## Résumé de bonnes pratiques

* Le DAO est souvent statique, et il concerne les méthodes d’accès en BDD ou dans le fichier JSON.
* Dans les POCO, ne pas oublier qu’il est possible de créer plusieurs constructeurs
* Chemin relatif : En développement on se situe dans le debug ; pour accéder à la racine sans mettre de chemin absolu : « ../../../dossier ou fichier »
* Dans le DAO, il faudrait que les méthodes restent génériques quelque soit l’objet utilisé : Se servir dans les services de la classe Object : serialiser transforme en Objet, en désérialisant on peut caster avec le type d’objet à récupérer.
* Mettre in Next Id dans le Json pour préciser le prochain Id à utiliser.
* Du coup, mettre en place une structure JSON dans une nouvelle classe, qui avec les propriétés ID qui correspondent au nextId, et une propriété correspondant à la liste des objets ( ex : Produits )
* Rappel : quand on sérialise des objects, implicitement ils deviennent tous potentionellement des OBJECT -> reste à les desérialiser en type désiré.
* Méthode sur liste Find( r => r.Id = id ) sert à récupérer un élément dans une liste
* Méthode sur liste FindIndex(r => r.Id = id) sert à récupérer l’index.
* Dans les formulaires, possibilité de récupérer les infos de l’élements cliqué : cela permet une gestion de logique métier des boutons avec une seule méthode action : exemple : (Button)(sender).Name == «fzef »
* Récupérer le Content d’un bouton : (string)(Button)(sender).Content = « ajouter »
* Dans une sous fenêtre , on peut imaginer récupérer l’objet, l’instance de la mainWindow, et une chaine de caractère « mode » qui conrrespond au content du bouton et défini un mode d’action
* Pour supprimer d’une liste, ne pas oublier les méthodes Remove et RemoveAll
* Possibilité de traiter l’evt double click ; par exemple, ici pour récupérer une ligne d’un DataGrid : (Object)(DataGridRow)sender.Item
* Dans le cas d’un BDD json, le servie connait la Path, et il est inutile d’ajouter des controleurs.