



Objectif : Écrire des tests automatisés sur le code suivant ...

```
public partial class MainForm : Form
{
    public MainForm()
    {
        InitializeComponent();
        _dataGridView1.AutoGenerateColumns = false;
    }

    private void btnRefresh_Click(object sender, EventArgs e)
    {
        using (MiamDbContext context = new MiamDbContext())
        {
            int nb = ((int)DayOfWeek.Friday - (int)DateTime.Today.DayOfWeek + 7) % 7;
            DateTime dateTime2 = DateTime.Today.AddDays(nb);
            List<Repas> repas = context.Repas.Where(x => DateTime.Today <= x.Date
                && x.Date <= dateTime2).ToList();

            if (!repas.Any())
            {
                _dataGridView1.DataSource = new List<Repas>();
                MessageBox.Show("Pas de repas!");
            }
            else
            {
                _dataGridView1.DataSource = repas;
            }
        }
    }
}
```

Technique 1 : Séparer 2 logiques (UI & Business) avec une « Sprout Class »

```
public partial class MyForm
{
    public void MéthodeOriginaleÀTester()
    {
        Stuff[] stuff = ... // logique XYZ utilisant par exemple _datePicker.Value
        this._uiComponent2.DataSource = stuff; // action sur le UI
        this.Title = $"{stuff.Length}"; // autre action sur le UI
    }
}
```



```
public partial class MyForm : IView
{
    public void MéthodeOriginaleÀTester()
    {
        var sprout = new SproutClass(this);
        sprout.MéthodeTestée(_datePicker.Value);
    }

    void IView.AfficherStuff(Stuff[] stuffs)
    {
        _uiComponent2.DataSource = stuffs;
    }

    void IView.ChangerTitre(string nvxTitre){
        this.Title = nvxTitre;
    }
}
```



```
public class SproutClass
{
    private readonly IView _classeOriginale;


    public SproutClass(IView classeOriginale)
    {
        _classeOriginale = classeOriginale;
    }

    public void MéthodeTestée(DateTime date)
    {
        Stuff[] s = ... // logique XYZ
        _classeOriginale.AfficherStuff(s);
        string nvxTitre = $"{s.Length}";
        _classeOriginale.ChangerTitre(nvxTitre);
    }
}
```




Technique 2 : Utiliser une factory pour « mocker » les classes à instanciation tardive (Disposable, Open/Close, Connect/Disconnect...)

```
public void MéthodeATester()
{
    using(DbContext dbContext
        = new DbContext())
    {
        string[] myStuff = dbContext.GetStuff();
        // ..
    }
}
```




```
public void MéthodeATester()
{
    using(IDBContext dbContext = _factory.Create())
    {
        string[] myStuff = dbContext.GetStuff();
        // ...
    }
}
```




Technique 3 : Introduire un paramètre pour se débarrasser d'une dépendance (System.DateTime.Today)

```
public DateTime MéthodeOriginaleÀTester()
{
    int nb = ((int)DayOfWeek.Friday -
        (int)DateTime.Today.DayOfWeek + 7) % 7;
    return DateTime.Today.AddDays(nb);
}
```




```
public DateTime MéthodeTestée(DateTime dateTime)
{
    int nb = ((int)DayOfWeek.Friday -
        (int)dateTime.DayOfWeek + 7) % 7;
    return dateTime.AddDays(nb);
}

public DateTime MéthodeOriginaleÀTester() {
    return MéthodeTestée(DateTime.Today);
}
```




Technique 4 : Utiliser l'injection de dépendance du pauvre pour se débarrasser d'une dépendance (MessageBox.Show)

```
public class ClassÀTester
{
    public void MéthodeATester() {
        string[] stuff = SomeBusinessLogic();
        if (!stuff.Any())
            MessageBox.Show("A message");
        SomeMoreBusinessLogic();
    }
}
```



```
public class UIFeedBack : IUIFeedBack
{
    void IUIFeedBack.ShowMessage(string mes)
    {
        MessageBox.Show(mes);
    }
}
```




```
public class ClassÀTester
{
    private readonly IUIFeedBack _uiFeedBack;

    public ClassÀTester(UIFeedBack uiFeedBack) {
        _uiFeedBack = uiFeedBack;
    }

    public ClassÀTester():this(new UIFeedBack()){ }

    public void MéthodeATester(){
        string[] stuff = SomeBusinessLogic();
        if (!stuff.Any())
            _uiFeedBack.ShowMessage("A message");
        SomeMoreBusinessLogic();
    }
}
```



Aide mémoire Moq

Un « **Stub** »
avec Moq

```
Mock<IDateTimeProvider> dateTimeProvider = new Mock<IDateTimeProvider>();
dateTimeProvider.Setup(x => x.Today).Returns(new DateTime(2018, 10, 23));
```

Un « **Spy** »
avec Moq

```
Mock<IUIFeedBack> ui = new Mock<IUIFeedBack>();
ui.Verify(x=>x.ShowMessage(It.IsAny<string>()), Times.Once);
```



Aide mémoire Resharper

Show action list

Alt+Entrée

Introduce Parameter

Ctrl+Alt+P

Refactor this

Ctrl+Shift+R

Introduce Variable

Ctrl+Alt+V

Search Everywhere

Ctrl+N

Extract Method

Ctrl+Alt+M