ASP.NET MVCを使ったTDD入門
～SI屋さんとWEB屋さんとの違い～

http://twitter.com/normlian
http://d.hatena.co.jp/waritohutsu
http://www.pixiv.net/member.php?id=147209

自己紹介

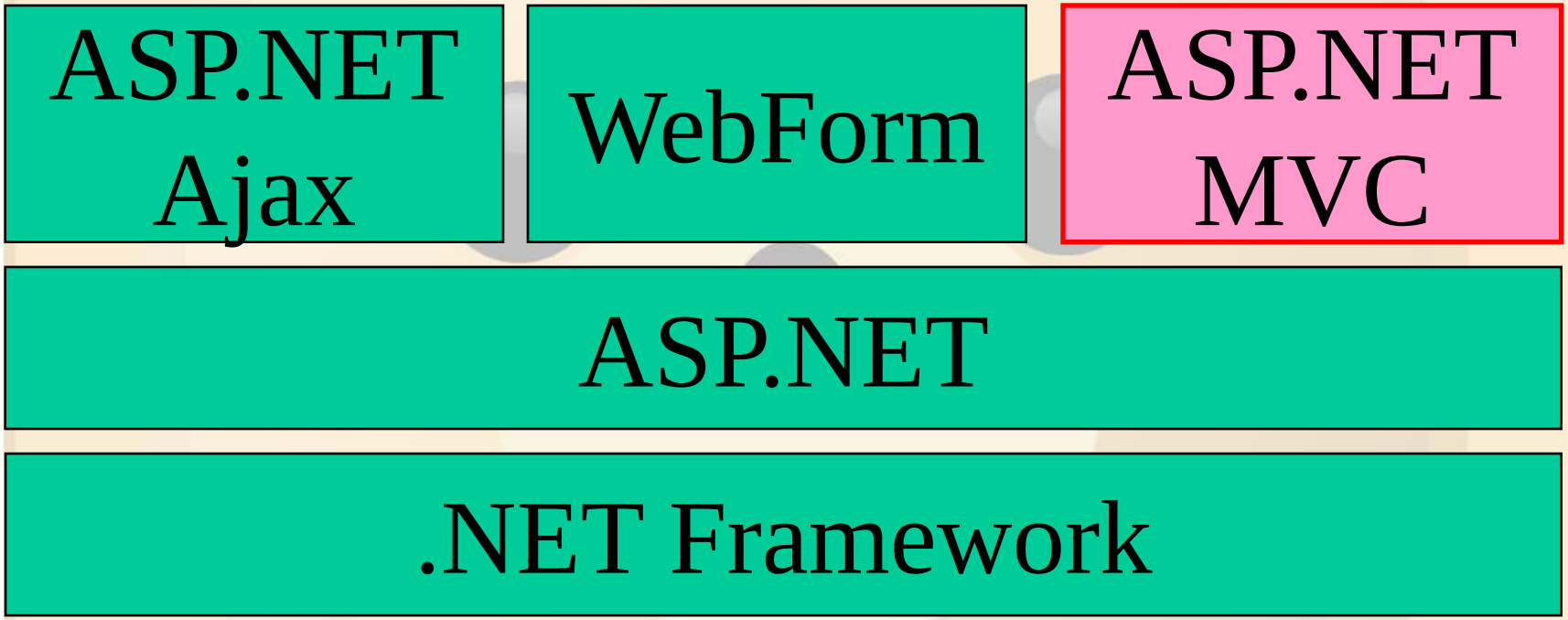
- HN　　：割と普通
- 本拠地　：横浜近辺
- 肩書き　：コードをあまり書けない SI屋
- 趣味　　：コードを書く＆絵を描く
- その他　：InfoQの翻訳者もどき

今日の流れ

- ASP.NET MVC って何者？
- ASP.NET MVC 誰がいつ使う？
- ASP.NET MVC をいじってみよう！
- ASP.NET MVC によるTDD開発

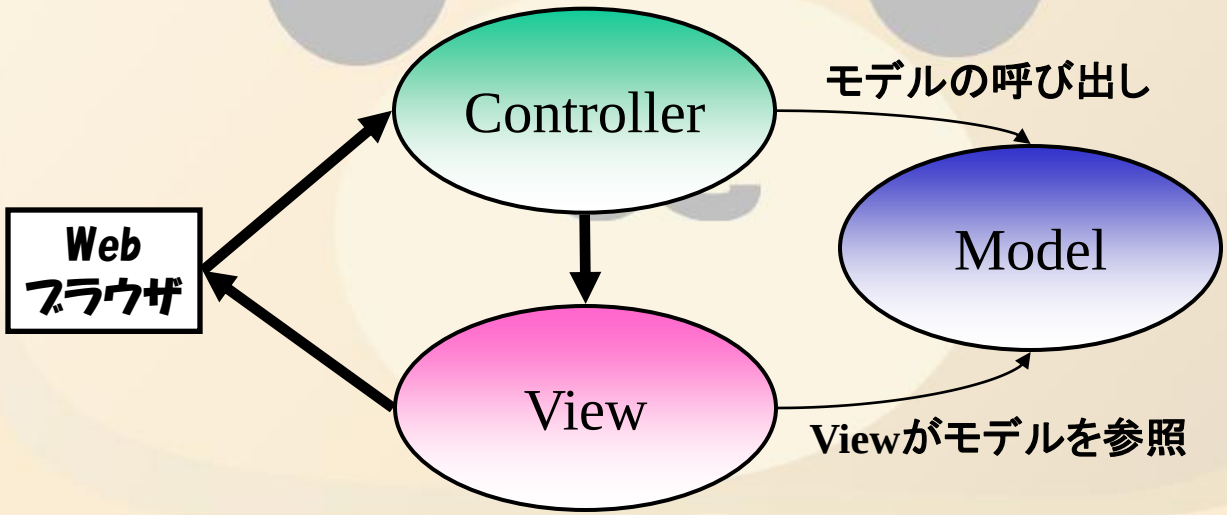
ASP .NET MVCって何者？

- **WEBアプリ開発用のフレームワーク**
  - **Codeplexでソースコードを公開中**



ASP .NET MVCって何者？（1/2）

- **ASP .NET で M･V･Cのモデルを構築**
  - **Model**
  - **View**
  - **Controller**



ASP .NET MVCって何者？（2/2）

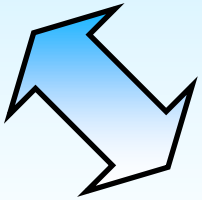
- 認証機能
- フィルタ機能
  - 時間があったらおまけで話します
- キャッシュ機能
- **単体テストの容易性**

今日の発表はここメインで♪

なんでASP .NET MVCつくったん？（1/3）

- ASP.NET自体のコンセプトは・・？

デスクトップアプリの開発手法
・WndProc、WM_XXX、イベントドリブン・・・



WEBアプリの開発
GET、POST、form・・・、セッション

従来のクラサバシステム開発者が、WEB開発にそのまま移行できる開発フレームワークを提供していた

なんでASP .NET MVCつくったん？（2/3）

- でも、それはそれで色々と問題が・・・

  - ViewStateの埋め込み何とかして・・・orz

```
<input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />↓
<input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />↓
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwUKMTY3NTUONDc3MWRk7daJLBzOTTLCcifUnhXeQC+aKwk=" />↓
```

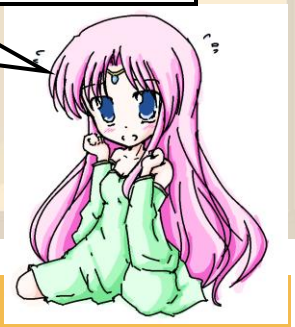
  - コードビハインドは良いけど、ユニットテスト化しにくいよ・・・orz
  - Javascript、cssの自動生成されて、他の部品とコンフリクトしますが何か？

なんでASP .NET MVCつくったん？(3/3)

- **純粋なWEB開発用フレームワークの台頭**
  - **Ruby on Rails (Ruby)**
  - **Django (Python)**
  - **Cake (PHP)**

  ⇒ TDD、WEB開発に特化
- 同じ型付言語のJavaでも色々と・・・
  - JSF、Struts、Spring、Wicket

従来のクラサバ開発者だけでなく、
WEB開発者も取り込んでみようかのー・・・かな？



今日の流れ

- ASP.NET MVC って何者？
- ASP.NET MVC 誰がいつ使う？
  - WEB屋さんとSI屋さんの違い
  - WebForm と MVC の住み分け
- ASP.NET MVC をいじってみよう！
- ASP.NET MVC でTDD開発

ASP .NET MVC 誰がいつ使う？

- 主な対象はいわゆるWEB屋さん
  - 気にするのは開発効率の向上、カスタマイズの容易性とか
- 設計メインなSI屋さんではない
  - 気にするのがコンポーネント化、標準化とか

- TDD開発、Agile開発等で強さを発揮しそう

※ 逆にウォーターフォールとかだと厳しそう

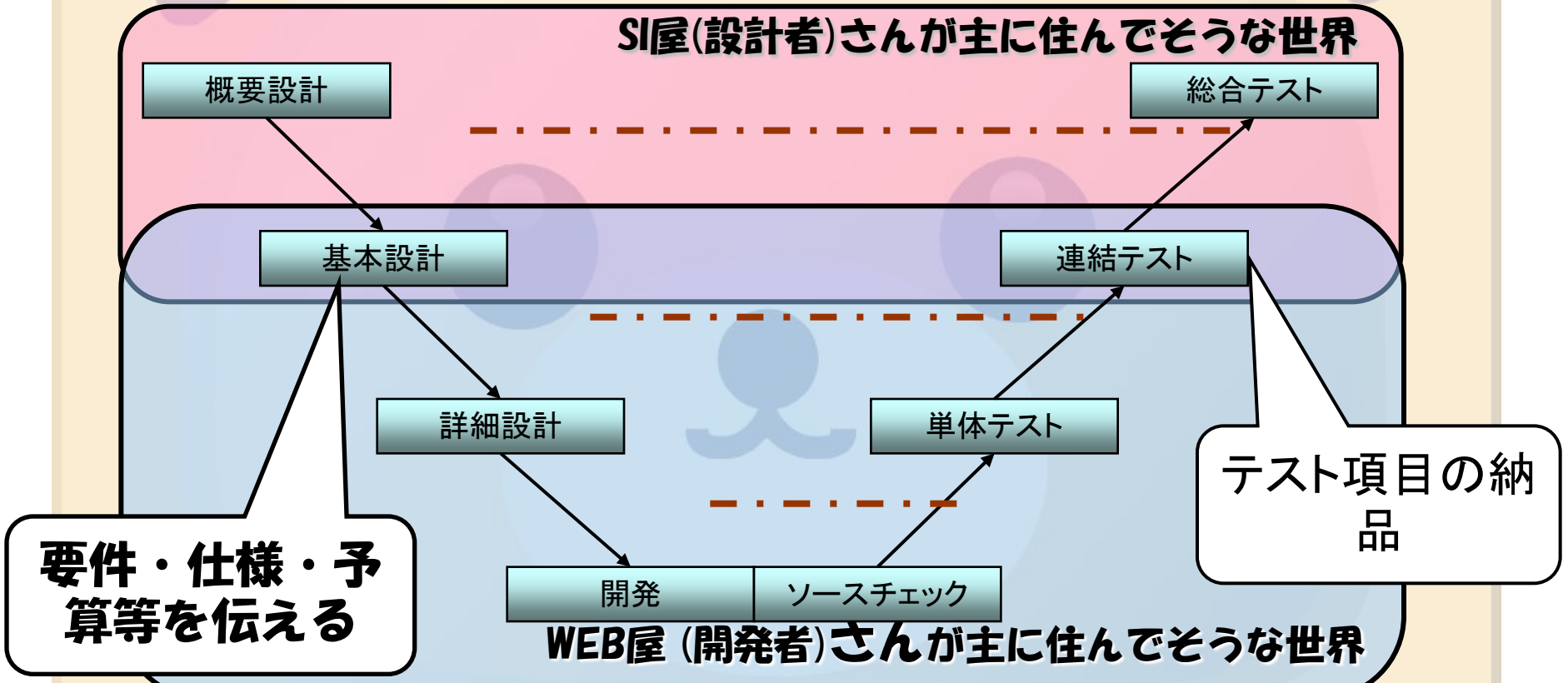
SI屋さんとWEB屋さんの違いってなにさ？(1/2)

- SI屋さん
  - 業務知識優先
  - 古き良きウォーターフォールベース
  - イントラ多し、典型的なCRUDアプリやら帳票やら
  - 特定ユーザがターゲット(特定の法人内とか)

- WEB屋さん
  - サービス企画、柔軟で高速な開発(ドキュメントは後から)
  - アジャイルが割かし多い筈じゃない？
  - URLやらXHTMLやら、デザインが超重要
  - ユーザが不特定多数多し(インターネット上)

いわゆるインターネットの“あちら側”と“こちら側”

SI屋さんとWEB屋さんの違いってなにさ？（2/2）



WEB屋さんとSI屋さんで気にするポイントが結構違う

わんくま同盟

WebFormとMVCのすみわけ

- SI屋さんの様なタイプ (コンポーネント指向)
  - 設計と開発との切り離しが容易
  - 典型的なCRUDアプリとかならコーディングレスでもOK
  - 基本設計とテスト項目の整合性チェックとかが楽
  - コンポーネントを用いた開発標準化が行いやすい

  → WebForm向き！！

- WEB屋さん的なタイプ(リソース指向)
  - Agile開発、TDD開発向き。
  - Ajaxなんかとの連携しやすいし、デザインにこりやすい
    - ポストバック、VIEWSTATE埋め込み、CSS生成はつらい
  - URLルーティングとか良い感じ
    - URL名もデザインできるよ

  → MVC向き！！

今日の流れ

- ASP.NET MVC って何者？
- ASP.NET MVC 誰がいつ使う？
- ASP.NET MVC いじってみよう！
  - Model
  - View
  - Controller
- ASP.NET MVC を使ったTDD開発

ASP.NET MVCを使ってみる ～概要 1/2～

- URLルーティングされ、実行されるControllerがマッピング
  - URLマッピングからコントローラ＆アクションを引き出す
  - リフレクションを使ってアクションを実行

```
public class MvcApplication : System.Web.HttpApplication
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapRoute(
            "Default",
            "{controller}/{action}/{id}",
            new { controller = "Home", action = "Index", id = "" }
        );

    }
```

ルーティングクラスを使ってマッピング

ASP.NET MVCを使ってみる ～概要 2/2～

- Controllerが実行され、画面にデータをマッピング
  - 以下は System.Web.Mvc.MvcHandler

```
protected internal virtual void ProcessRequest(HttpContextBase httpContext) {
    AddVersionHeader(httpContext);

    // Get the controller type
    string controllerName = RequestContext.RouteData.GetRequiredString("controller");

    // Instantiate the controller and call Execute
    IControllerFactory factory = ControllerBuilder.GetControllerFactory();
    IController controller = factory.CreateController(RequestContext, controllerName);
    if (controller == null) {
        throw new InvalidOperationException(
            String.Format(
                CultureInfo.CurrentUICulture,
                MvcResources.ControllerBuilder_FactoryReturnedNull,
                factory.GetType(),
                controllerName));
    }
    try {
        controller.Execute(RequestContext);
    }
    finally {
        factory.ReleaseController(controller);
    }
}
```

Controller 作成

Controller 実行

ASP.NET MVC使ってみる ～Controller 1/2～

- System.Web.Mvc.Controller で定義
- デフォルトでは、DefaultControllerFactory辺りからControllerが抜き出される
- Controllerのアクションを実行された際に、ActionResultクラスを返す

困ったらController.csクラスを眺めてみよう！

```
protected override void ExecuteCore() {
    TempData.Load(ControllerContext, TempDataProvider);

    try {
        string actionName = RouteData.GetRequiredString("action");
        if (!ActionInvoker.InvokeAction(ControllerContext, actionName)) {
            HandleUnknownAction(actionName);
        }
    }
    finally {
        TempData.Save(ControllerContext, TempDataProvider);
    }
}
```

ControllerActionInvoker.cs辺りを見るとControllerがどうやって実行されてるかわかるよん

ASP.NET MVC 使ってみる ～Controller 2/2～

- Controller辺りに余計な拡張をしようと思ったら・・
  - IControllerFactoryを実装したクラスに置き換えるといい感じ(現状ではDefaultControllerFactory辺りを継承したクラスでいじると良いかも)

```
public class QuillControllerFactory : DefaultControllerFactory
{
    QuillInjector injector = QuillInjector.GetInstance();

    override public IController CreateController(RequestContext requestContext, string controllerName)
    {
        Trace.WriteLine("RequestContext = " + requestContext.ToString() + ", controller = " + controllerName);
        IController controller = base.CreateController(requestContext, controllerName);
        injector.Inject(controller);
        return controller;
    }

    protected override IController GetControllerInstance(Type controllerType)
    {
        IController controller = base.GetControllerInstance(controllerType);
        injector.Inject(controller);
        return controller;
    }

    public override void ReleaseController(IController controller)
    {
        Trace.WriteLine("IController = " + controller.ToString());
        base.ReleaseController(controller);
    }
}
```

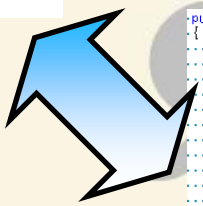
Controllerに対して、DIコンテナでインジェクションとか

ASP.NET MVCを使ってみる ～Model 1/2～

- System.Web.Mvc.IModelBinderを使ったりできるよ！
  - バインド用の独自インターフェースを使ってBinding
  - アクション実行の段階ではすでにモデルにバインディングされてる(ControllerActionInvoker内でバインド)！

```
[ModelBinder(typeof(HogeViewModelBinder))]
public class HogeViewModel
{
    public int Age { get; set; }
    public string Name { get; set; }
}
```

ViewModelクラス



```
public class HogeViewModelBinder : IModelBinder
{
    public object BindModel(ControllerContext controllerContext, ModelBindingContext bindingContext)
    {
        HogeViewModel model = new HogeViewModel();
        NameValueCollection collection = controllerContext.HttpContext.Request.Form;
        try
        {
            model.Age = int.Parse(collection["Age"]);
        }
        catch
        {
            bindingContext.ModelState.SetModelValue("Age", controllerContext.Controller.ValueProvider["Age"]);
            bindingContext.ModelState.AddModelError("Age", "数値を入力して下さい");
        }

        if (string.IsNullOrEmpty(collection["Name"]))
        {
            model.Name = @"";
            bindingContext.ModelState.SetModelValue("Name", controllerContext.Controller.ValueProvider["Name"]);
            bindingContext.ModelState.AddModelError("Name", "空文字は禁止です");
        }
        else
        {
            model.Name = collection["Name"];
        }

        return model;
    }
}
```

Binderクラス

ASP.NET MVCを使ってみる ～Model 2/2～

- Controller#UpdateModelメソッドを使うのも全然有り
  - Controller内で明示的にバインディングする
  - FormCollectionとModelsなクラスに対するマッピングをリフレクション使ってやってくれる

```
[AcceptVerbs(HttpVerbs.Post)]
public ActionResult Edit(string id, FormCollection collection)
{
    var book = _BookDatabaseEntities.Book.First(b => b.isbn == id);
    try
    {
        UpdateModel(book);
        _BookDatabaseEntities.SaveChanges();
        return RedirectToAction("Index");
    }
    catch
    {
        return View(book);
    }
}
```

「プロパティ名 == collection[“name属性”]」の値をマッピング

ASP.NET MVCを使ってみる ～View 1/2～

- Controller戻り値ActionResultの中で、ViewResultが*.aspxのレンダーに対応してる

```
public override void ExecuteResult(ControllerContext context) {
    if (context == null) {
        throw new ArgumentNullException("context");
    }
    if (String.IsNullOrEmpty(ViewName)) {
        ViewName = context.RouteData.GetRequiredString("action");
    }

    ViewEngineResult result = null;

    if (View == null) {
        result = FindView(context);
        View = result.View;
    }

    ViewContext viewContext = new ViewContext(context, View, ViewData, TempData);
    View.Render(viewContext, context.HttpContext.Response.Output);

    if (result != null) {
        result.ViewEngine.ReleaseView(context, View);
    }
}
protected override ViewEngineResult FindView(ControllerContext context) {
    ViewEngineResult result = ViewEngineCollection.FindView(context, ViewName, MasterName);
    if (result.View != null) {
        return result;
    }

    // we need to generate an exception containing all the locations we searched
    StringBuilder locationsText = new StringBuilder();
    foreach (string location in result.SearchedLocations) {
        locationsText.AppendLine();
        locationsText.Append(location);
    }
    throw new InvalidOperationException(String.Format(CultureInfo.CurrentUICulture,
        MvcResources.Common_ViewNotFound, ViewName, locationsText));
}
```

Viewの検索

Viewでレンダリング

ViewEngineCollectionからViewを検索

ASP.NET MVCを使ってみる ～View 2/2～

- IViewEngineの標準実装として用意されているのはWebFormViewEngine
- ~/Views仮想パス下から{controller}/{action}.aspxとかを探して、そのWebFormViewを作成

```
public class WebFormViewEngine : VirtualPathProviderViewEngine {

    private IBuildManager _buildManager;

    public WebFormViewEngine() {
        MasterLocationFormats = new[] {
            "~/Views/{1}/{0}.master",
            "~/Views/Shared/{0}.master"
        };

        ViewLocationFormats = new[] {
            "~/Views/{1}/{0}.aspx",
            "~/Views/{1}/{0}.ascx",
            "~/Views/Shared/{0}.aspx",
            "~/Views/Shared/{0}.ascx"
        };

        PartialViewLocationFormats = ViewLocationFormats;
    }
```

今日の流れ

- ASP.NET MVC って何者？
- ASP.NET MVC は誰がいつ使うのか？
- ASP.NET MVC をいじってみよう！
- ASP.NET MVC を使ったTDD開発

ASP.NET MVC上でのTDD開発(1/2)

- 開発サーバを起動することなくテスト可能
- ActionResultのModel、ViewDataに対してチェックを行う

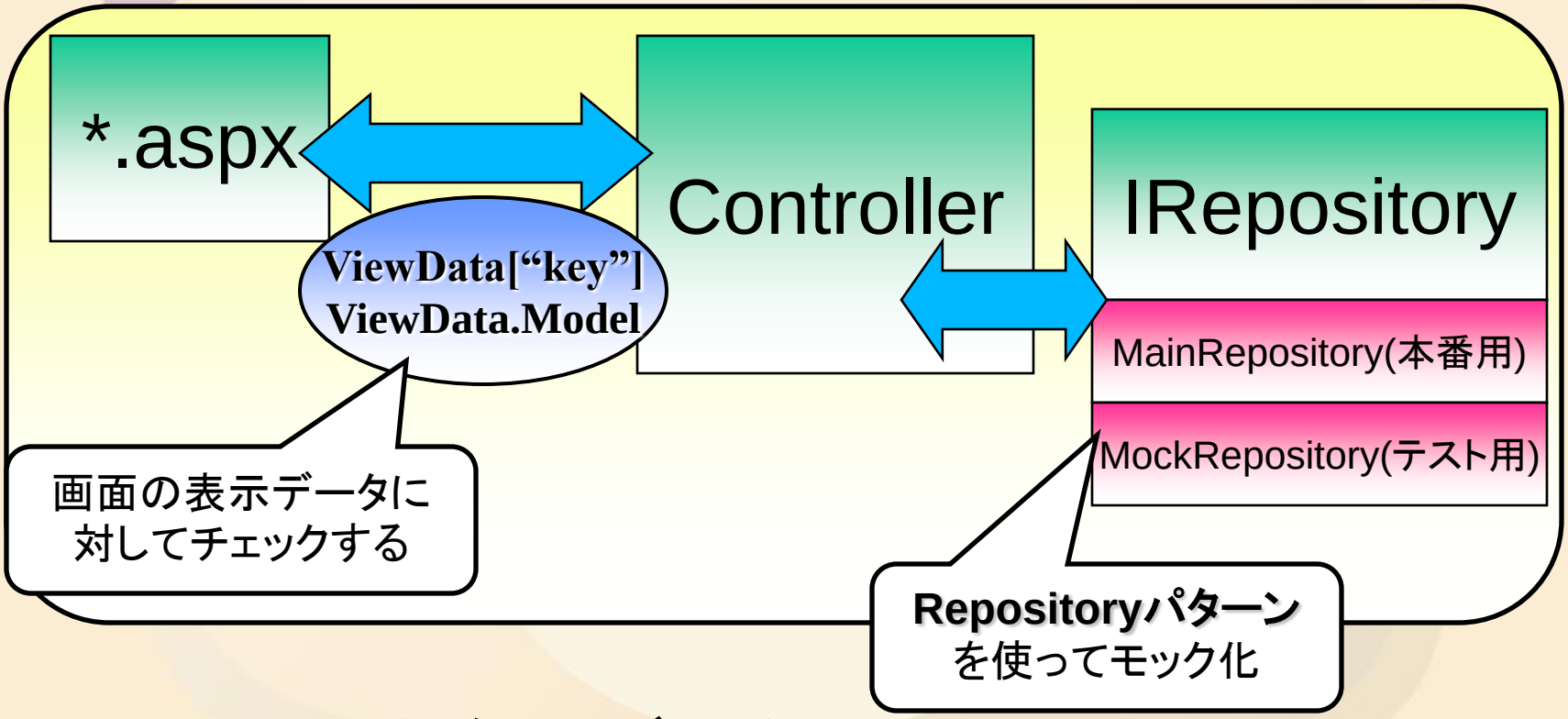
```
[TestMethod]
public void About()
{
    //Arrange
    HomeController controller = new HomeController();

    //Act
    ViewResult result = controller.About() as ViewResult;

    //Assert
    Assert.IsNotNull(result);
}
```

```
[TestMethod]
public void Index()
{
    //Arrange
    HomeController controller = new HomeController();

    //Act
    ViewResult result = controller.Index() as ViewResult;

    //Assert
    ViewDataDictionary viewData = result.ViewData;
    Assert.AreEqual("Welcome to ASP.NET MVC!", viewData["Message"]);
}
```

基本的にControllerの
引数・戻り値がテスト対象

ASP.NET MVC上でのTDD開発 (2/2)

- **単体テスト効率化の“コツ”**



実際にデモをしてみます！

おまけ その1　Filter開発

- **Controller実行時に適用されるフィルタ**

1. IAuthorizationFilter#OnAuthorization()
2. IActionFilter#OnActionExecuting()
3. コントローラのアクション実行
4. IActionFilter#OnActionExecuted()
5. IResultFilter#OnResultExecuting()
6. ActionResult#ExecuteResult()
7. IResultFilter#OnResultExecuted()

例外が発生

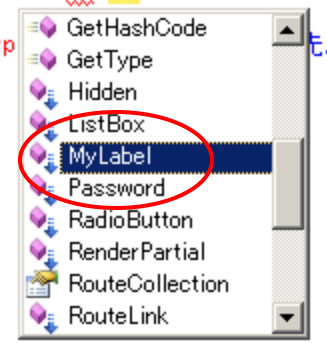
※IExceptionFilter#OnException()

おまけ その2　HtmlHelperの拡張

- *.aspx を作成する際に用いるヘルパークラス

```
[AspNetHostingPermission(System.Security.Permissions.SecurityAction.LinkDemand, Level =
AspNetHostingPermissionLevel.Minimal)]
public static class MyLabelExtensions
{
    public static string MyLabel(this HtmlHelper htmlHelper, string content)
    {
        return MyLabel(htmlHelper, content, @"noname");
    }

    public static string MyLabel(this HtmlHelper htmlHelper, string content, string name)
    {
        return MyLabel(htmlHelper, content, name, null);
    }
```

```
<% using (Html.BeginForm(@"                    ,@"Priority"
    { %>
<dl>
    <dt>優先順位</dt>
    <dd>
        <%= Html.TextBox(@"priority_name") %>
        <%= Html.my %>
    </dd>
    <input typ                                    先順位作成" />
</dl>
<% } %>


sp:Content>
```



やり過ぎるとWebFormと区別が付かなくなったりするので程々に

まとめ

- 純WEB屋さんならMVCが良い感じ
- 元々がデスクトップアプリ開発者なら、無理に変えなくても良いかも

- 拡張性が非常に高いのは素晴らしいよね
- CodePlexに行くと、ソースコードが落とせますよ
  - Futureパッケージ系は楽しそう(非同期実行とか)