Examen 1 - Formatif

# 420-5W5

## Consignes

* Toute documentation est permise pour l’examen formatif et sommatif.

# Révision 1 – Tests unitaires et couverture de code

Créer un nouveau projet MVC. Ajoutez les modèles Bill et Item et le service BillService **(Le code est sur les prochaines pages**). Générez un test unitaire pour la fonction Create de BillService.

Révision : pour créer des tests unitaires pour UneMethode d’UnService, simplement faire :

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

(Ça génère même le projet !)

Ajoutez des tests unitaires jusqu’à avoir **couvert complètement le code (vérifiez avec l’outil pour voir la couverture de votre code)**. Utilisez **InMemoryDatabase** comme dans les notes de cours. Il faut valider chaque message d’erreur des **exceptions** et vos tests doivent tous passer **(tout en vert)**. Pour valider une exception, vous pouvez faire :



(Suite prochaine page)

BillService.cs

public class BillService

{

    private ApplicationDbContext \_context;

    public BillService(ApplicationDbContext context)

    {

        \_context = context;

    }

    public Bill? CreateBill(string name, IEnumerable<Item> items)

    {

        if (name == null || name.Length == 0)

            throw new Exception("You need a name to create a bill");

        if (items.Count() == 0)

            return null;

        foreach(Item item in items)

        {

            if (item.Price < 0)

            {

                throw new AreYouInsaneException("Not paying for you to take something!");

            }

            if (item.Price == 0)

            {

                throw new AreYouInsaneException("Not giving free stuff either!!");

            }

        }

        Bill bill = new Bill()

        {

            Name = name,

            Items = items

        };

        \_context.Add(bill);

        return bill;

    }

}

Bill.cs

public class Bill

{

    public int Id { get; set; }

    public string Name { get; set; }

    public decimal Price { get; set; }

    public virtual IEnumerable<Item> Items { get; set; }

}

Item.cs

public class Item

{

    public int Id { get; set; }

    public string Name { get; set; }

    public decimal Price { get; set; }

}

AreYouInsaneException.cs

public class AreYouInsaneException : Exception

{

    public AreYouInsaneException(string message) : base(message) { }

}

# Question 2 – Animations et délais

À partir de l’exercice sur les animations que nous avons déjà fait en classe. Ajoutez un bouton « Shake then bounce » pour chacun des cubes (css + angular). Modifiez les temps des 2 animations pour 3 secondes, sans délai. Le bouton « shake then bounce » doit faire jouer l’animation shake pendant 3 secondes et ensuite l’animation bounce pendant 3 secondes.

# Révision 3 – Guard et cookies

Créer un nouveau projet Angular. Remplacez la page principale par 2 checkbox et un bouton.

Les 2 check boxes sauvegarde leur état dans un cookie et ont les textes suivants :

« J’aime les choses sucrés ! »

« J’aime les choses salés ! »

Le bouton affiche « J’ai faim ! » et mène vers la page /**caramelAuSel** par défaut (avec une image de caramel au beurre salé)

Ajoutez un **guard** pour la page **caramelAuSel** et rediriger vers :

/**verreDEau** si l’utilisateur n’aime ni le sucre, ni le sel (avec une image de verre d’eau)

/**bonbon** si l’utilisateur n’aime pas le sel, mais aime le sucre (avec une image de bonbon)

/**sel** si l’utilisateur aime le sel, mais pas le sucre (avec une image de sel)

# Révision 4 –Reactive Forms

Faire la validation du formulaire suivant :

* Votre chiffre préféré (nous appellerons la valeur de ce chiffre N)
* Une description
* Bouton soumettre

Les deux champs sont requis.

Le chiffre préféré doit être entre 1 et 9 (ben oui… c’est un chiffre, pas un nombre).

La description doit contenir exactement N mots qui doivent contenir N lettres.

Donc si le chiffre préféré est 4, la description : « Vous êtes trop lent » est valide.

Mais la description « Salut » donne le message d’erreur : « La description doit avoir 4 mots. »

Et la description «Vous vous êtes trompés » est invalide (le 4e mot a 7 lettres).

Le message d’erreur : « Le mot trompés contient 7 lettres et devrait en contenir 4 ».

Le bouton soumettre est seulement actif si le formulaire est valide, mais il ne fait rien.