



**Flex Camp**  
Sofia 2008

# Работа с данни във Flex

Бранимир Ангелов Ангелов

- Предизвикателствата, които RIA поставя към данните.
- Основи на данните във Flex
- Data binding
- Колекции
- Services

## Какво е различното във Flex

- Данните в стандартните web приложения
- Различното в RIA
- Flex използва чиста data service ориентирана архитектура



**Flex Camp**  
Sofia 2008

# Основи на данните във Flex

- Основни “парадигми”
  - Слабо типизирани данни
  - XML обекти
  - Силно типизирани данни



# Слабо типизирани данни

- Слабо типизирани данни
  - Actionscript Object class

Actionscript:

```
var person : Object =  
    new Object();  
person.name = "Name";
```

MXML:

```
<mx:Object id="person"  
    name="Name" />
```

- Не подържат възможност за нотифициране при промени



# Слабо типизирани данни

- ObjectProxy class
- Нотификация при промени.

Actionscript:

```
var person : ObjectProxy =  
    new ObjectProxy();  
person.name = "Name";
```

MXML:

```
<mx:Model id="person">  
    <person>  
        <name>Name</name>  
    </person>  
</mx:Model>
```

## Слабо типизирани данни

- Предимства:
  - Възможност за динамично добавяне на свойства
  - Не се налага да пишем почти никакъв код
- Недостатъци
  - Голям риск за грешки.
  - Не може да си сигурен, че всичко работи преди да стартираш приложението.
    - Дори и тогава не може да си напълно убеден
  - Не може да се използват възможностите на IDE за code completion и refactoring.

- XML обекти

Actionscript:

```
var person : XML =  
    <person>  
        <name>Name</name>  
    </person>;
```

MXML:

```
<mx:XML id="person">  
    <person>  
        <name>Name</name>  
    </person>  
</mx:XML>
```

- ECMAScript for XML (E4X)

- mymodel.(title == "Pocket Symphony").@rating = "6.8";

- Предимства
  - Възможностите на E4X
    - Изрази
    - Dot нотацията
- Недостатъци
  - Всички недостатъци характерни за слабо типизираните типове
  - Трудно е да се мигрира към силно типизирани данни заради спецификата на E4X изразите
    - `mymodel.(title == "Pocket Symphony").@rating = "6.8";`

- Силно типизирани данни
  - Посредством атрибута [Bindable] можем да активираме възможността за нотификация при промени.

Actionscript (Person.as):

```
[Bindable]
public class Person
{
    public var name : String;
}
```

MXML (Person.mxml):

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Object
    xmlns:mx="http://www.adobe.com/2006/mxml">
    <mx:String id="Name"></mx:String>
</mx:Object>
```



- Data binding
  - Процесът на свързване на данните от един обект с друг обект.
  - Има три начина за задаване:
    - `BindingUtils.bindProperty()` и `BindingUtils.bindSetter()`
    - `<mx:Binding source="person.name" destination="text1.text" />`
    - `<mx:Text id="text1" text="{person.name}" />`



- Един източник към много дестинации

```
<mx:List id="list" dataProvider="{model}" />  
<mx:DataGrid id="dataGrid" dataProvider="{model}" >  
...  
</mx:DataGrid>
```

```
<mx:Binding source="model" destination="dataGrid.dataProvider" />  
<mx:Binding source="model" destination="list.dataProvider" />
```



- Много източници към една дестинация

```
<mx:Binding source="model.firstCollection" destination="list.dataProvider" />  
<mx:Binding source="model.secondCollection" destination="list.dataProvider" />  
<mx>List id="list" />
```

- Използване на функции като източник
  - Чрез bindable аргументи
  - В резултат на събитие



- Через bindable аргументи

```
<mx:Text text="{formatName(person)}" />
```



- В резултат на събитие

```
<mx:Text text="{person.getFullName()}" />
```

```
public class Person
{
    [Bindable("nameChanged")]
    public function getFullName() : String
    {
        ....
    }
}
```



- Binding chains
  - `<mx:Binding source="model.person.name" destination="name.text" />`



- Атрибутът [Bindable]
  - [Bindable] над дефиницията на клас
  - [Bindable] над свойство или функция
  - [Bindable(“eventName”)] над свойство или функция

- Възможност за нотификация при промяна
  - `CollectionEvent`
- Механизми за работа с `paged` данни
  - `ItemPendingError`
- Абстракции
  - `IList`,  `IHierarchicalData`
- Изгледи върху данните
  - `ICollectionView`,  `IHierarchicalCollectionView`

- ArrayCollection
  - Реализира IList, ICollectionView
  - Чрез метода си getItemAt(index) може да участва в binding изрази



- REST
  - HTTPService
    - ResultFormat
      - Object
      - E4X
      - Text
    - Конвертирането до силно типизирани данни трябва да се извършва ръчно

- RPC
  - Web Services
    - Възможност за изграждане на proxy, в резултат на динамичната обработка на WSDL-а
    - Възможност за изграждане на статично proxy, чрез Web Service Import Wizard
      - Изграждане на статично силно типизирано прокси
      - Изтеглянето и обработката на WSDL-а вече не се извършва runtime, с което се покачва бързодействието на приложението



- Remote Object
  - Използва AMF протокола
  - Изключително бързодействие
  - Съпоставянето на класовете се извършва посредством [RemoteClass] атрибут



- Обработка на резултата
  - Комуникацията е асинхронна
  - Чрез събитията result и fault
  - Чрез data binding към lastResult
  - Чрез използване на AsyncToken

- Data services
  - Синхронизиране на колекции между клиент и сървър
  - Разрешаване на конфликти
  - Push към клиента
  - Осигуряване на бърздействие при големи количества данни
    - Paged data
    - Мързеливо зареждане на йерархични данни
      - [Managed]

- Основни методи за работа
  - `<mx:DataService id="service" destination="items" />`
  - Чрез методите `fill`, `commit` извършваме синхронизацията със сървърната колекция
  - `get/create/delete/releaseItem()` - извършваме управление на ниво `item`
  - Съпоставянето на сървърните и клиентските класове се извършва чрез `[RemoteClass]`
  - `DataConflictEvent` - излъчва се при възникване на конфликт
  - Server-side assembler - `fill/get/count/sync()`
    - `AbstractAssembler` - `updateItem/createItem/deleteItem()`

- LiveCycle Data Services ES Express – безплатна версия за една машина с един процесор



- Data wizards \*
- Лесно генериране на прости CRUD приложения
- PHP с HTTPService
- .NET с WebService
- Java с LCDS



- Data binding
  - <http://blogs.adobe.com/flexdoc/pdf/databinding.pdf>
- LCDS
  - [http://www.adobe.com/go/trylifecycle\\_dataservices](http://www.adobe.com/go/trylifecycle_dataservices)