**Fiche d’Exercices Angular : Router, Formulaires, Services, Observables, Requêtes HTTP et Pipes**

**Objectifs**

1. Apprendre à configurer le routage pour naviguer entre différents composants.
2. Manipuler des formulaires Angular (template-driven et réactifs).
3. Créer et utiliser des services avec Observables pour gérer des données partagées.
4. Effectuer des requêtes HTTP pour interagir avec une API.
5. Utiliser et créer des Pipes pour transformer les données dans les templates.

**Exercice 1 : Configuration du Routage**

1. Créez un projet Angular avec trois composants :
   * HomeComponent
   * AboutComponent
   * ContactComponent
2. Configurez les routes suivantes :
   * / pour HomeComponent.
   * /about pour AboutComponent.
   * /contact pour ContactComponent.
3. Ajoutez des liens de navigation dans un <nav> utilisant routerLink.
4. Affichez le composant correspondant dans <router-outlet>.

**Exercice 2 : Navigation avec Paramètres**

1. Ajoutez une route dynamique pour afficher les détails d’un utilisateur : /user/:id.
2. Créez un composant UserDetailComponent pour afficher un utilisateur basé sur son id.
3. Simulez une liste d’utilisateurs et ajoutez un lien dans chaque élément pour naviguer vers /user/:id.
4. Récupérez l’id dans UserDetailComponent en utilisant ActivatedRoute.

**Exercice 3 : Création d’un Formulaire Template-Driven**

1. Créez un formulaire template-driven pour ajouter un produit avec les champs suivants :
   * Nom (texte, obligatoire).
   * Description (texte).
   * Prix (numérique, minimum 1).
2. Validez les champs et affichez des messages d’erreur si les conditions ne sont pas respectées.
3. Affichez les données saisies dans la console lors de la soumission.

**Exercice 4 : Création d’un Formulaire Réactif (optionnel)**

1. Créez un formulaire réactif pour enregistrer un utilisateur avec les champs suivants :
   * Nom (texte, obligatoire).
   * Email (texte, valide avec un motif d’email).
   * Mot de passe (texte, obligatoire, au moins 6 caractères).
2. Affichez des messages d’erreur pour chaque champ si les validations échouent.
3. Ajoutez un bouton qui est désactivé tant que le formulaire n’est pas valide.

**Exercice 5 : Création d’un Service pour Gérer des Données**

1. Créez un service ProductService pour gérer une liste de produits.
2. Ajoutez une méthode pour récupérer les produits.
3. Créez deux composants :
   * ProductListComponent pour afficher la liste.
   * ProductAddComponent pour ajouter un produit.
4. Partagez la liste des produits entre les deux composants via le service.

**Exercice 6 : Utilisation des Observables**

1. Modifiez ProductService pour utiliser un **BehaviorSubject** pour stocker et émettre les produits.
2. Permettez à ProductListComponent de s’abonner à cet Observable et de réagir aux mises à jour.
3. Ajoutez une méthode pour ajouter un produit, émettant la nouvelle liste à tous les abonnés.

**Exercice 7 : Requêtes HTTP avec HttpClient**

1. Utilisez une API publique comme https://jsonplaceholder.typicode.com/posts.
2. Créez un service PostService avec les méthodes suivantes :
   * getPosts() : Récupère tous les posts.
   * createPost(post) : Ajoute un nouveau post.
3. Affichez les posts dans un composant PostListComponent.
4. Ajoutez un formulaire pour créer un post, envoyant les données via createPost.

**Exercice 8 : Pipes Intégrés**

1. Utilisez des pipes intégrés pour transformer les données dans un composant PipeDemoComponent :
   * Affichez une date dans un format personnalisé avec date.
   * Transformez une chaîne en majuscules avec uppercase.
   * Affichez un prix avec currency.
2. Ajoutez un champ texte pour modifier une valeur et appliquez ces pipes dynamiquement.

**Exercice 9 : Création d’un Pipe Personnalisé (Optionnel)**

1. Créez un pipe ReversePipe qui inverse les caractères d’une chaîne.
   * Exemple : "Angular" devient "ralugnA".
2. Ajoutez un champ texte dans le template pour tester le pipe.
3. Utilisez le pipe dans un tableau pour afficher des noms inversés.

**Exercice 10 : Application Complète (Optionnel)**

1. Construisez une application simple pour gérer une liste de tâches.
2. Fonctionnalités à inclure :
   * Ajout, modification et suppression de tâches via un formulaire.
   * Utilisation d’un service pour partager les tâches entre les composants.
   * Requêtes HTTP pour sauvegarder et récupérer les tâches depuis une API.
   * Utilisation de pipes pour :
     + Formater les dates des tâches.
     + Marquer les tâches terminées avec un style personnalisé.