



Flex Camp
Sofia 2008

Шаблони и практики за изграждане на RIA с Adobe Flex

Бранимир Ангелов Ангелов

За какво ще говорим?

- Концепции и процеси които могат да ви бъдат полезни при разработването на RIA приложения.
- Шаблони, практики и архитектури, както и някои програмни техники.
- Всичко което ще видите се прилага и е показало, че работи.

- Изграждане на RIA “отпред назад”.
- Процесът Feature Drive Development
- Шаблони, практики и frameworks свързани с Flex
- Програмни техники
- Ресурси



Flex Camp
Sofia 2008

Разработване “отпред назад”

- “RIAs combine the flexibility, responsiveness, and ease of use of desktop applications with the broad reach of the web. RIAs provide a dynamic web experience that is rich and engaging, as well as interactive”

- Основният акцент е поставен върху изживяването на потребителя (user experience)
- Осигуряването на това изживяване трябва да заема подобаващо място в процеса на разработка
- Единственият начин да разбереш, е да изживееш.

- Грабване на вниманието на клиента чрез работещи mock приложения.
 - Клиента може да се докосне до своето приложение.
 - Ще стимулират неговото мислене и ще разберете какви точно са целите му.
 - Ще имате общ и недвусмислен език.
 - Ще изградите по-точна картина на изискванията към приложението.
- Mock приложенията стават бързо и лесно с Flex.

- Изграждане на модела на данните.
 - Оформяне на суров модел на данните.
 - Възможност за ранно и лесно адресиране на основните въпроси свързани с него и тяхното уточняване.
 - Модела на данните трябва да бъде оптимално независим от неговата визуална презентация.

- Свързване със статични данни.
 - По-пълна представа за това, как ще изглежда реалното приложение.
 - Уверяваме се, че интерфейса ни осигурява достатъчно добра поддръжка за необходимите данни.
 - Използване на XML файлове за статичен източник.

- Изграждане на необходимото API
 - По-добро знание за данните, които ще използва приложението и каква част от логиката трябва да седи в клиентския или в сървърния слой.
 - Лесно, ефективно и сигурно разработване на услугите предоставящи данни.
 - Акуратна картина на необходимите функционалности.



Flex Camp
Sofia 2008

Разработване “отпред назад”

- Свързване на потребителския интерфейс с реалните данни
 - Вече сме подsigурили съвместимостта, така че приложението ни е “живо”.



Flex Camp
Sofia 2008

Разработване “отпред назад”

- Предимства
 - Точни изисквания.
 - Ранна възможност за осъществяване на обратна връзка с клиента.
 - Бърза и ефективна разработка
 - Клиентите вече са от нашия отбор.



Flex Camp
Sofia 2008

Разработване “отпред назад”

- Приложение
 - Просто концепция.
 - Инкорпорирайте я внимателно в изчисления от вас процес на разработка.
 - Бъдете разумни.

- Итеративен процес, в който на дадени периоди от време доставяме на клиента реално работещо приложение.
- Често се използва при създаването на приложение с голям набор от характеристики.
- Съдържа пет основни етапа

Feature Driven Development

- Дефиниране на общия модел на приложението
- Изграждане на списък с възможностите на приложението
- Разработване на дадена възможност
 - Планиране
 - Дизайн
 - Реализация

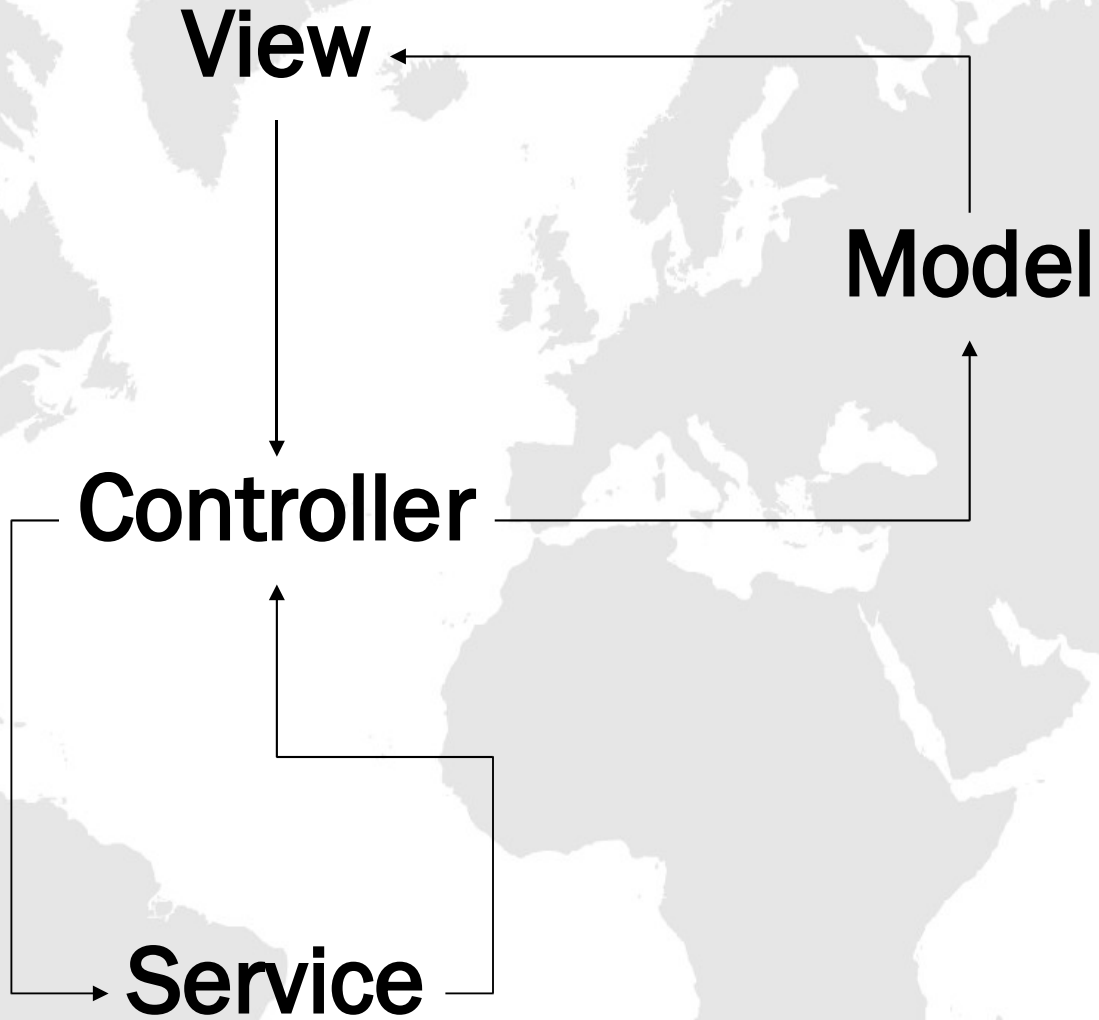


- Cairngorm
 - Един от най-популярните framework за Flex
 - Представява микроархитектура
 - Инспириран от част от шаблоните в Core J2EE Patterns Catalog.
 - Предоставя ни структурирана и широко разширяема архитектура за създаването на RIA приложения.
 - Лесно илюстрира реализирането на RIA чрез Feature Driven Development процес



Flex Camp
Sofia 2008

Cairngorm от високо





Flex Camp
Sofia 2008

Frameworks

- Cairngorm
 - <http://www.cairngormdocs.org/>
- PureMVC
 - <http://puremvc.org/>
- Model-Glue Flex
 - <http://www.model-glue.com/flex.cfm>



- Добре познатите ни шаблони за дизайн от каталога Design Patterns: Elements of Reusable Object-Oriented Software.
- Ще разгледаме някои шаблони и практики в контекста на възможностите на Flex .
 - Шаблони и практики свързани с Data Bindings
 - IoC във Flex
 - Unit testing



- Имаме нужда да изпълним даден метод когато дадено свойство на обект се промени – Observe tag.

Реализация:

```
package com.adobe.ac.util
{
    public class Observe
    {
        public var handler : Function;

        public function set source( source : * ) : void
        {
            handler.call();
        }
    }
}
```

Използване:

```
<util:Observe
    source="{ model.myProperty }"
    handler="{ this.myFunction }"/>
```

- Използвайте строги референции към модела при data binding.
- Задължително **винаги** следете compile time и runtime warnings.



- Оптимизиране и структуриране на сложните и/или неефективни bindind expressions посредством адаптер

```
<mx:List dataProvider="{getCustomerNames(model.customers, model.product)}" />  
<mx:TextBox text="{getTotalCost(model.customers, model.product)}" />
```



```
<adapters:ProductStatisticModel id="statistic" customers="{model.customers}"  
    product="{model.product}" discount="{model.discount}" />  
  
<mx:List dataProvider="{statistic.names}" />  
<mx:TextBox text="{statistic.cost}" />
```



- Основни идеи на Inversion of Control

```
package com.samples
{
    public class Loader
    {
        public function load() : void
        {
            ...
            var bar : IBar = new LoopingProgressBar();
            ...
        }
    }
}
```

- Проблемът е, че по този начин имаме висока зависимост между класовете
- Да изнесем създаването и управлението на компонентните зависимости.



- **ТИПОВЕ**
 - Dependency Lookup
 - Dependency Pull
 - Contextualized Dependency Lookup
 - Dependency Injection
 - Constructor Dependency Injection
 - **Setter Dependency Injection**



- Какво става, ако използваме setter injection

```
package com.samples
{
    public class Loader
    {
        private var _bar : IBar;
        public function set bar(value : IBar) : void
        {
            ...
            bar = value;
            ...
        }

        public function get bar() : IBar
        {
            return _bar;
        }

        public function load() : void
        {
            //използваме променливата bar
        }
    }
}
```



- Пример за управление на зависимостите между MVC компонентите

```
<control:controller id="controller" model="{model}" />  
<model:StockModel id="model" />  
<view:StockDisplay id="view" model="{model}" buy="controller.buy(event.stocks)" />
```



- Предимства
 - Намаляме кода, който по Качва зависимостите
 - Изнасяме и управляваме зависимостите на точно определено място
 - Увеличаваме тествуваемостта
 - Постигаме по-добър дизайн на самото приложение



Flex Camp
Sofia 2008

Програмни техники

- Flex Modules и RSLs
 - Намаляване на големината на приложението
 - По-добро структуриране, което води до по-добър loading experience
 - <http://labs.adobe.com/technologies/flex/videos/fbmodul>
 - http://labs.adobe.com/wiki/index.php/Flex_3:Feature_Introductions:Flex_3_RSLs

- Metadata програмиране
 - Възможност за запазване на custom metadata тагове
 - keep-metadata-tags
 - Reflection API
 - DescribeType()
 - Приложения



Flex Camp
Sofia 2008

Програмни техники

- Unit Testing
 - FlexUnit
 - <http://code.google.com/p/as3flexunitlib/>
 - http://www.adobe.com/devnet/flex/articles/unit_testing.html
 - Продължителна интеграция
 - http://life.neophi.com/danielr/files/DanielRinehart_Continuouslr



- Adobe Developer Connection
 - <http://www.adobe.com/devnet/>
- Flex Coders Yahoo Group
 - <http://tech.groups.yahoo.com/group/flexcoders/>
- Flex
 - <http://flex.org/>
- Adobe Labs
 - <http://labs.adobe.com/>