# TP fil rouge POE.NET

## Introduction

Ce TP est à prendre dans la continuité de l'ensemble des modules de la formation, il a pour but de proposer aux stagiaires la finalisation d'un projet complet permettant de mêler front-end et back-end (Angular + programmation Java).

Le TP prend également en compte la mise en production et la configuration des serveurs Tomcat et Apache sur une distribution Linux ( à priori Ubuntu car je n'ai pas vu de distribution particulière demandée ?? ).

Pour toute question relative à ce projet, vous pouvez joindre le formateur à l'adresse mail suivante:

**contact@the-tiny-spak.com**

## Technologies utilisées

- Angular
- Javascript, Typescript
- HTML5
- CSS3
- Java et Java EE
- Webpack
- Tomcat, Apache
- Linux

## Résumé

*Le but est de monter un projet de type "boutique en ligne" tel qu'étudié pendant la session Angular. Pendant cette session, nous les stagiaires apprennent à monter une application de type SPA (Single Page Application) dont les fonctionnalités sont listées ci-dessous.*

*A noter que l'ensemble des appels serveurs se feront sur une WEB API REST programmée en Java, desservie par un serveur Tomcat, hébergée sur une machine Linux. Les stagiaires auront appris à manipuler l'ensemble de ces notions à travers les différents modules.*

*Dans un premier temps, les stagiaires travailleront en attaquant des fichiers .json statiques pour ensuite se fournir en données à travers les vrais webservices.*

## Affichage d'un catalogue produit

**route** **: /#/catalog**
**composant(s)** **: CatalogComponent & ProductComponent**

Le but est d'afficher un catalogue de produits au sein d'un composant nommé **CatalogComponent** , lequel affiche autant d'instance de **ProductComponent** qu'il y a de produits.

Les produits sont obtenus en faisant appel à un service nommé **CatalogService**, lequel effectue une requête sur le serveur ( par le biais du service **Http** du **HttpModule** ), et télécharge ainsi le contenu d'un fichier .json représentant l'intégralité du catalogue produit.

Un objet de type **ProductVo** ( équivalent d'un JavaBean ) représente un produit, le catalogue est donc un tableau d'objets de type **ProductVo**, ces derniers sont utilisés pour afficher les données du produit au sein du **ProductComponent**.

Chaque **ProductComponent** affiche une partie des informations seulement, afin d'accéder au détail produit, il a été décidé de cliquer sur un lien nous emmenant vers le détail produit.

## Affichage du détail produit

**route** : **/#/detail/:ref**
**composant(s)** : **ProductDetailComponent**

Le but est d'afficher le détail du produit, ainsi l'utilisateur peut donc lire la description produit, visusaliser son prix, connaître sa référence. Le **CatalogService** est utilisé ainsi que l'objet **ActivatedRoute** ( on se branche sur sa propriété .**params** qui est un **observable** et qui nous retourne le **paramètre** correspondant à la **référence** produit ).

## Mise en place d'un panier

- **route** : **/#/cart**
- **composant(s)** : **CartComponent**

- Création d'un **CartService**, permettant d'ajouter / supprimer un élement du panier

- Création d'un **CartComponent** permettant de visualiser l'ensemble des produits contenus dans le panier ainsi que les totaux HT et TTC (TVA 20%), il est également possible d'enlever un produit du panier.

- Ajout d'un bouton "Add To Cart" au sein du **ProductDetailComponent**

- Réutilisation des objets de type **ProductVo**, pas besoin de créer un **CartVo**