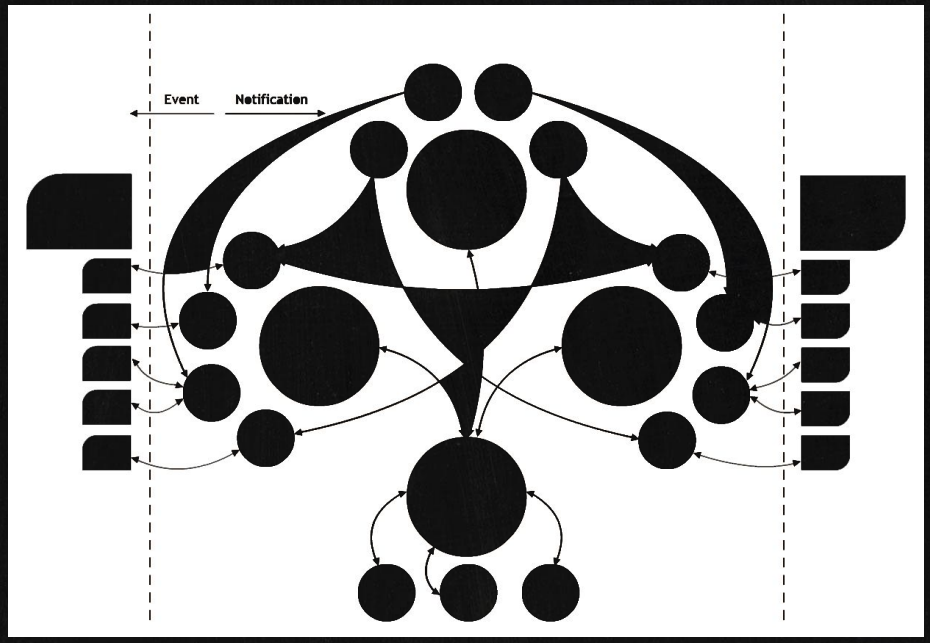
PureMVC в картинках



- для ленивых –

Ростислав Сирык
http://flash-ripper.com/

Способы разработки приложений

- Хотелось бы:
  - **«Раз!» - «работка»**
- Есть:
  - **«Раз!» -** «...а получилось как обычно»
- Должно быть:
  - **1. Раз-Ра-Бот-Ка**
    («по-э-тап-но»)

PureMVC для ленивых

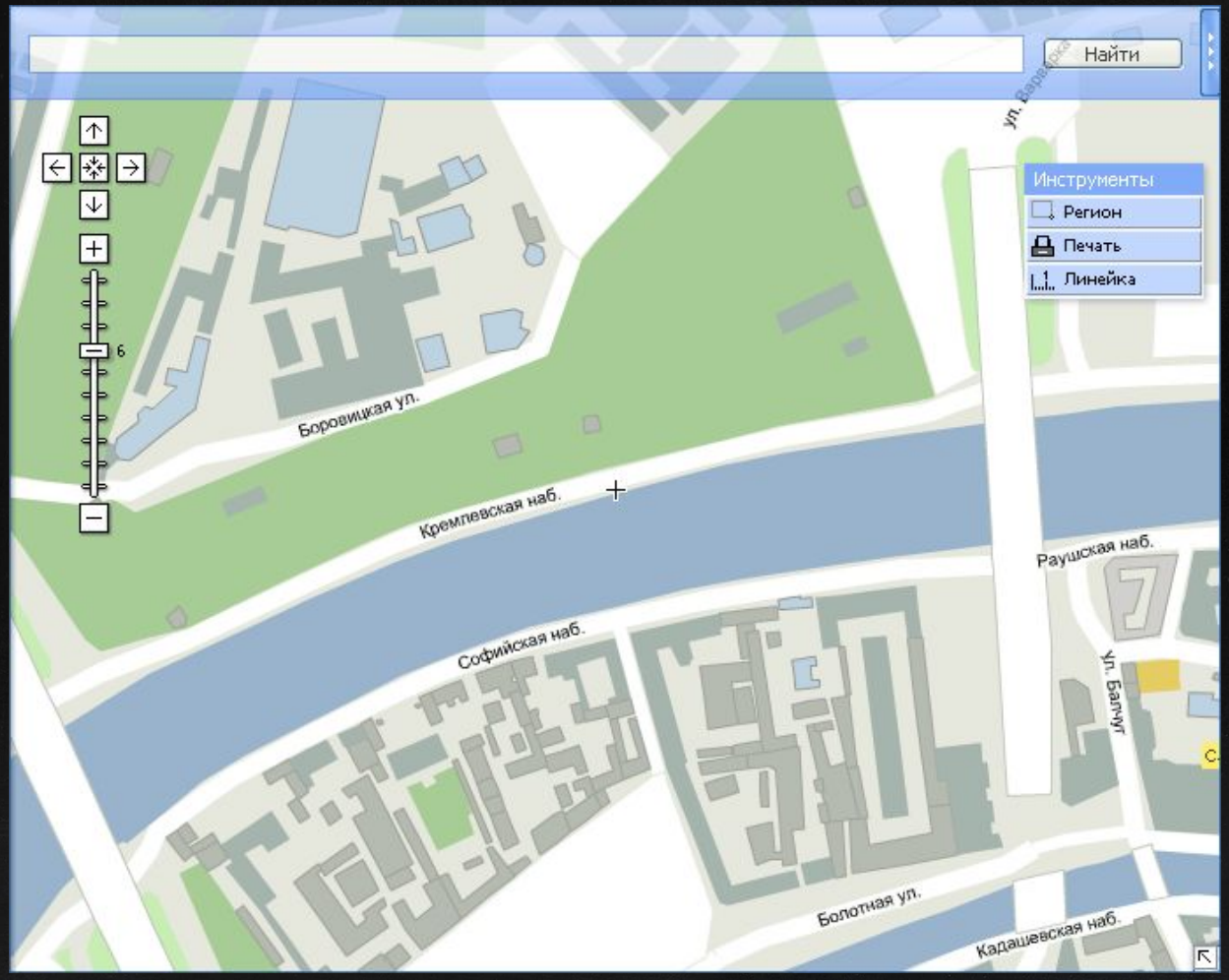
- Вопрос: Зачем нужен PureMVC?
- Ответ: Для экономии лени.
  1. Лень – это друг человека.
  2. Количество лени во Вселенной ограничено и неизменно ;-(
  3. Лень не берется из ниоткуда не исчезает в никуда.
  4. **Слишком умные отбирают лень у просто умных**
- PureMVC — оружие «слишком» умных.

Векторная карта Москвы

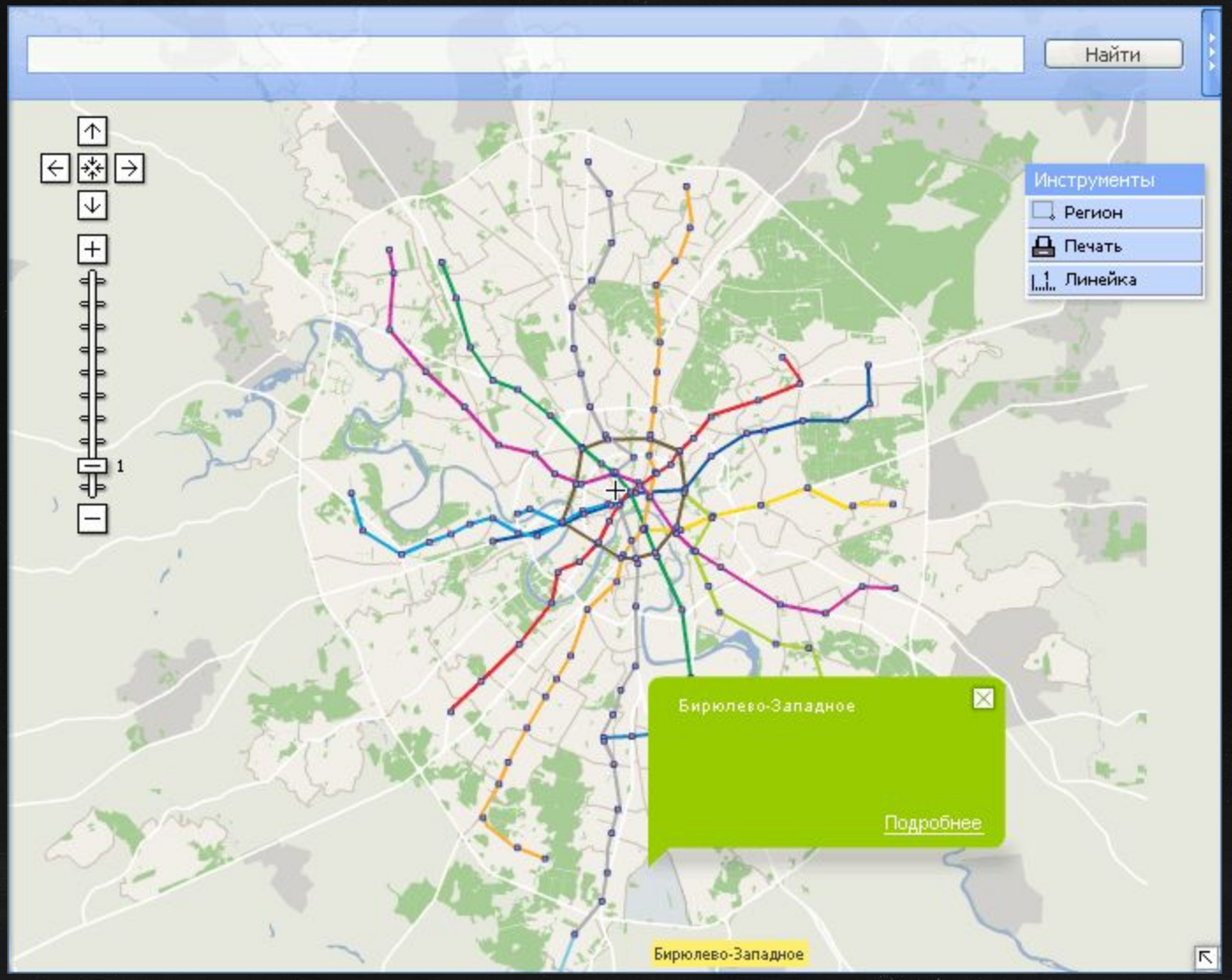
- пример PureMVC-приложения -

Векторная карта как дерево

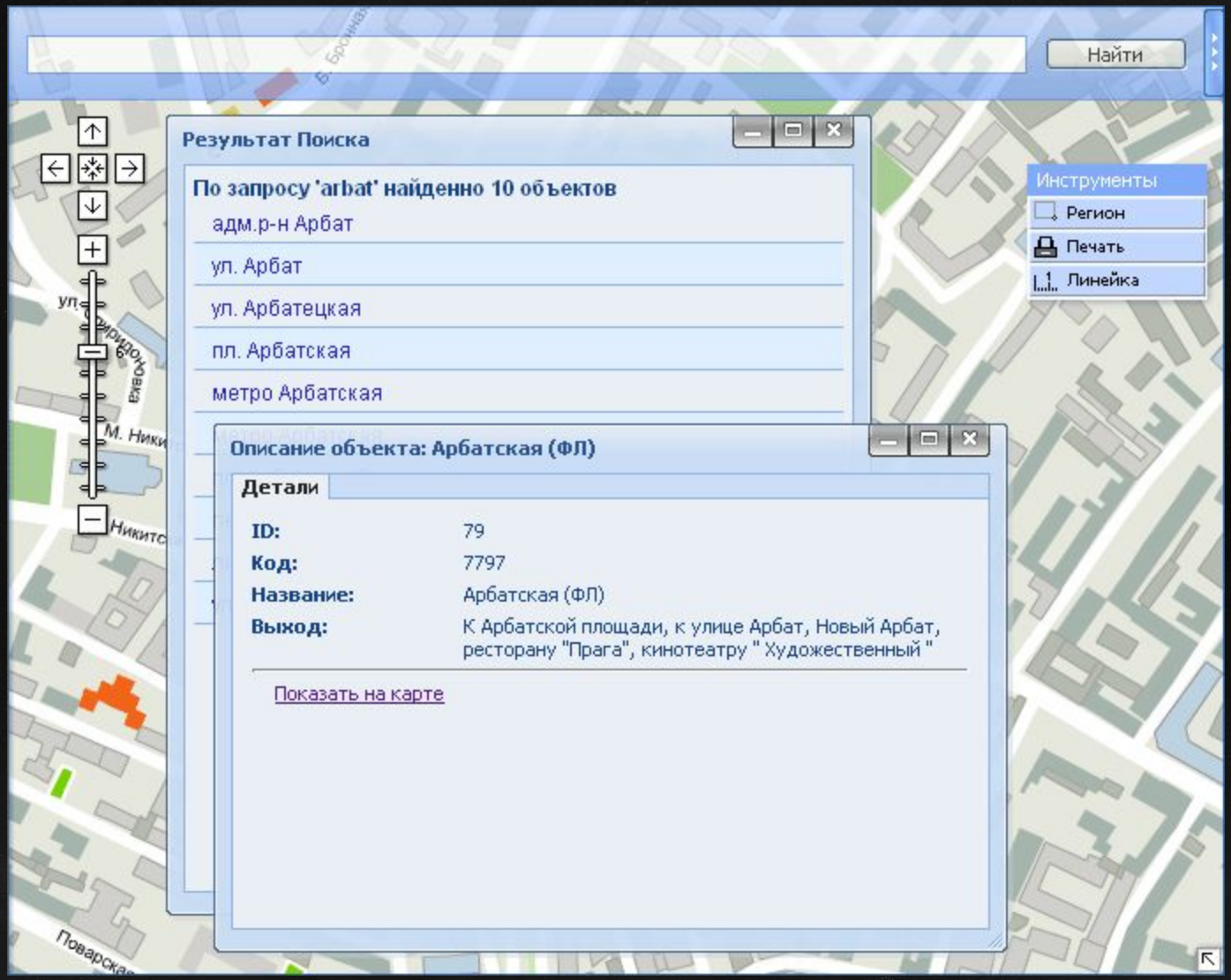
На вид двумерная, но внутри четыре



Общий вид карты



Работа карты с браузером



Требования к карте

- Карта Москвы с точностью до метра
- 9 уровней масштаба
- Гибкая настройка стиля и поведения
- Инструменты карты
- «Чтобы работала лучше Mos2.ru»
- Рабочая версия http://217.10.32.73:8080/map/

Большая переделка

- рефакторинг -

Если вы попали в серьезную переделку

Рефакторинг или **Реорганизация** — процесс полного или частичного преобразования внутренней структуры программы при сохранении её внешнего поведения.

http://ru.wikipedia.org/wiki/Рефакторинг

Подходы к рефакторингу

- Наивный (нет подхода):
  - как-нибудь да получится («какой-какой «рефакторинг?»)
- Самоуверенный:
  - вера в идеальный код, не нуждающийся в переделке
- Эгоистичный:
  - пусть рефакторит компьютер, он железный
- Осознанный:
  - делаем с учетом будущего, придерживаемся правил
- Стратегический:
  - применение паттернов, проектирование, фреймворки.

Методы рефакторинга

- **Изменение сигнатуры метода**
  - Заключается в добавлении, изменении или удалении параметра метода.
- **Инкапсуляция поля**
  - Было: public var x: Number;
  - Стало: private var _x: Number; // добавляется getter-setter
- **Выделение метода**
  - Самокомментирующийся код: если фрагмент кода требует комментария, то его следует выделить в отдельный метод и назвать так, чтобы данный комментарий стал ненужным.
- **Перемещение метода**
  - Перемещается метод, который чаще обращается к другому классу, чем к своему собственному.

Проблемы от рефакторинга

- Проблемы, связанные с базами данных
- Проблемы изменения интерфейсов
- Трудности при изменении дизайна

PureMVC

- и паттерны проектирования -

Что такое PureMVC

- PureMVC - это классический мета-паттерн «Model-View-Controller»
- **Proxies** = Модель
- **Mediator** = Представление
- **Commands** = Контроллер
  - + **Notifications** (Оповещения) для коммуникации.
  - + **Façade** (Фасад) для координирования

Шутка про k=3

- Математик идет по улице.
  Видит — филармония, афиша,
  "Камерный оркестр Джо Пауэлла".
  О, говорит, интересно, зайду.
  Через полчаса растроенный выходит:
  "Тьфу, тривиальный случай, **k=3**"
- Математик что-то знал (**Паттерны?**).

Схема PureMVC

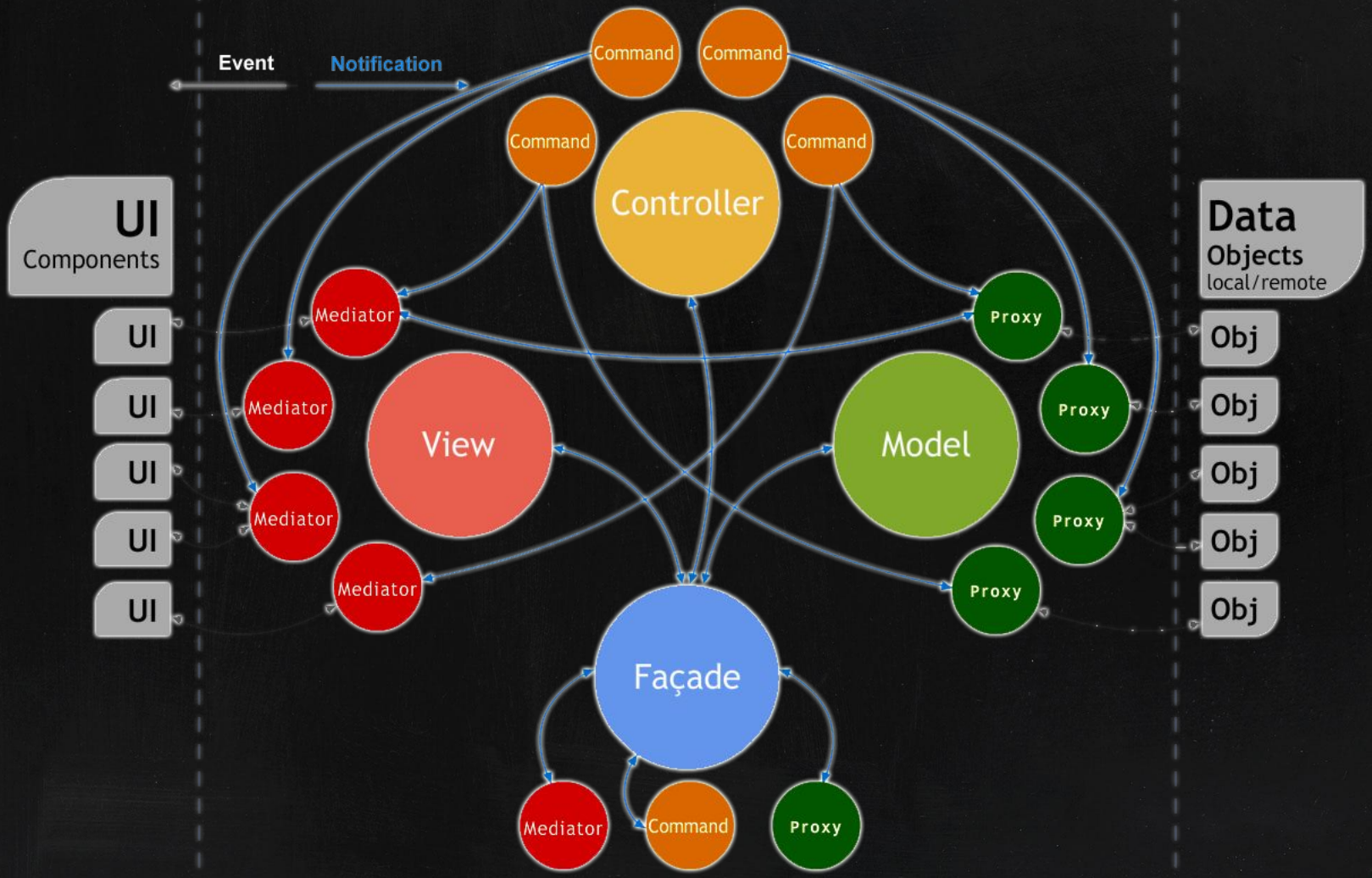
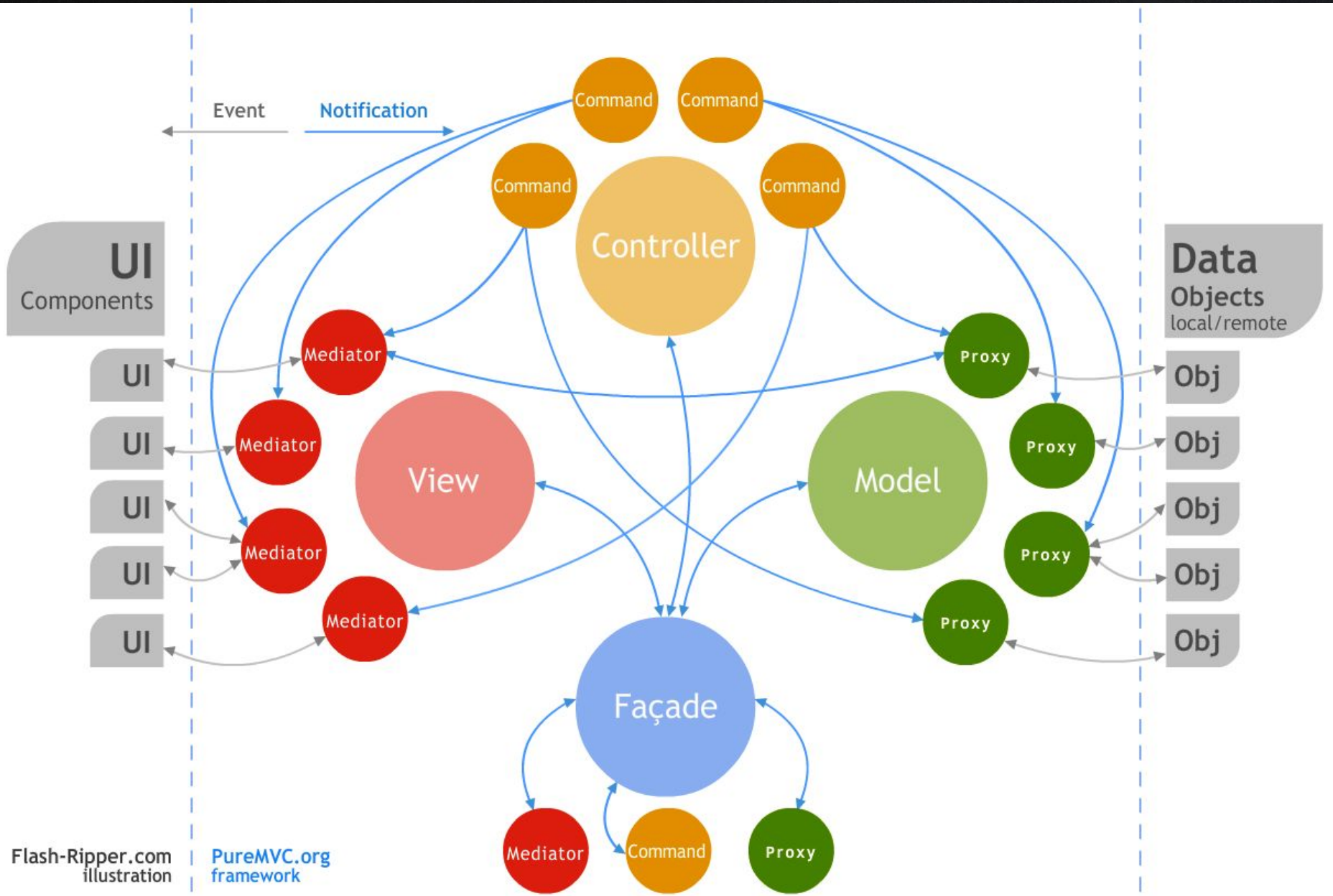


Схема PureMVC (0xffffff)



Что дает PureMVC

- Дисциплина разработки
- Общий язык для архитекторов
- Более устойчивый к рефакторингу проект

Структура PureMVC

- **Фасад:**
  - Показать все, что скрыто.
- **Медиатор:**
  - Скрыть все, что видно.
- **Посредник:**
  - Отдать то, что взято.
- **Команда:**
  - Сделать, когда нужно.
- **Оповещение:**
  - Сделаю все, что скажут.
  - Пойду туда, куда пошлют.
  - Принесу то, не знаю что.

Популярность PureMVC

AS2
Java
AS3 Standard
AS3 MultiCore
ColdFusion
C#
Perl
PHP
Python
Ruby

Мы хотим знать разницу

- **PureMVC or Cairngorm?**

  >> Интересует Ваше мнение по этим библиотекам
  >> и вообще по паттерным решениям для Flex.

  Без сомнения и то и другое полезно. Но если бы еще авторы их уделяли достаточно внимания "расжевыванию" своих замыслов реализованных во фреймворке.

  Качественное "расжевывание" - это когда даже студенту-программисту незнакомому с паттернами будет ясно что к чему. Потому что когда специалист не может оценить риски по миграции - это уже либо бестолковое описание, либо "особая форма маркетинга" (сокрытие информации о недостатках продукта) ИМХО.

  [http://groups.google.com/group/fpug/browse_thread/thread/3770e9c4a296dad9]

Изучение PureMVC

- http://puremvc.org
  - Goals & Benefits
  - Conceptual Diagram
  - Framework Overview
  - Best Practices

Статьи о PureMVC на русском

- 10 советов по PureMVC
- Что мы знаем о flash/flex фреймуорках?
- Пример Flash-галереи на PureMVC
- Как создать простой FLV-плеер во Flex и Flash, используя PureMVC

Мои статьи

- -= Чисто MVC =- (тут важны комменты)
- Архитектура и ключевые фигуры фреймворка PureMVC
- Фасад (Façade) — ядро и лицо фреймворка PureMVC
- Как устроены Модель, Вид и Управление во фреймворке PureMVC
- Кто использует Cairngorm?

- Обновления по статьям, новые ссылки:
  http://flash-ripper.com/archives/002194.php

Развитие PureMVC: Мультитон

- **Multiton** вместо Singleton
  - http://en.wikipedia.org/wiki/Multiton_pattern
- **Мультитон** – паттерн, подобный Синглтону, позволяющий создание более одного экземпляра класса.
- Вместо Одиночки – группа Одиночек, доступных по ключам (экземпляров).
  - Клиенты не могут добавлять ключи.
  - Никогда не возвращает пустую ссылку.
- Польза: упрощение работы с общими ресурсами в приложении (shared objects).
- Централизованный доступ к хранилищу.

Пример использования PureMVC

- в приложении карты -

Обновление карты-1: Вид

Класс MapView (boundary)

```
package view
{
   public class MapView extends Sprite
   {
    private function updateMap():void
    {
       mapMediator.onBBoxChanged(layersToLoad);
    }
   }
}
```

Обновление карты-2: Медиатор

Класс MapMediator

```
package view
{
   public class MapMediator extends Mediator
   {
    internal function onBBoxChanged(layersToLoad: Array):
   void
    {
        var noteBBoxUpdate: Notification = new
   Notification(ApplicatonFacade.NOTE_BBOX_UPDATE);
        noteBBoxUpdate.setBody(layersToLoad);
        facade.notifyObservers(noteBBoxUpdate);
    }
   }
}
```

Обновление карты-3: Фасад

Класс ApplicationFacade

```
package
{
   public class ApplicatonFacade extends Facade implements IFacade
   {
    public static const NOTE_BBOX_UPDATE: String= "noteBBoxUpdate";

    // Commands registration
    override protected function initializeController(): void
    {
        super.initializeController();

        // Layer Model.updatePatchesInRectangle
        registerCommand(NOTE_BBOX_UPDATE, UpdateBBoxCommand);
      }
    }
}
```

Обновление карты-4: Команда

Класс UpdateBBoxCommand

```
package controller
{
   public class UpdateBBoxCommand extends SimpleCommand
   {
    override public function execute(note:
   INotification): void
    {
        ApplicatonFacade.msProxy.updateMap(note.getBody()
   as Array);
    }
   }
}
```

Обновление карты-5: Прокси

Класс MapServerProxy

```
package model
{
  public class MapServerProxy extends Proxy
  {
   public function updateMap
  (layersToLoad:Array):void
   {
      mapModel.loadLayers(layersToLoad);
   }
  }
}
```

Обновление карты-6: Модель

Класс MapModel

```
package model
{
   internal class MapModel
   {
    internal function loadLayers (arrLayersToLoad:Array):void
    {
       // собственно загрузка данных
    }
   }
}
```

PureMVC спешит на помощь

- Рефакторинг 1: Смена источника данных с Geo XML на AMF

PureMVC помогает еще раз

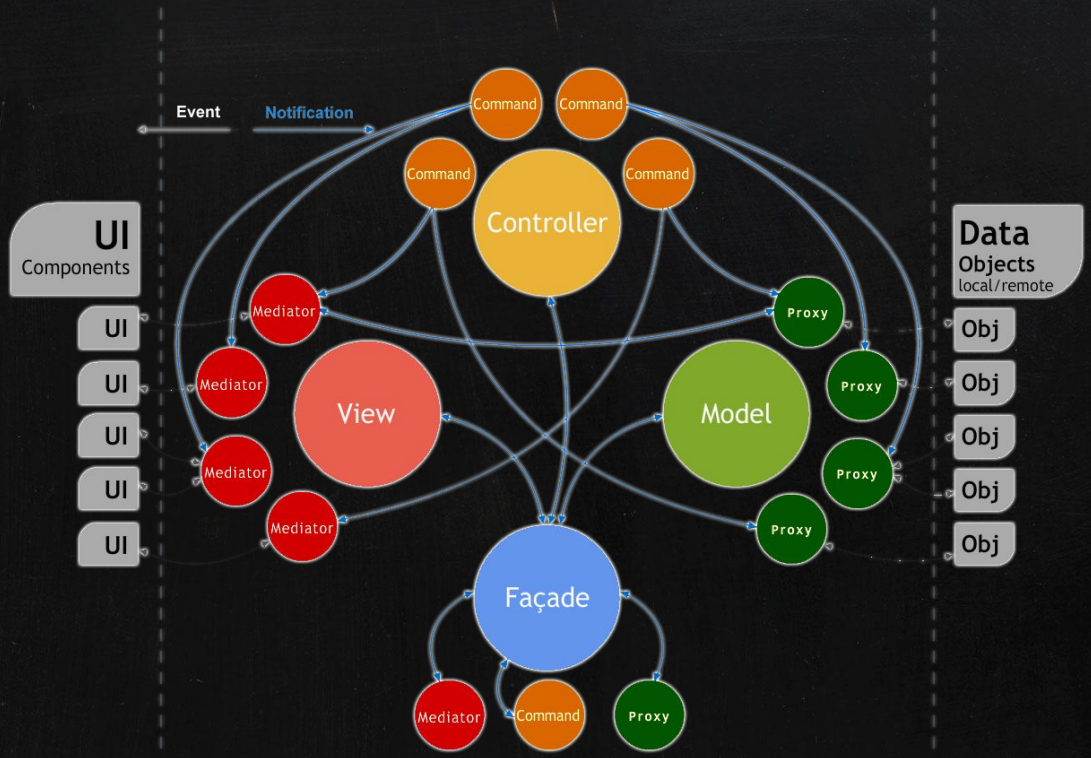
- Рефакторинг 2. Древовидный рендеринг карты.
- Смена сигнатуры метода
  - Было: private function renderGeoEntity(data: FlashGeoEntity): void
  - Стало:
    private function renderGeoEntity(): void // после initGeoEntity
- Переход от Спрайтов к Шейпам
- Откат на Спрайты ☻

5 советов по PureMVC

1. Используйте **Медиаторы**: создавайте API для компонентов вида в Медиаторах, а не используйте их методы напрямую.
2. Используйте **Оповещения** почаще (но не напрямую от Медиатора к Прокси).
3. Используйте **Команды** и МакроКоманды.
4. Используйте **Remote Proxy**.

Почему проект выжил

1. Благодаря разделению кода с самого начала
2. Благодаря разделению кода с самого начала
3. Благодаря разделению кода с самого начала
4. Благодаря разделению кода с самого начала
5. Благодаря разделению кода с самого начала
6. Благодаря разделению кода с самого начала
7. Благодаря разделению кода с самого начала
8. Благодаря разделению кода с самого начала
9. Благодаря разделению кода с самого начала
10. Благодаря разделению кода с самого начала
11. Благодаря разделению кода с самого начала
12. Благодаря разделению кода с самого начала


Event
Notification
Command
Command
Command
Command
Controller
UI
Components
UI
UI
UI
UI
UI
Mediator
Mediator
Mediator
Mediator
View
Model
Proxy
Proxy
Proxy
Proxy
Data
Objects
local/remote
Obj
Obj
Obj
Obj
Obj
Façade
Mediator
Command
Proxy

Q & A