doing.
    - Try to avoid more than 2 loops i.e. for loops.

# Sample Class (For reference)

* Download **OrderAction.java** from https://github.com/mosoni/guidelines