Информатика

MVC

© Марченко Антон Александрович 2017 г.

Архитектура приложений

- Метафора архитектуры зданий
- Высокоуровневая структура системы
  - Структура, ответственности, интерфейсы, взаимодействия
  - Обладающая атрибутами качества
    - Надежность, отказоустойчивость, расширяемость, удобство использования, …

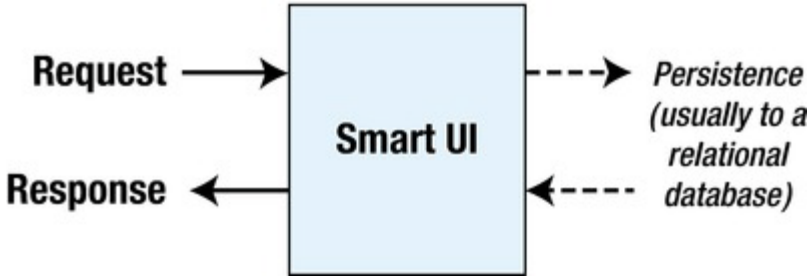
Паттерны

- Для удовлетворения атрибутам качества применяются архитектурные паттерны
- Примеры:
  - Layered pattern
    - Независимые модули, взаимодействие только со смежными
  - Broker pattern
    - Посредник (шина) для общения модулей
  - Client-Server pattern
    - Доступ большого числа клиентов к ограниченному числу ресурсов

Smart UI

- UI содержит всю логику и работает с данными напрямую
- Логика в обработчиках событий UI

Минусы?

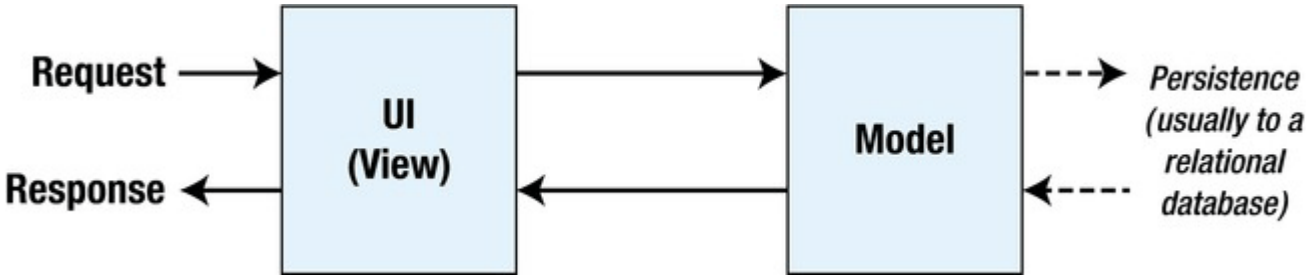


Бизнес логика

- Enterprise данные и правила работы с ними
- Поведение компоненты
- Реализация модели процессов реального мира в программном виде

Model-View

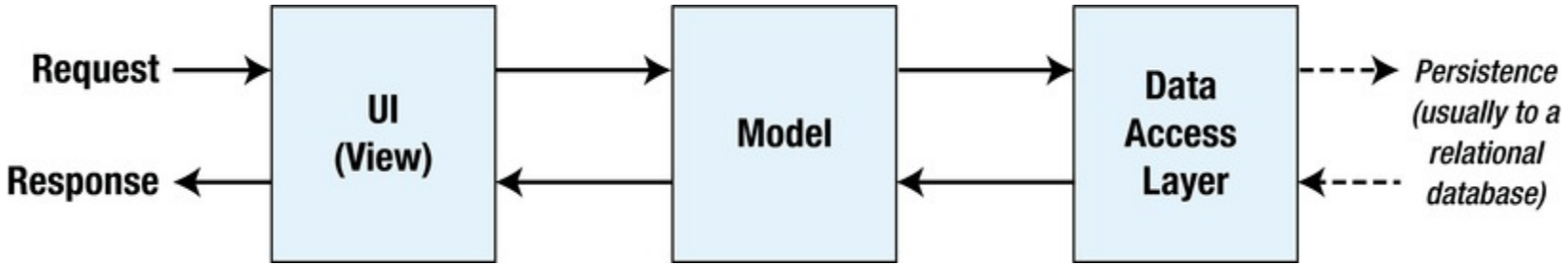
- Отделение модели предметной области от представления (UI)
  - данных и действий от отображения



Плюсы, минусы?

Three-tier

- Разделение UI, модели предметной области и доступа к данным



Плюсы, минусы?

MVC

- Будем говорить об архитектурных паттернах MVC
  - По сути, MV* (MVC) - парадигма
  - MVC, MVP, MVVM – паттерны
    - Отличие парадигмы от паттерна?

MVC до появления www

- Сформулирован в 1978-79 годах пионерами графических интерфейсов из Xerox PARC
  - Трюгве Реенскаугом с коллегами
  - Теми самыми, у кого заимствовала идеи Apple
- Тим Бернерс Ли заложил основы web позднее в 90-ых

Исследования UI в Xerox PARC

- Велись около 10 лет
- Аккумулировались решения, выделялись принципы, в том числе в области ООП и разработки больших/сложных систем
- В совокупности архитектурные идеи вылились Реенскаугом в MVC
- Получили первую значимую реализацию в Smalltalk80
  - Уже без участия Реенскауга

Развитие MVC

- Как часто бывает в информатике, работы не были доступны широкой публике
- Первая серьёзная публикация вышла спустя 10 лет после создания (88 год)
- Мартин Фаулер изучал MVC по работующей версии Smalltalk-80

Разногласия по поводу MVC

- Первоначальное отсутствие доступной информации породило мифы и искаженные трактовки MVC
- Многие считают его паттерном проектирования, хотя он является комбинацией нескольких паттернов
- Я называю его архитектурным паттерном
  - Понимая под этим набор архитектурных идей, принципов, которые можно реализовать по разному, используя разные паттерны проектирования

Цитата Фаулера об MVC

- *“MVC часто называют паттерном, но я не вижу особой пользы воспринимать его как паттерн, поскольку он включает в себя множество различных идей. Разные люди читают про MVC в различных источниках и извлекают от туда разные идеи, но называют их одинаково — «MVC». Это приводит к большой путанице и кроме того служит источником недоразумений и непониманию MVC, как будто бы люди узнавали про него через «испорченный телефон»…. Я уже потерял счет сколько раз я видел что-то, описываемое как MVC, которое им не оказывалось.”*

Инфа из первоисточников

- Доклад Реенскауга от 1979 года

- Работа Реенскауга MVC. Its past and present от 2003 года

Идеи MVC

- Разделить модель от представления (view, вид, визуальное представление)
- Модель независима ни от кого и ничего ни о ком не знает
- Представление отображает модель

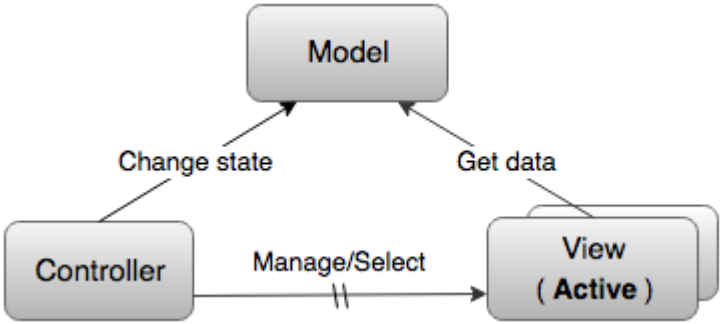
Варианты представления

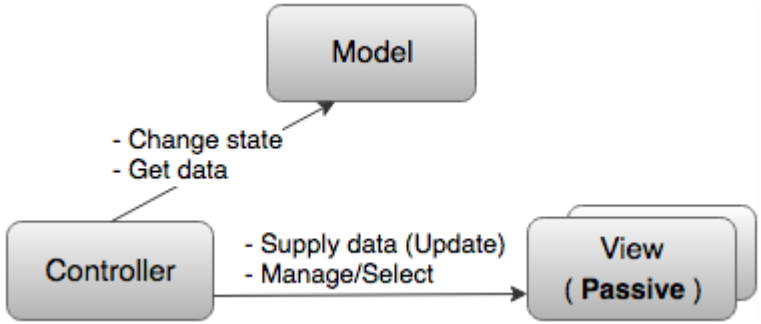
- **Активное** (Active view)
  - Знает о модели, самостоятельно обращается к ней за данными
- **Пассивное** (Passive view)
  - Не имеет прямой связи с моделью, общается через посредника – Контроллера (Controller)

Контроллер

- Самый неоднозначный компонент
- В любом случае знает о модели и может её изменять (в ответ на действия пользователя)
- Может управлять представлениями

Разновидности MVC

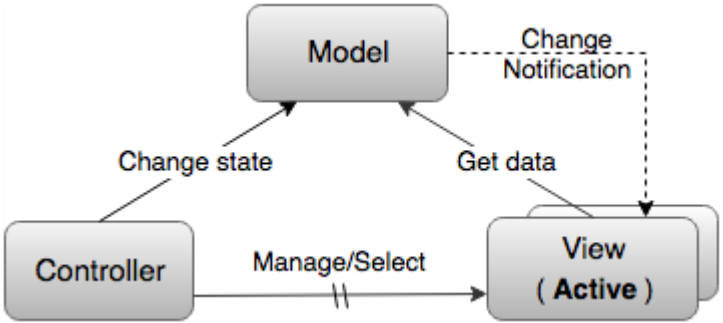


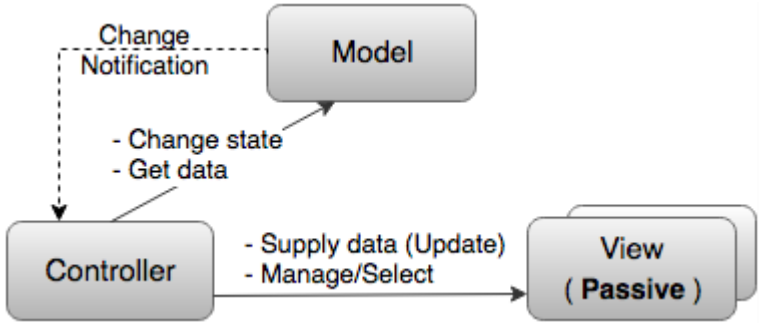


Варианты моделей

- Как и представление, модель может быть активной и пассивной
  - Активная – оповещает об изменениях подписчиков (активный вид или контроллер)
  - Паттерн Observer позволяет предоставить интерфейс подписки, оставаясь независимым от подписчиков (слабое связывание)

Более полные диаграммы MVC



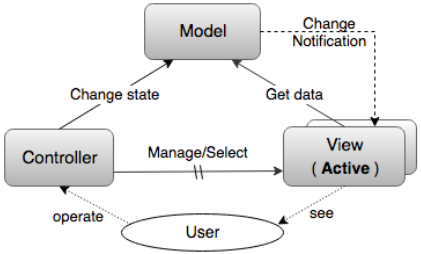


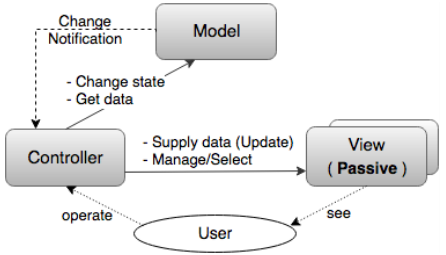
Другой взгляд на активность модели

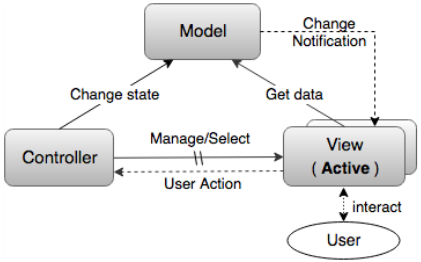
- Встречаются источники, в которых пассивная и активная модель называются в зависимости от включения в неё бизнес логики приложения
  - Чаще называют Тонкой и Толстой моделью

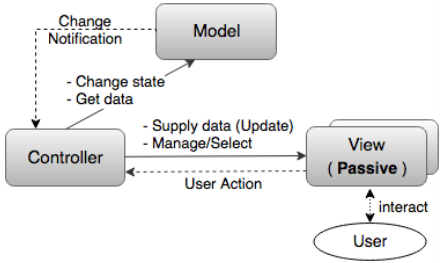
Пользователь

- Пользователь может взаимодействовать с приложением через
  - Контроллер
  - Представление









Неадекватный MVC из Web

- Три уровня
  - Клиент – сервер – база данных
- Модель – база данных
- Контроллер – сервер
- Представление – клиент (тонкий)

Почему неадекватный?

- В модели только данные, всё в контроллере. А ещё он от всех зависит.
  Обычно стараются наоборот минимизировать зависимые части
- А еще в контроллер пихают логику управления GUI
- См. *толстый тупой уродливый контроллер (ТТУК)*

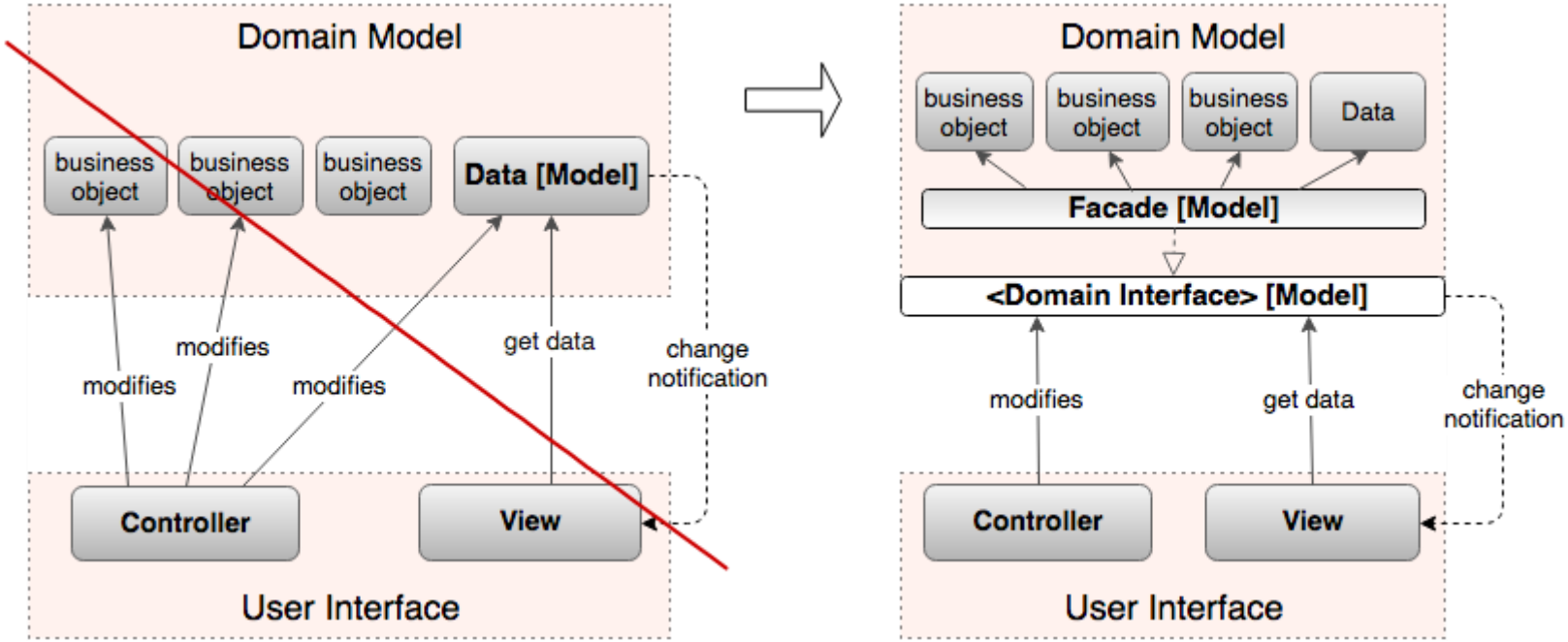
Нормальный MVC

- Отделение модели предметной области (доменной) от пользовательского интерфейса
  - Бизнес логика отдельно от UI
- Независимость модели (используя Observer, например)
- Пользовательский интерфейс – это представление и контроллер

Модель MVC != доменной модели

- Модель MVC – интерфейс и фасад для модели предметной области, которая может быть сколь угодно сложной и состоять из множества объектов
- Представление и контроллер взаимодействуют с интерфейсом и объектом-фасадом его реализующим

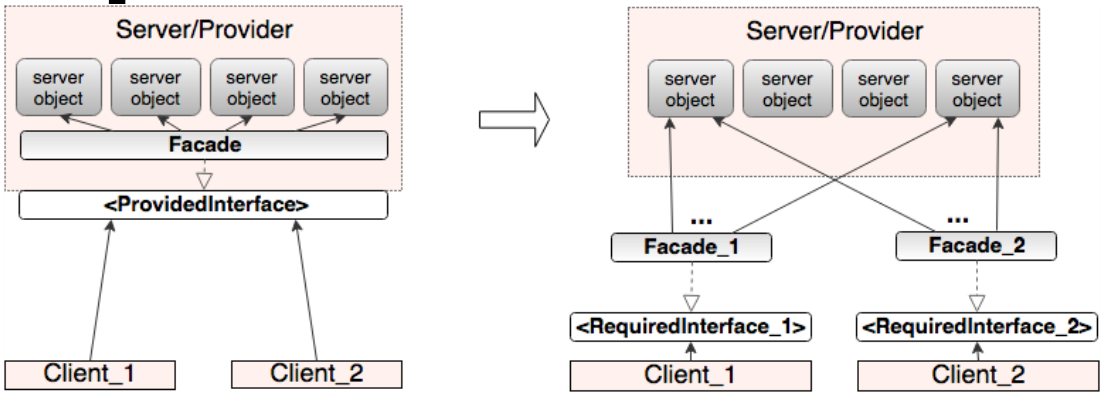
Еще одна диаграмма MVC

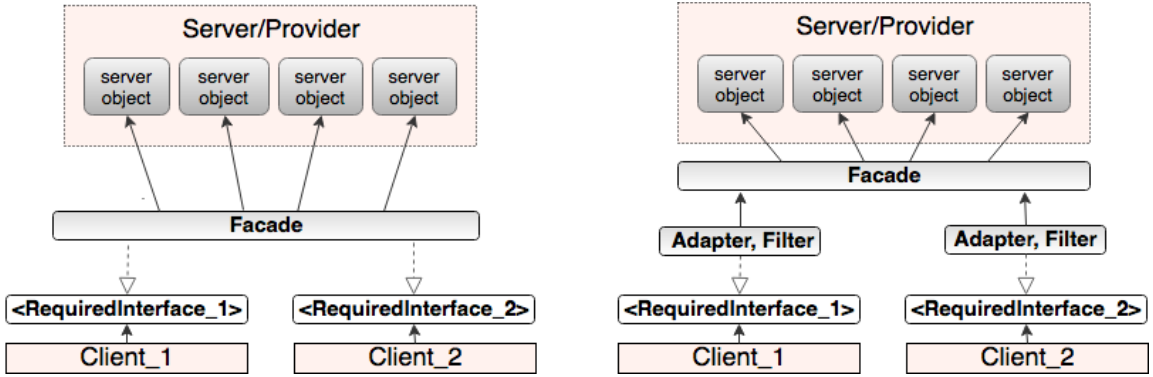


Client oriented VS Server oriented

- Сервер-ориентированная архитектура
  - Сервер-главный, предоставляет интерфейс, клиенты подстраиваются
- Клиент-ориентированная архитектура
  - Интерфейсы и фасад определяются из потребностей клиентов
  - Interface Segregation Principle

Диаграмма Server oriented





Архитектура микросервисов

- Вместо большого приложения со сложными внутрипроцессными взаимодействиями – несколько небольших приложений (возможно на разных серверах), соответствующих ограниченным контекстам, взаимодействующих по интерфейсам (HTTP, например)

Взаимодействие с микросервисами

- Шлюз (API Gateway) для взаимодействия с множеством микросервисов
- API Gateway предоставляет различные API для клиентов
  - См. API Gateway

Типичные ошибки

- Обращение к доменным объектам напрямую
  - Нужно интерпретировать и адаптировать данные, используя модели-посредники (содержащие ссылки)

Типичные ошибки

- Копирование доменных данных!
- Пример правильного использования – ValueHolder, который хранит ссылку на данные модели (а не сами данные)

Типичные ошибки

- Модель MVC – доменная модель и данные
- А должна быть интерфейсом и реализующим её объектом (фасадом, адаптером, заместителем)
- Нужно реализовывать удобный и безопасный доступ к данным

MVP

- MVC с пассивным представлением, когда оно напрямую не связано с моделью называют MVP
  - Больше соответствует web-прилоежениям
- Model – View – Presenter
  Модель – Представление - Представитель

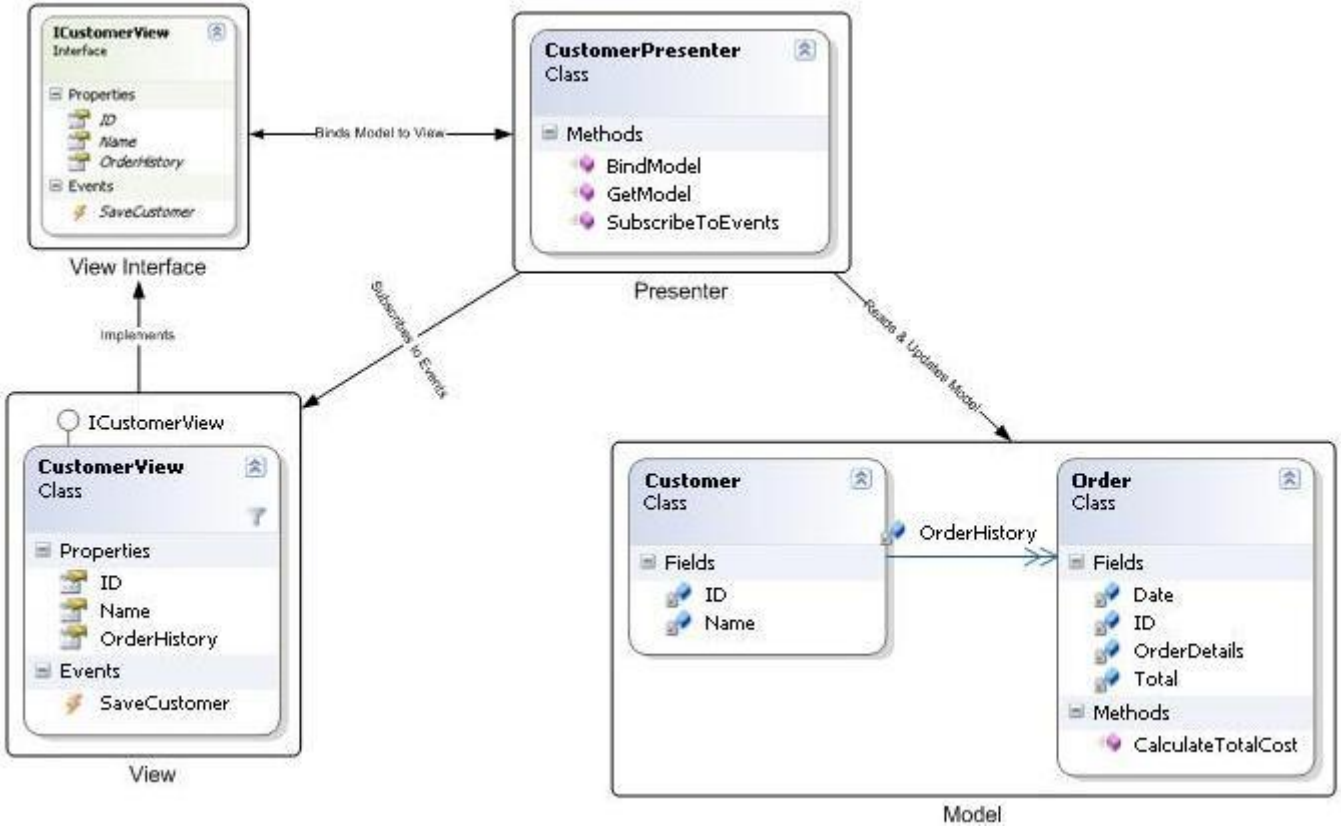
Разновидности MVP

- **Passive View**
  - не знает о модели, работает через представителя
  - Легковесное
  - Работает только с примитивными данными
- **Presentation Model**
  - Может обрабатывать объекты бизнес логики
- **Supervising Controller**
  - Использует binding для связывания представления с моделью

Трактовки названия

- Представитель в MVP
  - Представляет действия пользователя бэкенду
  - Представляет ответ пользователю
  - Посредник
- Контроллер
  - Не является посредником
  - Не обновляет представление
  - Отделяет действия пользователя и модель

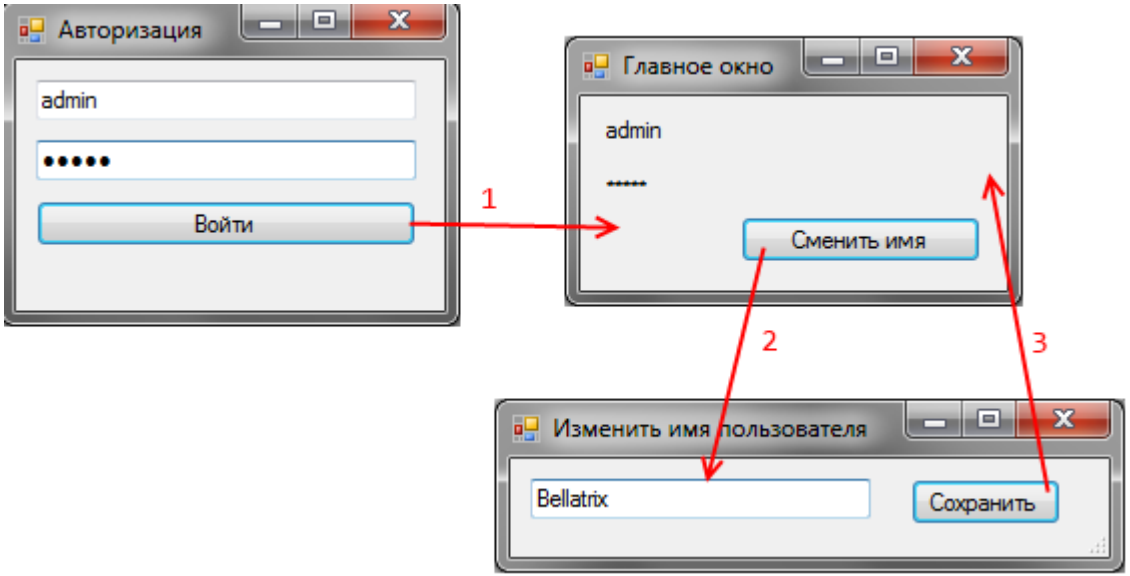
MVP


ICustomerView
Interface
Properties
ID
Name
OrderHistory
Events
SaveCustomer
View Interface
Binds Model to View
CustomerPresenter
Class
Methods
BindModel
GetModel
SubscribeToEvents
Presenter
Implements
Subscribes to Events
Reads & Updates Model
ICustomerView
CustomerView
Class
Properties
ID
Name
OrderHistory
Events
SaveCustomer
View
Customer
Class
Fields
ID
Name
OrderHistory
Order
Class
Fields
Date
ID
OrderDetails
Total
Methods
CalculateTotalCost
Model

MVP в WindowsForms

Пример из статьи https://habrahabr.ru/post/211899/

Passive View



Проекты в решении

- Отдельные сборки
  - Модель
  - Представителей
  - UI (формы)
  - Тестов

Что есть

- Интерфейсы представления, модели
- Представители
- Тесты для представителя
- Тесты для модели
- Формочки (UI), реализующие интерфейс представления

Работа приложения

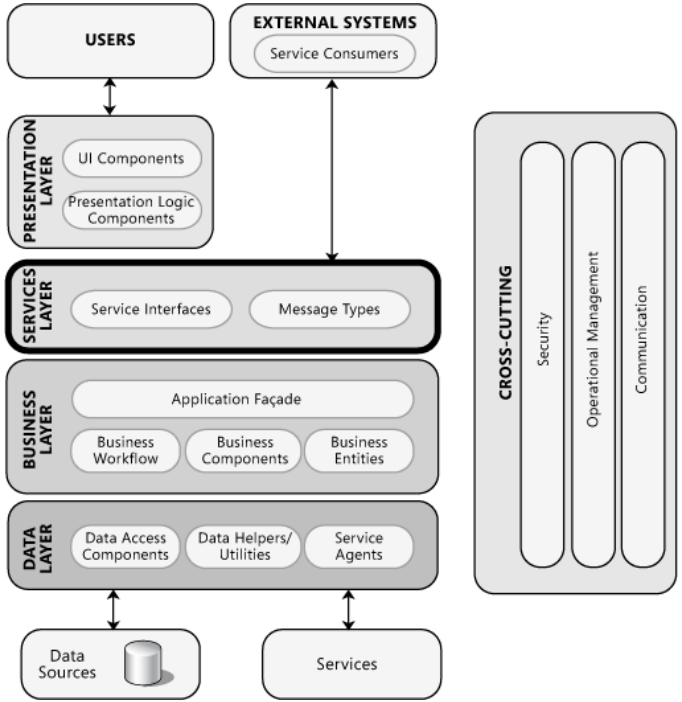
- В контроллере ссылки представления и модели (композиция или агрегация)
- Запуск через контроллер
- При закрытии окна контроллер запускает следующее окно (переключает ApplicationContext)
- Можно отображать несколько страниц на окне и вместо закрытия окна, открывать в текущем другую страницу

MVVM

- Есть еще вариант MVC с «умными» элементами управления в представлении, которые сами реагируют на уведомления и посылают команды в модель
  - Data & command binding
    - 2005, Microsoft: WPF, Silverlight
- Поэтому в MVVM нет ‘C‘ (посредника-контроллера)

Почитать

- http://rsdn.org/article/patterns/ModelViewPresenter.xml
- https://habrahabr.ru/post/321050/
- https://habrahabr.ru/post/211899/
- http://www.martinfowler.com/eaaDev/SupervisingPresenter.html
- http://www.martinfowler.com/eaaDev/PassiveScreen.html
- http://www.martinfowler.com/eaaDev/PresentationModel.html



Вопросы?

e-mail: marchenko@it.kfu.ru