Эффективная кроссплатформенная разработка .NET-приложений с использованием MVVM

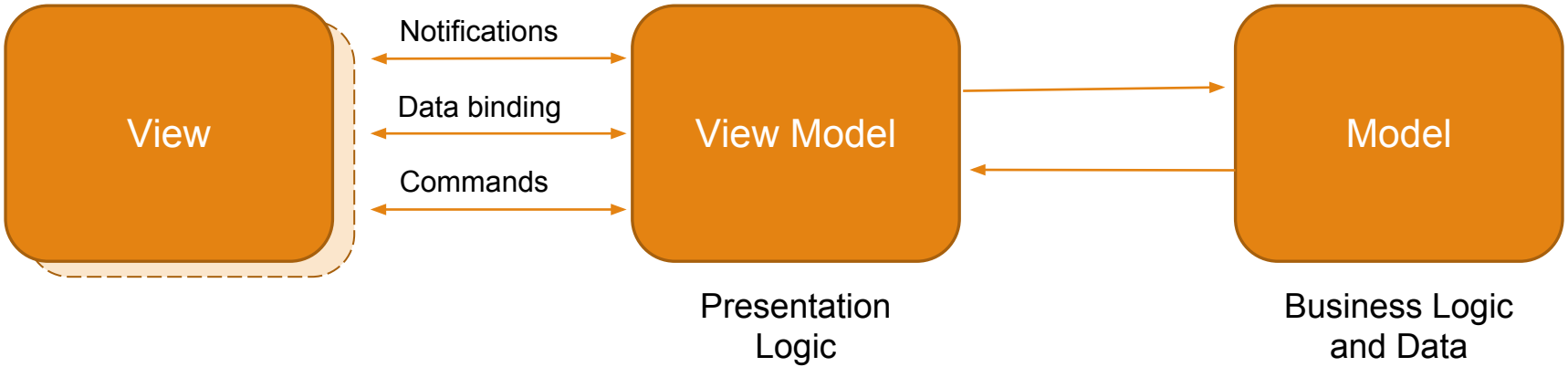
dmitry.garavsky@devexpress.com

MVVM? А зачем оно надо?

"When capabilities are extended, the codebase should become smaller".
@Thinking Out Loud

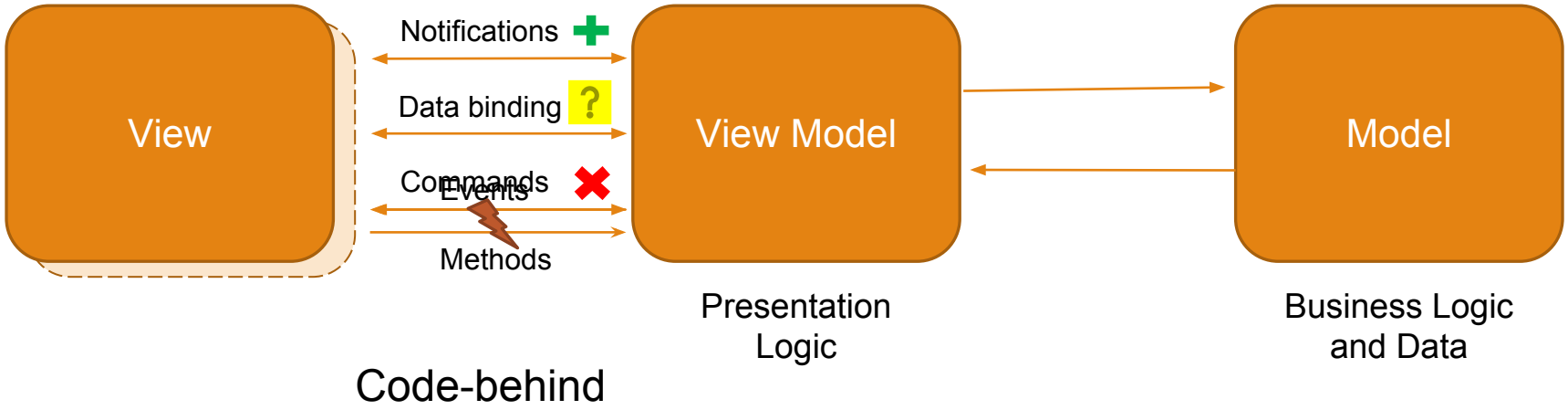
Четкое разделение бизнес-логики и логики представления.
Отсюда все вытекающие бенефиты и профиты.

MVVM. Типичная схема...



… и типичные проблемы:

А есть ли тут кроссплатформенность?



От MVVM к MVPVM. Presenter.

Выделяем узконаправленный код в специализированные классы

```
class AccountCollectionViewPresenter {
    public CollectionViewPresenter(GridView gridView, AccountCollectionViewModel viewModel) {
        gridView.FocusedRowObjectChanged += (s, e) => {
            viewModel.SelectedEntity = e.Row as Model.Account;
        };
        ((INotifyPropertyChanged)viewModel).PropertyChanged += (s, e) => {
            if(e.PropertyName == "SelectedEntity") {
                var entity = viewModel.SelectedEntity;
                if(entity != null)
                    gridView.FocusedRowHandle = gridView.LocateByValue("Id", entity.ID);
                else
                    gridView.FocusedRowHandle = GridControl.InvalidRowHandle;
            }
        };
    }
}
```

От MVVM к MVPVM. Presenter.

Выделяем узконаправленный код в специализированные классы

```
class AccountCollectionViewPresenter {
    public CollectionViewPresenter(GridView gridView, AccountCollectionViewModel viewModel) {
        gridView.FocusedRowObjectChanged += (s, e) => {
            viewModel.SelectedEntity = e.Row as Model.Account;
        };
                else
                    gridView.FocusedRowHandle = GridControl.InvalidRowHandle;
            }
        };
    }
}
```

Presenter повсюду.

- User Control и его Code-Behind
- Отдельный класс
- Отдельный метод для настройки контрола
- Специфичный кусок Code-Behind
- Обработчик события
- Привязка(Binding)

MVPVM без буквы “P”.

больше удобства, меньше кода

- Привязки к данным.
- Команды и привязки к командам.
- Поведения и сервисы.
- Бонус – удобный механизм реализации уведомлений, зависимостей, команд

Уведомления об изменении

```
public class LoginViewModel : INotifyPropertyChanged {
    public event PropertyChangedEventHandler PropertyChanged;
    string userNameCore;
    public string UserName {
        get { return userNameCore; }
        set {
            if(userNameCore == value) return;
            this.userNameCore = value;
            PropertyChangedEventHandler handler = PropertyChanged;
            if(handler != null)
                handler(this, new PropertyChangedEventArgs("UserName"));
        }
    }
}
```

дать возможность стороннему наблюдателю узнать об изменении значения свойства или состояния целевого объекта

Уведомления об изменении

```
public class LoginViewModel : INotifyPropertyChanged {
    public event PropertyChangedEventHandler PropertyChanged;
    string userNameCore;
    public string UserName {
```

```
public class LoginViewModel : BindableBase {
    string userNameCore;
    public string UserName {
        get { return userNameCore; }
        set { SetProperty(ref userNameCore, "UserName");}
    }
}
```

```
        }
    }
}
```

Уведомления об изменении

```
public class LoginViewModel : INotifyPropertyChanged {
    public event PropertyChangedEventHandler PropertyChanged;
    string userNameCore;
    public string UserName {
        }
    }
}
```

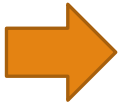
Plain **O**ld **C**lr **O**bject

POCO-трансформация


```
DevExpress.Mvvm.POCO.ViewModelSource
```

POCO-трансформация (INPC-проход)

POCO-class



Метаданные для генерации
- Свойства и методы
- Атрибуты свойств

Уведомления об изменении

```
public class LoginViewModel : INotifyPropertyChanged {
    public event PropertyChangedEventHandler PropertyChanged;
    string userNameCore;
    public string UserName {
```

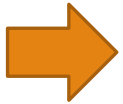
POCO-ViewModel:

```
public class LoginViewModel {
    public virtual string UserName { get; set; }
}
```

```
        }
    }
}
```

POCO-трансформация (INPC-проход)

Метаинформация

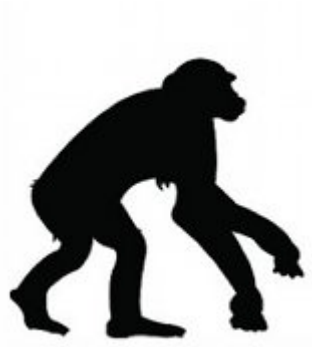


Сгенерированный тип-наследник

```
var viewModel = ViewModelSource.Create<ViewModel>();
```

Зависимости свойств

```
public class LoginViewModel : INotifyPropertyChanged {
    public event PropertyChangedEventHandler PropertyChanged;
    string userNameCore;
    public string UserName {
        get { return userNameCore; }
        set {
            if(userNameCore == value) return;
            this.userNameCore = value;
            PropertyChangedEventHandler handler = PropertyChanged;
            if(handler != null)
                handler(this, new PropertyChangedEventArgs("UserName"));
            OnUserNameChanged(); // OnUserNameChanged(oldValue)
        }
    }
}
```



дать возможность явно уведомить сторонних наблюдателей об изменении значения свойства или состояния целевого объекта

Зависимости свойств

```
public class LoginViewModel : INotifyPropertyChanged {
    public event PropertyChangedEventHandler PropertyChanged;
```

```
public class LoginViewModel : BindableBase {
    string userNameCore;
    public string UserName {
        get { return userNameCore; }
        set {
            SetProperty(ref userNameCore,
                "UserName", OnUserNameChanged);
        }
    }
}
```

```
}
```

Зависимости свойств

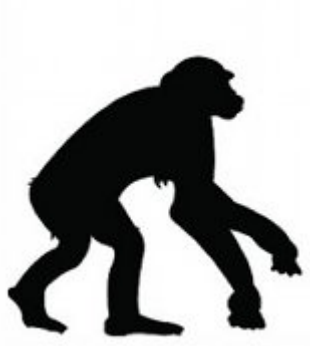
```
public class LoginViewModel : INotifyPropertyChanged {
    public event PropertyChangedEventHandler PropertyChanged;
    string userNameCore;
    public string UserName {
    }
}
```

POCO-ViewModel:

```
public class LoginViewModel {
    public virtual string UserName { get; set; }
    protected void OnUserNameChanged() {/*...*/}
}
```

Ручное обновление зависимостей

```
public class LoginViewModel : INotifyPropertyChanged {
    public event PropertyChangedEventHandler PropertyChanged;
    string userNameCore;
    public string UserName {
        get { return userNameCore; }
        set {
             if(userNameCore == value) return;
            this.userNameCore = value;
PropertyChangedEventHandler handler = PropertyChanged;
if(handler != null)
    handler(this, new PropertyChangedEventArgs("UserName"));
        }
    }
}
```



Ручное обновление зависимостей

```
public class LoginViewModel : INotifyPropertyChanged {
    public event PropertyChangedEventHandler PropertyChanged;
    string userNameCore;
```

POCO-ViewModel:

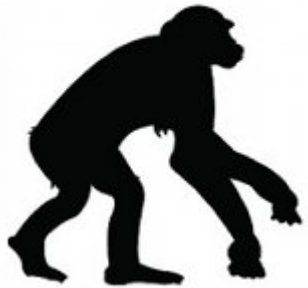
```
// Extension method
this.RaisePropertyChanged(x => x.UserName);
```

```
            }
        }
    }
}
```

Команды

```
public class DelegateCommand<T> : System.Windows.Input.ICommand {
public class MyViewModel1 {
    public MyViewModel() {
        this.SayHelloCommand = new DelegateCommand(SayHello);
    }
    public System.Windows.Input.ICommand SayHelloCommand {
        get;
        private set;
    }
    public void SayHello() { /* do something */ }
}
```

дать возможность инкапсулировать действие в отдельном объекте



Команды

POCO-ViewModel:

POCO-ViewModel:

```
public class MyViewModel {
    public Task DoSomething () {
        /* asynchronous !!! */
    }
}
```

Как это использовать?

Data-Bindings (XAML):

```
<dxe:TextEdit Text="{Binding UserName}"/>
```

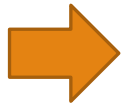
Commands (XAML):

```
<Button Command="{Binding SayHelloCommand}" />
<Button Command="{Binding SayCommand}"CommandParameter="Hi!"/>
```

Как написать простейший Event-trigger

Как написать простейший Event-trigger

Метаинформация



Сгенерированный метод-обработчик события

```
EventHandlerType eventHandler = (s,e) => OnEvent();
```

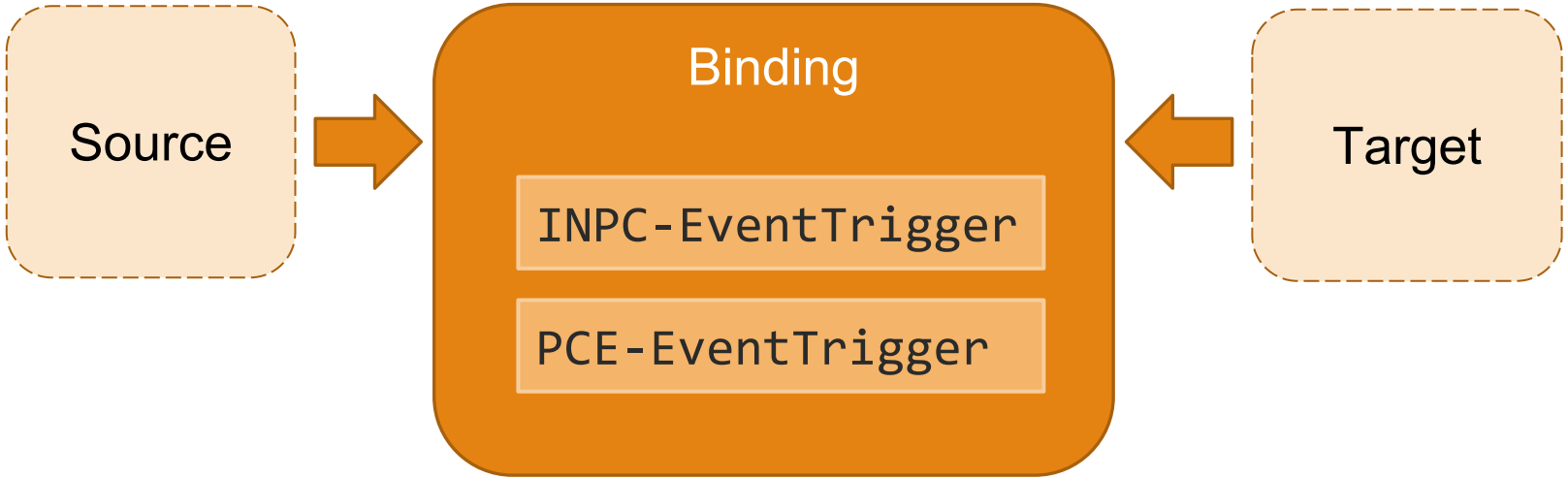
Как написать простейший Event-trigger

```
static Delegate GetHandler(
    Type handlerType,
    MethodCallExpression triggerExpression,
    ParameterExpression[] handlerParameters) {
    // Create&Compile Expression of specific type
    return Expression.Lambda(handlerType,
            triggerExpression,
            handlerParameters
        ).Compile();
}
```

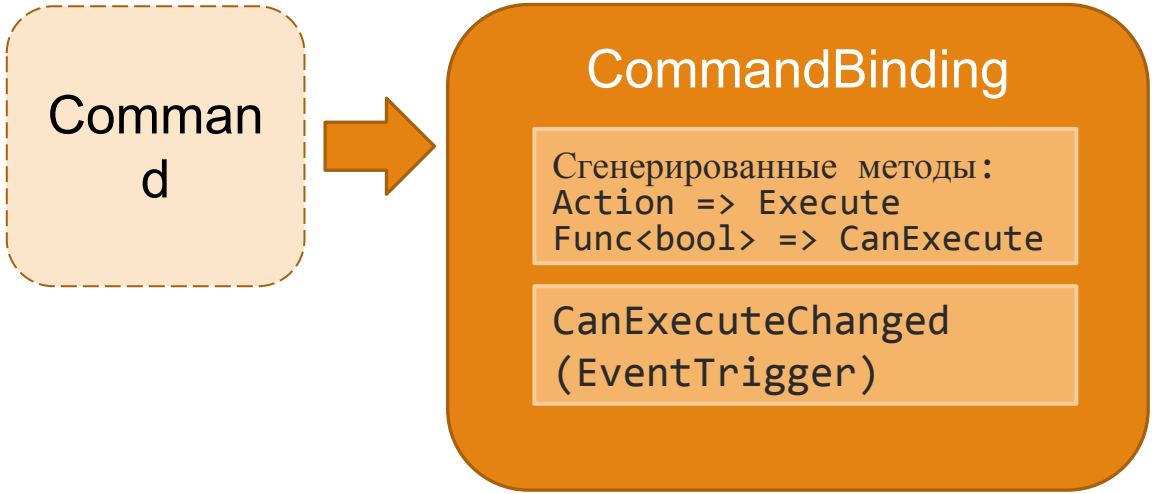
Как написать простейший Event-trigger

```
public class EventTrigger<TEventArgs> {
    protected TEventArgs Args {
        get;
    }
    protected virtual void OnEvent() {
        /* do something */
    }
}
```

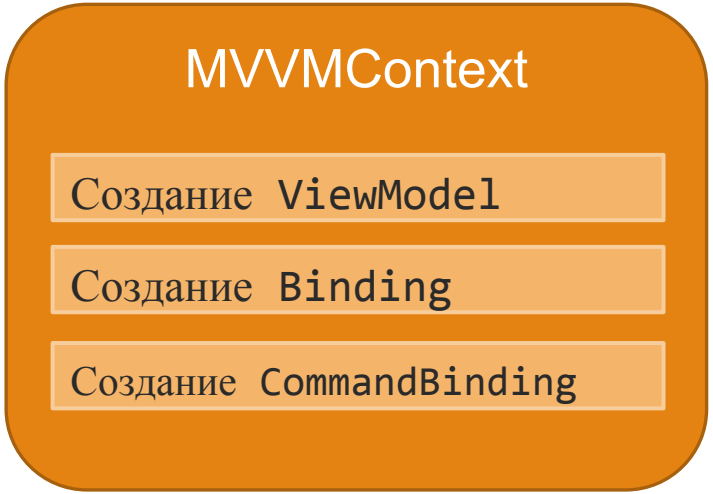
Как сделать простейший Binding.



Как сделать Binding к команде.



MVVMContext – собираем все в кучу



Как это использовать?

Code-behind(MVVM Context API):

```
public LoginViewModel ViewModel {
    get { return mvvmContext.GetViewModel<LoginViewModel>(); }
}
// ...
mvvmContext.SetBinding(tbUserName,
    t => t.Text, ViewModel, "UserName");
```

Code-behind(MVVM Context Fluent-API):

```
var fluent = mvvmContext.OfType<MyViewModel>();
fluent.SetBinding(tbTitle, t => t.Text, x => x.Title);
```

Как это использовать?

Code-behind (MVVMContext API):

```
public MyViewModel ViewModel {
    () => ViewModel.Say(null), ViewModel, () => ViewModel.Name);
```

Code-behind (MVVMContext Fluent API):

```
var fluent = mvvmContext.OfType<MyViewModel>();
fluent.BindCommand(btnSayHello, x => x.SayHello());
fluent.BindCommand(btnSay,(x,s) => x.Say(s), x => x.Name);
```

Теперь попробуем все это на практике

https://github.com/DmitryGaravsky/DevExpress.Mvvm.Examples

Интерактивность. Сервисы.

ViewModel:

```
public class MyViewModel {
    protected IMessageBoxService MessageBoxService {
        get { /*...*/ }
    }
    public void SayHello() {
        MessageBoxService.Show("Hello!");
    }
}
```

дать возможность взаимодействовать с пользователем не нарушая концепцию разделения слоев

Интерактивность. Сервисы.

POCO ViewModel:

```
protected IMessageBoxService MessageBoxService {
    get { this.GetService<IMessageBoxService>(); }
}

protected virtual IMessageBoxService MessageBoxService {
    get { throw new System.NotImplementedException(); }
}
```

Как это использовать?

Code-behind (MVVMContext API):

```
mvvmContext.RegisterService(new MessageBoxService());
//...
var mbService = mvvmContext.GetService<IMessageBoxService>();
mbService.Show("Something happens!");
```

Сервисы и проблема навигации.

- Независимость от View
- Поддержка разных типов UI
- Кроссплатформенность внутри одной платформы

Поведения.

дать возможность наделить объект дополнительным функционалом действуя снаружи объекта

```
public class ConfirmationBehavior
```

Code-behind (MVVMContext API):

```
//...
mvvmContext.AttachBehavior<ConfirmationBehavior>(this);
```

```
    protected override string GetConfirmationText() {
        return "Form will be closed. Are you sure?";
    }
}
```

Поведения.

Code-behind (MVVMContext Fluent API):

```
//...
mvvmContext.WithEvent<CancelEventArgs>(this, "Closing")
    .Confirmation(
      settings => {
        settings.Caption = "Closing Confirmation";
        settings.Text = "Form will be closed. Press OK to confirm.";
        settings.Buttons = ConfirmationButtons.OKCancel;
        settings.ShowQuestionIcon = false;
      });
```

Поведения. EventToCommand.

```
public class ClickToSayHello :
    EventToCommandBehavior<MyViewModel, EventArgs> {
    public ClickToSayHello()
     : base("Click", x => x.SayHello()) {
    }
}
```

Code-behind (MVVMContext API):

```
//...
mvvmContext.AttachBehavior<ClickToSayHello>(thirdPartyButton);
```

Поведения. EventToCommand.

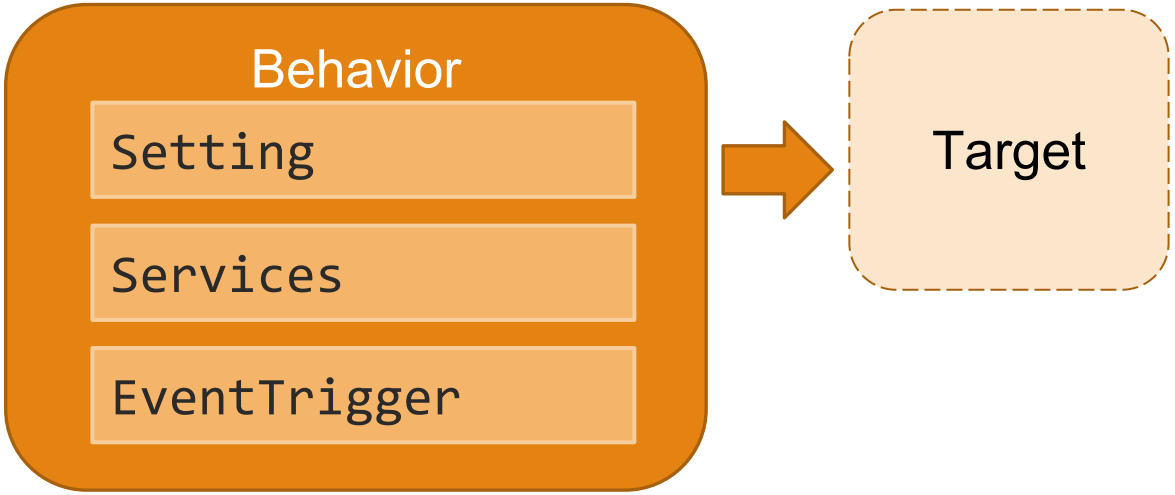
Code-behind (MVVMContext Fluent API):

```
mvvmContext.WithEvent<ViewModel, EventArgs>
    (thirdPartyButton, "Click")
    .EventToCommand(x => x.SayHello());
```

Code-behind (MVVMContext Fluent API):

```
fluent.WithEvent<RowClickEventArgs>(gridView, "RowClick")
   .EventToCommand(
       x => x.Edit(null), x => x.SelectedEntity,
       a => (a.Clicks==2) && (a.Button == MouseButtons.Left));
```

Как сделать Behavior?



Поведения и проблема INPC для Model.

Классы Model как правило не INPC

Code-behind (MVVMContext Fluent API):

```
var fluent = mvvmContext.OfType<AccountViewModel>();
fluent.SetObjectDataSourceBinding(
    bindingSource, x => x.Entity, x => x.Update());
```

Проблема Legacy кода. ViewModel.

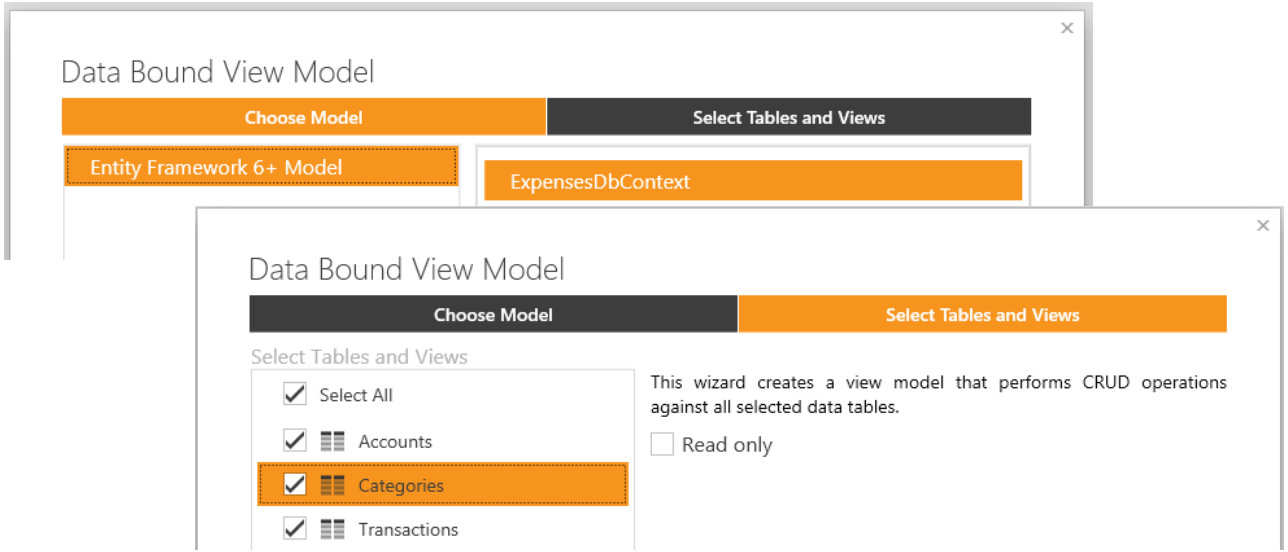
Хотим использовать все наработки

Code-behind (MVVMContext Fluent API):

```
var viewModel = new LegacyViewModel("Legacy ViewModel");
// initialize the MVVMContext with the specific ViewModel's
instance
mvvmContext.SetViewModel(typeof(LegacyViewModel), viewModel);
```

Проблема кроссплатформенного DataLayer

Подойдет ли нам Entity Framework?



Проблема кроссплатформенного GUI

Нет ответа. Есть мнение.

Подведем итоги

MVVM подход + MVVM.Utils

- Бесплатное и кроссплатформенное ядро (WPF/Silverlight/WinRT/UWP/WinForms + MONO)
- Вся мощь системы привязок на любой платформе
- Эффективность и уверенность в положительном результате

DevExpress®

Спасибо за внимание!

Гаравский Дмитрий
dmitry.garavsky@devexpress.com

WWW.DEVEXPRESS.COM

support@devexpress.com