API – MANUEL UTILISATEUR

INSTALLATION

Ajouter en référence la dll de notre API

```
using APIFiMag ;
using APIFiMag.Datas ;
using APIFiMag.Importer ;
using APIFiMag.Exporter ;
```

CONSTRUCTION D'UN OBJET DATA

Pour stocker les données, nous utilisons une classe `abstract public class Data`
Selon le type de données voulu, il faut utiliser le constructeur de la classe héritée correspondant :

- Pour des données historiques sur des cours :

```
public DataActif(List<string> symbol,
                 List<HistoricalColumn> columns,
                 DateTime debut,
                 DateTime fin)
```

- Pour une récupération de cours en temps réel :

```
public DataRealTime(string symbol)
```

- Pour des taux de change récupérés sur le site fxtop.com :

```
public DataFXTop(Currency symbol,
                 List <Currency> columns,
                 DateTime debut, fin,
                 Frequency freq)
```

- Pour des taux d'intérêts :

```
public DataIRate(InterestRate symbol, DateTime debut, DateTime fin)
```

- Pour faire une récupération à partir d'un fichier de configuration XML :

```
public DataXML(XmlDocument doc)
```

Par exemple pour construire un objet contenant des données de taux de change :

```
List<Currency> curr = new List<Currency>();
curr.Add(Currency.USD);
curr.Add(Currency.JPY);
curr.Add(Currency.LBP);
DataFXTop xchange = new DataFXTop(Currency.EUR, curr,
                                  new DateTime(2012, 6, 9),
                                  DateTime.Now,
                                  Frequency.Yearly);
```

RECUPERATION DE DONNEES

Une fois un objet de type `Data` construit, il reste à remplir son champ `DataSet Table`
Pour cela on appelle la méthode `public void ImportData(Import i)` en choisissant l'implémentation de l'interface `Import` voulue parmi :

```
public class ImportYahoo      : pour des données historiques sur des cours
public class ImportGoogle     : pour des données historiques sur des cours
public class ImportReal       : pour des cours en temps réel (prix, bid et ask)
public class ImportEBF        : pour des taux d'intérêts
public class ParserFXTop      : pour des taux de change
```

Par exemple, après avoir construit un objet `DataActif donnees_histo` pour des données historiques :

```
donnees_histo.ImportData(new ImportYahoo());
```

Ou pour des taux d'intérêts, avec un objet `DataIRate donnees_taux` :

```
donnees_taux.ImportData(new ImportEBF());
```

EXPORTATION DE DONNEES

Classes d'exportation de données qui implémentent l'interface `Export` :

```
public class ExportCSV
public class ExportJSON
public class ExportMDF
public class ExportXML
```

Les constructeurs associés sont :

```
public ExportCSV(string filePath, CultureInfo culture)
public ExportCSV(string filePath)
public ExportJSON(string filePath)
public ExportMDF(string filePath)
public ExportXML(string filePath)
```

Dans le cas de l'exportation en format CSV le paramètre optionnel `CultureInfo culture` permet de spécifier la norme voulue.
En effet dans la norme française les séparateurs sont des points-virgules, dans les autres normes ce sont des virgules.
Si le paramètre n'est pas spécifié, c'est la culture de la machine exécutant le programme qui est choisie.

Exemple d'utilisation avec une donnée `Data d` qui a été remplie au préalable :

```
d.Export(new ExportCSV("output_format_CSV.csv", new CultureInfo("en-US")));
```

EXEMPLE COMPLET DE RECUPERATION DE DONNEES

```
using APIFiMag ;
using APIFiMag.Datas ;
using APIFiMag.Importer ;
using APIFiMag.Exporter ;


// On veut récupérer des taux de change sur fxtop.com,
// on commence par construire un DataFXTop

            List<Currency> curr = new List<Currency>();
            curr.Add(Currency.USD);
            curr.Add(Currency.JPY);
            curr.Add(Currency.LBP);

// Ici, on s'intéresse à l'euro par rapport aux autres :
// EUR/USD, EUR/JPY, EUR/LBP

            DataFXTop xchange = new DataFXTop(Currency.EUR,
                                    curr,
                                    new DateTime(2012, 6, 9),
                                    DateTime.Now,
                                    Frequency.Yearly);


// On récupère les données, on les parse et on remplit xchange

            xchange.ImportData(new ParserFXTop());


// On exporte les données récupérées, dans le format de notre choix,
ici CSV
            xchange.Export(new ExportCSV("recupFxTop.csv"));
```