Les tests unitaires

Yannick Mazières

Les balises pour les tests (MsTest)

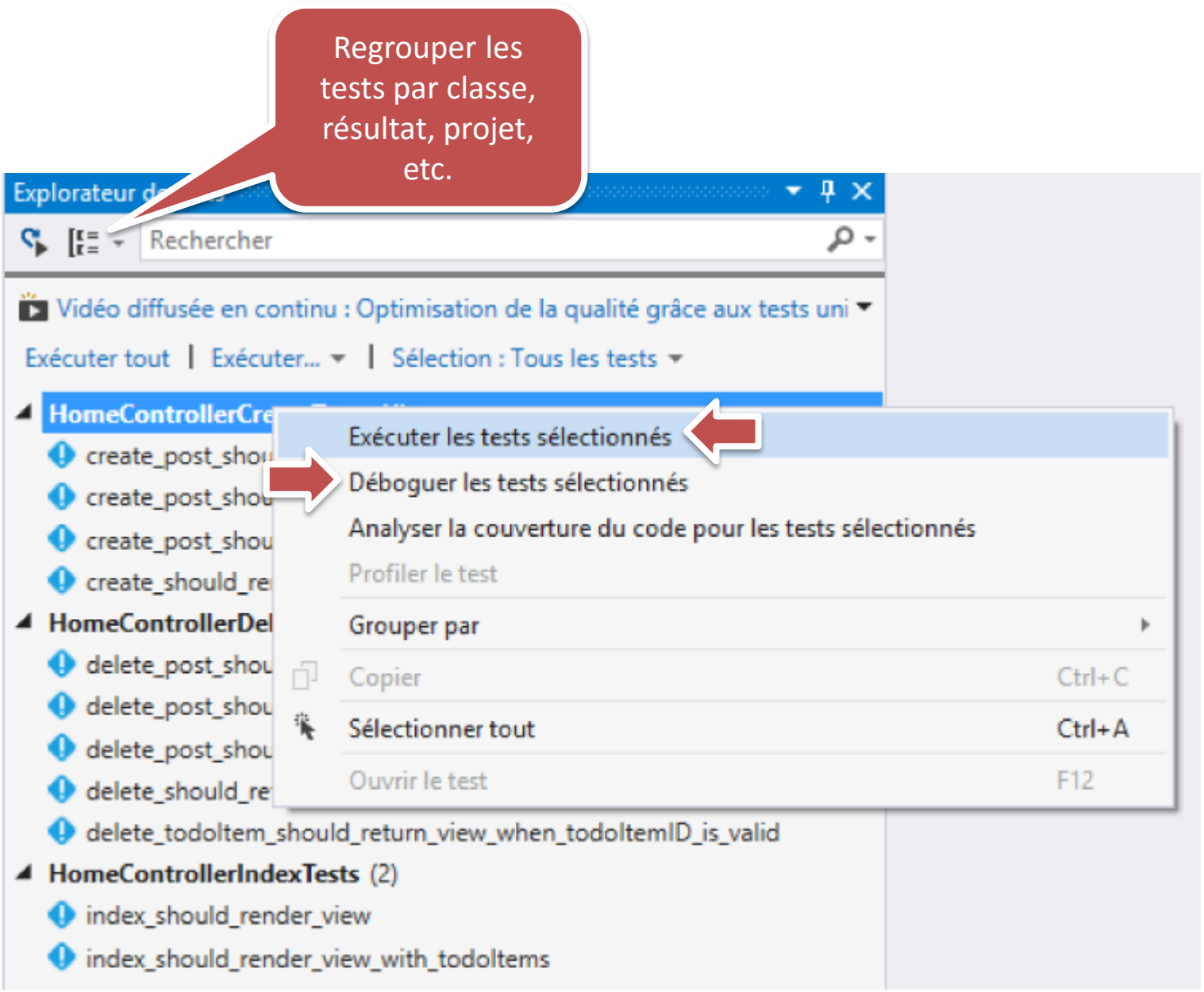
[TestInitialize] s'exécute **avant** chacun des tests

Utiliser **[Ignore]** au lieu de commenter le test

[TestCleanup] s'exécute **à la fin** de chacun des tests

```
[TestClass]
public class Nom_de_la_classe
{
    [TestInitialize]
    public void test_initialize()
    {
        //code…
    }

    [TestMethod]
    public void nom_du_test_1()
    {
        //code…
    }

    [Ignore]
    [TestMethod]
    public void  nom_du_test_2()
    {
        //code…
    }

    [TestCleanup]
    public void test_cleanup()
    {
        //code…
    }
}
```

L'explorateur de tests de VS2013



Exemple 1

On veut tester si la méthode **index** de **HomeController** retourne une vue

CONTRÔLEUR

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View();
    }
}
```

TEST

```
[TestClass]
public class HomeControllerTests
{
    [TestMethod]
    public void index_should_render_view()
    {
        // Arrange
        var homeController = new HomeController();
        // Action
        var result = homeController.Index() as ViewResult;
        // Assert
        Assert.AreEqual(result.ViewName, "");
    }
}
```

Mise en place pour le test

Exécution du comportement à tester

Validation des résultats

"" est la vue par défaut

Exemple 2

On veut tester si la méthode **index** de **HomeController** retourne une vue avec des données

```
@model IEnumerable<Todo.Web.Models.TodoItem>
<html>
   <head>
      <title>title</title>
   </head>
   <body>
      @foreach (var todoItem in Model)
      {
         <div>
            @todoItem.Title
         </div>
      }
    </body>
</html>
```

VUE

```
public class HomeController : Controller
{
  public ActionResult Index()
  {
     var model = new List<TodoItem>()
     {
        new TodoItem()
        {
           Title = "Faire le ménage"
        },
        new TodoItem()
        {
           Title = "Laver le chat"
        }
     };
     return View(model);
  }
}
```

CONTRÔLEUR + MODÈLE

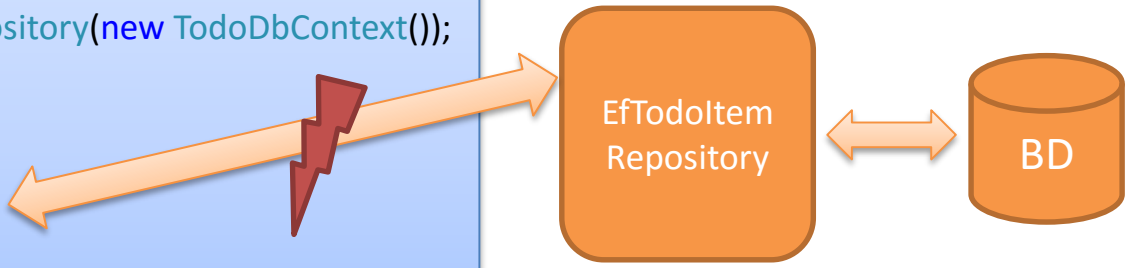
TEST

```
[TestMethod]
public void index_should_render_view_with_todoItems()
{
        var homeController = new HomeController();

        var result = homeController.Index() as ViewResult;
        var model = result.Model as IEnumerable<TodoItem>;

       Assert.AreNotEqual(model.Count(), 0);
}
```

Suppression des commentaires *Arrange*, *Action*, *Assert* mais **séparation en paragraphe**.

Exemple 3 - Test avec un *repository*

```
public class HomeController : Controller
{
    private   EfTodoItemRepository todoItemRepository ;

    public HomeController()
    {
        todoItemRepository = new EfTodoItemRepository(new TodoDbContext());
    }
    public ActionResult Index()
    {
        var model =  todoItemRepository.GetAll();
        return View(model);
    }
}
```




```
[TestMethod]
public void index_should_render_view_with_todoItems()
{
    //arrange
    var homeController = new HomeController();
    var todoItemRepository =
            new EfTodoItemRepository(new TodoDbContext());
    var todoItem = new TodoItem()
    {
        Id = 1,
        Title = "Faire cuire un oeuf"
    };
    todoItemRepository.Add(todoItem);
    //action
    var result = homeController.Index() as ViewResult;
    var model = result.Model as IEnumerable<TodoItem>;
    //assert
    Assert.AreNotEqual(model.Count(),0);
}
```

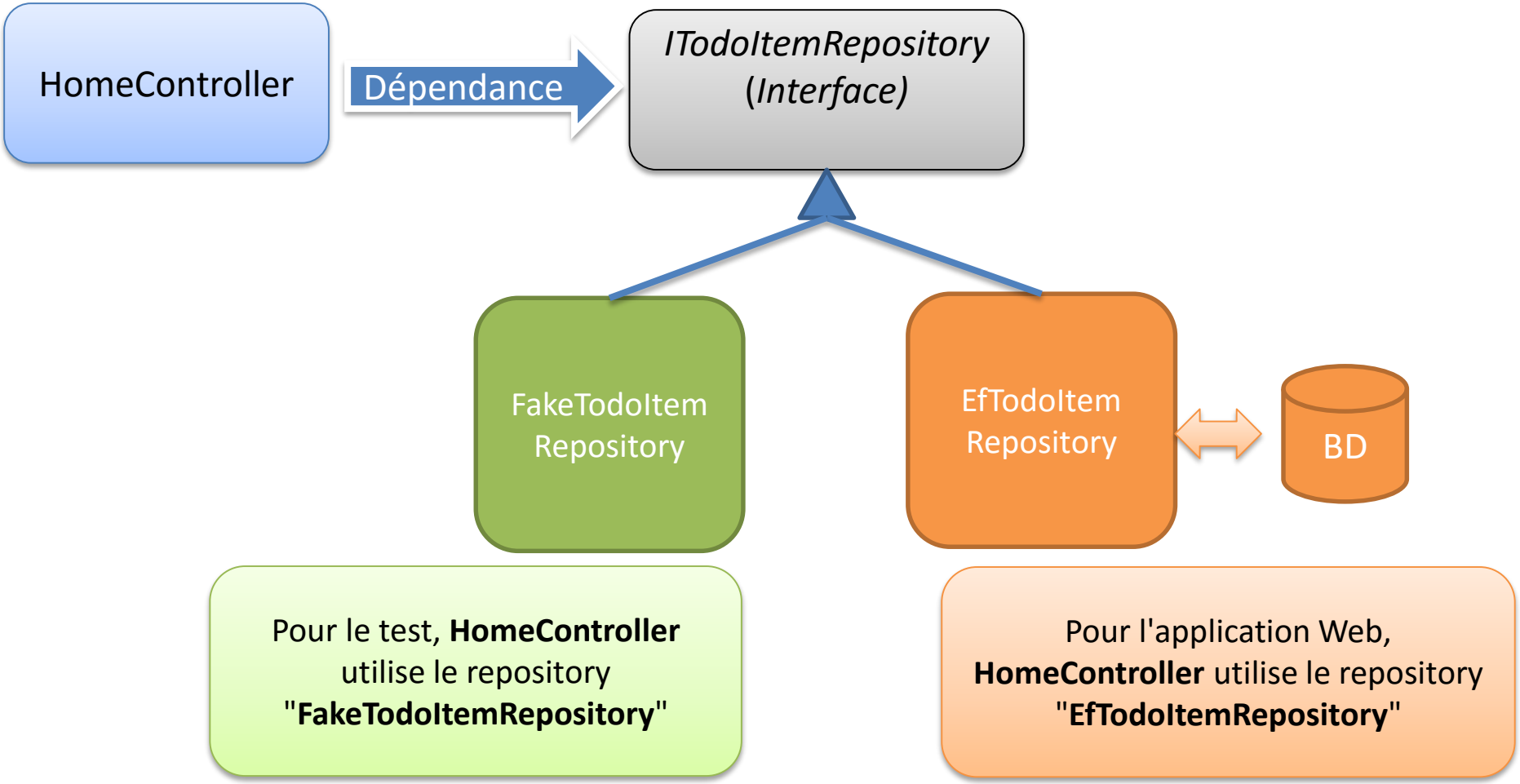
Quel est le principal problème avec ce test unitaire ?

Il n'est pas unitaire.
Il est impossible de tester unitairement **Index** étant donné la **dépendance** avec la classe **EfTodoItemRepository.**

Que faire pour tester et couper la dépendance à **EfTodoItemRepository** ?

Injecter la dépendance (par le constructeur de **HomeController**) à un objet qui simule le comportement de la BD (**FakeTodoItemRepository**).

Injection de dépendance



L'injection d'une dépendance peut se faire de plusieurs façons. Dans le cadre du projet synthèse, ce sera par le constructeur.

Injection de dépendance par le constructeur

```
public class HomeController : Controller
{
    private ITodoItemRepository _todoItemRepository;

    public HomeController()
    {
        _todoItemRepository = new EfTodoItemRepository(new TodoDbContext());
    }

    public HomeController(ITodoItemRepository todoItemRepository)
    {
        _todoItemRepository = todoItemRepository;
    }

    public ActionResult Index()
    {
        var model = _todoItemRepository.GetAll();
        return View(model);
    }
}
```

Constructeur appelé par l'application Web (framework MVC.Net)

Constructeur appelé par les tests (injection de la dépendance à **FakeTodoItemRepository**)

Utilisation de **EfTodoItemRepository** ou **FakeTodoItemRepository** selon le constructeur appelé

Nous verrons plus tard comment avoir **un seul constructeur** pour les tests et pour l'application web.

Le code...

```
public interface ITodoItemRepository
{
        List<TodoItem> GetAll();
        TodoItem GetById(int todoItemId);

        void Add(TodoItem todoItem);
}
```

```
public class EfTodoItemRepository : ITodoItemRepository
{
  private TodoDbContext _dbContext;

  public EfTodoItemRepository(TodoDbContext dbContext)
  {
     _dbContext =  dbContext;
  }
  public List<TodoItem> GetAll()
  {
      return  _dbContext.TodoItems.ToList();
  }
  public void   Add(TodoItem todoItem)
  {
       ...
```

```
public class FakeTodoItemRepository : ITodoItemRepository
{
        private List<TodoItem> _todoItems;

        public FakeTodoItemRepository()
        {
            _todoItems = new List<TodoItem>();
        }
        public List<TodoItem> GetAll()
        {
            return _todoItems;
        }
        public void Add(TodoItem todoItem)
        {
            _todoItems.Add(todoItem);
        }
        public TodoItem GetById(int todoItemId)
        {
            return _todoItems.Find(x => x.Id == todoItemId);
        }
}
```

Tester si une vue affiche des données attendues

À tester

```
// Une section de code d'une méthode Delete en « http get »
...
TodoItem totoItem = _todoItemRepository.GetById(todoItemId);
return View(totoItem);
```

Le test

```
// Arrange
var fakeTodoItemRepository = new FakeTodoItemRepository();
var homeController = new HomeController(fakeTodoItemRepository);
var todoItem = new TodoItem()
{
    Id = 1,
    Title = "Laver le chat dans la piscine"
};
fakeTodoItemRepository.Add(todoItem);

//Action
var result =  homeController.Delete(todoItem.Id) as ViewResult;
var model = result.ViewData.Model as TodoItem;

//Assert
Assert.AreEqual(todoItem,model);
```

Tester une erreur dans le modèle

(les données du modèle ne sont pas valides)

À tester

```
// Une section de code d'une méthode Create en « http post »
...
if (!ModelState.IsValid)
{
        return View();
}
```

Le test

```
//Arrange
var fakeTodoItemRepository = new FakeTodoItemRepository();
var homeController = new HomeController(fakeTodoItemRepository);
var todoItem = new TodoItem()
{
    Id = 1,
    Title = "Faire cuire un boeuf"
};
homeController.ModelState.AddModelError("Error", "Error");
//Action
var result = homeController.Create(todoItem) as ViewResult;

//Assert
Assert.AreEqual(result.ViewName, "");
```

Tester une redirection

À tester

```
// Une section de code d'une méthode Create d'un « http post »
...
return RedirectToAction("Index");
```

Le test

```
//Arrange
var fakeTodoItemRepository = new FakeTodoItemRepository();
var homeController = new HomeController(fakeTodoItemRepository);
var todoItem = new TodoItem()
{
   Id = 1,
   Title = "Faire cuire un oeuf"
};

// Action
var result = homeController.Create(todoItem) as RedirectToRouteResult;
var action = result.RouteValues["Action"];

// Assert
Assert.AreEqual("Index", action);
```

Tester HttpNotFound

À tester

```
// Une section de code d'une méthode Edit en « http get »
…
return HttpNotFound();
```

Le test

```
//Arrange
var fakeTodoItemRepository = new FakeTodoItemRepository();
var homeController = new HomeController(fakeTodoItemRepository);
const int ID_INEXISTANT = 999999999;

//Act
var result = homeController.Edit(ID_INEXISTANT);

//Assert
Assert.IsInstanceOfType(result, typeof(HttpNotFoundResult));
```

Tester l'ajout d'un item dans un *repository*

À tester

```
// Une section de code d'une méthode Create en « http post »
...
_todoItemRepository.Add(todoItem);
```

Le test

```
// Arrange
var fakeTodoItemRepository = new FakeTodoItemRepository();
var homeController = new HomeController(fakeTodoItemRepository);
var todoItem = new TodoItem()
{
    Id = 1,
    Title = "Chanter une chanson à grand-papa"
};

// Action
homeController.Create(todoItem);

// Assert
Assert.IsTrue(fakeTodoItemRepository.Added);
```

Fake repo

```
public class FakeTodoItemRepository : ITodoItemRepository

{
        public bool  TodoItemsAdded = false;
        private List<TodoItem> _todoItems;

        public FakeTodoItemRepository()
        {
            _todoItems = new List<TodoItem>();
        }
        ...

        public void Add(TodoItem todoItem)
        {
            _todoItems.Add(todoItem);
            todoItemAdded = true;
        }
}
```

Il existe plusieurs possibilités pour tester si un item à été ajouté dans le *repository*. Autres suggestions ?