1.1.

Dans le dossier Classes, créez un nouveau fichier nommé ArticlesRss.cs contenant une classe RssElement. Cette classe possédera les propriétés suivantes :

Category : Du type Category, elle permettra d'associer les articles récupérés à leur catégorie dans notre application.

EndPoint : Du type Uri et caractérisant l'adresse HTTP à laquelle récupérer le contenu XML

Articles : La liste d'articles que nous souhaitons récupérer

Notre classe RssElement possèdera également une unique méthode, Fetch, qui réalisera la construction des objets Article et retournant un booléen : vrai si l'opération a réussi, false si elle a échoué.

1.2.

La méthode Fetch doit tout d'abord charger de manière asynchrone le flux RSS en appelant la méthode LoadFromUriAsync de la classe XmlDocument. En cas d'erreur à cette étape, la méthode doit retourner faux.

Après la récupération, utilisez les différentes balises contenues dans l'élément XML channel pour définir les différentes propriétés de la catégorie :

Title

Description

Link

Language

Copyright

Elle doit ensuite récupérer la liste des éléments IXmlNode correspondant aux balises item présentes dans notre flux RSS. Pour chacun de ces éléments, il nous faut récupérer leurs attributs :

title

link

description

pubDate

Après avoir vérifié qu'ils n'étaient pas absents, nous avons toutes les informations nécessaires pour créer notre objet Article, à l'exception de son identifiant. Utilisons la propriété Category de notre objet RssElement pour définir la catégorie de l'article en cours.

Après avoir traité tous les articles, cette méthode Fetch doit retourner vrai.

1.3.

Pour terminer notre classe RssElement, il nous suffit d'inclure la dernière propriété non définie de nos objets Article, leur identifiant.

Dans la balise link de nos articles, un format se dessine. Tous les articles ont un lien de la forme suivante : https://www.supinfo.com/articles/single/{identifiant}-{titre de l'article}

Utilisez les méthodes d'instance des chaînes de caractère pour isoler l'identifiant. N'oubliez pas de transformer cet identifiant en type int soit via la méthode Parse, soit via la méthode TryParse.

Ce format d'adresse étant spécifique à notre application, créez une constante dénommée ID\_PATTERN dans RssElement pour le stocker.

.4.

Créons maintenant la classe RssReader. Elle possèdera les propriétés suivantes :

Elements : Une liste d'objets RssElement

ElementsChanged : Une propriété EventHandler symbolisant la fin de chargement d'un objet RssElement.

LoadCompleted : Une propriété EventHandler symbolisant la fin de chargement de tous les objets RssElement.

Pour réduire les risques d'appels multiples, nous utiliserons également le modèle de design Singleton. Si vous ne connaissez pas le modèle Singleton, n'hésitez pas à suivre ce lien.

Enfin, nous stockerons au sein d'une constante dénommée ARTICLES\_URI le modèle de l'adresse HTTP utilisé pour récupérer les flux RSS. Pour rappel, elle est de la forme suivante : [https://www.supinfo.com/api/supinfo?action=rss&tags={id}](https://www.supinfo.com/api/supinfo?action=rss&tags=%7bid%7d)

1.5.

Créons maintenant différentes méthodes au sein de notre classe RssReader :

Load : Cette méthode prend en paramètre un objet Uri et un objet Category

Son but est de créer un nouvel objet RssElement, d'en définir les propriétés EndPoint avec le premier paramètre et Category avec le second, puis d'en appeler la méthode Fetch avec le mot-clef await.

Start : Cette méthode sera appelée par le constructeur pour démarrer la récupération des données.

Au sein de cette méthode, il faudra créer une boucle itérant de 1 à 19 et en son sein d'appeler la méthode Load définie ci-dessus. En utilisant l'itérateur, nous pouvons créer les deux paramètres.

1.6.

Créons maintenant la liaison avec nos événements.

Pour ce faire, la première étape est d'ajouter dans la classe RssElement une propriété booléenne Completed, qui nous permettra de suivre l'état du chargement. Cette propriété sera définie à vrai quand l'opération asynchrone de la méthode Fetch sera complétée.

Ensuite, il nous faut créer une méthode OnElementCompleted, prenant en paramètre notre objet RssElement récemment complété. Cette méthode devra :

Invoquer l'événement ElementsChanged. Son premier paramètre sera l'objet RssElement venant juste d'être complété. Le deuxième argument sera EventArgs.Empty, une propriété spéciale de la classe EventArgs utilisée quand aucun paramètre supplémentaire ne doit être transmis.

Vérifier si tous les objets RssElement sont complétés. Si c'est le cas, il nous faudra invoquer le deuxième événement, LoadCompleted. Le première paramètre de cet appel sera l'objet RssReader, et le deuxième paramètre EventArgs.Empty.

1.7.

Dernière étape pour la liaison avec notre page en XAML. Il nous faudra notre instance de classe RssReader non initialisée en propriété de notre classe MainPage.

Dans le constructeur de notre page, obtenons l'instance de RssReader et écoutons l'événement LoadCompleted. Visual Studio vous permet d'auto-compléter cette instruction en écrivant RssReader.LoadCompleted +=, puis en appuyant sur la touche Tabulation.

Dans cette nouvelle méthode, récupérons l'ensemble des articles grâce à LINQ et sa méthode SelectMany, et pour chacun, ajoutons les à la collection observable Articles de notre MainPage.

N'oubliez pas de supprimer vos articles de test rédigés à la section 2 de ce chapitre.