**Feature-Toggle**

Before we understand the feature toggle, we need to understand the development models we generally use:

1. Feature based model
2. Trunk based model

**Feature based model:**  
Feature based model is the development model in which if 2 or more developers are working on the same repository, they initially create a separate branch and start development on the same. Once the development completed and code is ready to move into real use, they merge the code to the main branch, it’s simple overview of the development approach that had been used under this model.  
  
But now as the continuous integration and development came into picture in which daily/frequently code need to be pushed to the main branch. This is where trunk-based model came into picture.

**Trunk based model:**

A trunk-based model focuses on continuous integration and frequent updates to a shared codebase. Developers work with a single branch (the "trunk") and integrate changes continuously, reducing the complexity of merging and aligning code.  
  
Now with this there are some issues that if 2 developers were working on a repository, where they pull the repository every morning and merge their changes at the end of the day. What if at the last minute the developer-2 got a mail that his feature need not to be deliver this week.  
  
Now this becomes very complex for developers to cherry pick their code and remove it from the main branch and developer 1 needs to change the login/code if he had written something with respect to the developer 2’s feature.

To solve this issue **feature toggle came into picture**.

Which works on the principle of   
  
*“****DEPLOY*** *when you* ***want*** *and* ***RELEASE*** *when you are* ***ready****”.*

**FF4J (Feature flipping for Java):**    
  
FF4J is one the open-source solution for providing support for enabling the feature toggle.  
There are many other solutions as well but for the Digital platform architecture it fits best as it is available in nexus as well and provide good support to spring boot applications. And also support Rest APIs for non-java apps.

**Flipping Strategies:**

There are many flipping strategies which we need to analyse further for understanding the real time usage of the same and how Digital platform can take benefit from the same. Some of them are listed below:   
   
**Single or Group Flip**: In this we can assign features in the same groups, so that is some features need to be removed or make the group of features together we don’t need to individually on or off each and every feature.

**Office hours:** To make the feature available between specific duration of the day.

**Release Date:** To make the feature available from specific date.

There are many others which we better understand when we perform a POC on the same.

**Structure:**Through Console we can create the feature name and save those details into the database.  
In service we need to use the FF4J class method feature check. Which will we need to add as a condition on the API’s which return the value only when get confirmation that feature is on and return the else statement when feature is off.  
  
There is approach to use the same this library to **enable this feature for UI code as well** which we **need to test once** it **worked for** **backend.  
  
As of now it looks a solution we can try to work in POC to better understand the flow and how we can use it’s feature in Digital Platform**.

Services

FF4J Server

UI CONSOLE

DB