

# Разработка приложений для OS X

## Лекция 6

Key-Value Coding. Document-based app. Undo/redo.

Hexlet University, 2012

# KVC

```
someObject.someProperty = someValue;
```

```
[someObject setValue:someValue  
    forKey:@"someProperty"];
```

# KVC

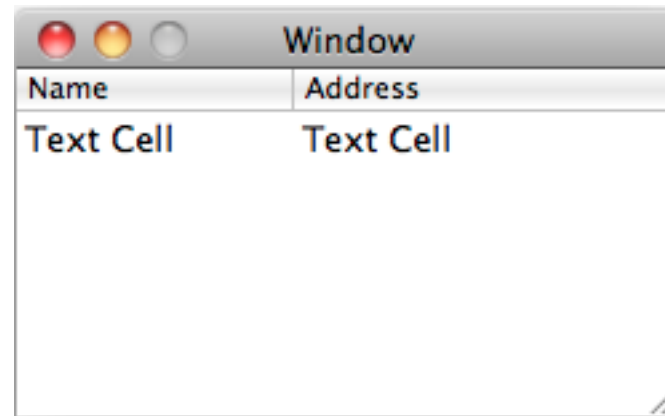
```
someObject.someProperty = someValue;
```

```
[someObject setValue:someValue  
    forKey:@"someProperty"];
```

**Отделение действия от свойства**

Name	Address
Text Cell	Text Cell

```
- (void)tableView:(NSTableView *)aTableView
    setObjectValue:(NSString *)anObject
    forTableColumn:(NSTableColumn *)aTableColumn
    row:(int)rowIndex
{
    if ([[aTableColumn identifier] isEqual:@"name"])
    {
        [[records objectAtIndex:rowIndex] setName:anObject];
    }
    else if ([[aTableColumn identifier] isEqual:@"address"])
    {
        [[records objectAtIndex:rowIndex] setAddress:anObject];
    }
}
```



Window	
Name	Address
Text Cell	Text Cell

```
- (void)tableView:(NSTableView *) aTableView
    setObjectValue:(NSString *) anObject
    forTableColumn:(NSTableColumn *) aTableColumn
    row:(int) rowIndex
{
    [[records objectAtIndex:rowIndex] setValue:anObject
                                             forKey:[aTableColumn identifier]];
}
```

# Еще пример

```
@interface Person : NSObject  
{
```

```
    NSString* firstName;  
    NSString* lastName;  
    NSString* email;  
}
```

```
- (void) setFirstName: (NSString *)value;  
- (void) setLastName: (NSString *)value;  
- (void) setEmail:    (NSString *)value;  
@end
```

```
NSDictionary* input;  
Person* person = [[Person alloc] init];
```

```
NSEnumerator* e = [input keyEnumerator];  
id dictKey, dictValue;
```

```
while ( dictKey = [e nextObject] )  
{  
    dictValue = [inputValues valueForKey: dictKey];  
    [person setValue: dictValue forKey: dictKey];  
}
```

# Еще проще!

```
NSDictionary * inputValues;  
Person       * person = [[CDCPerson alloc] init];  
  
[person setValuesForKeysWithDictionary: inputValues];
```

# Получение значений

```
Person * person;  
NSArray * keys;
```

```
keys = [NSArray arrayWithObjects: @"firstName",  
                                   @"lastName",  
                                   @"emailAddress",  
                                   nil];
```

```
NSDictionary* output;  
output = [person dictionaryWithValuesForKeys: keys];
```



# Key paths

```
myObject.dance.speed = 2;
```

```
[myObject valueForKeyPath: @"dance.speed"];
```

```
[myObject setValue: 2 forKeyPath: @"dance.speed"];
```

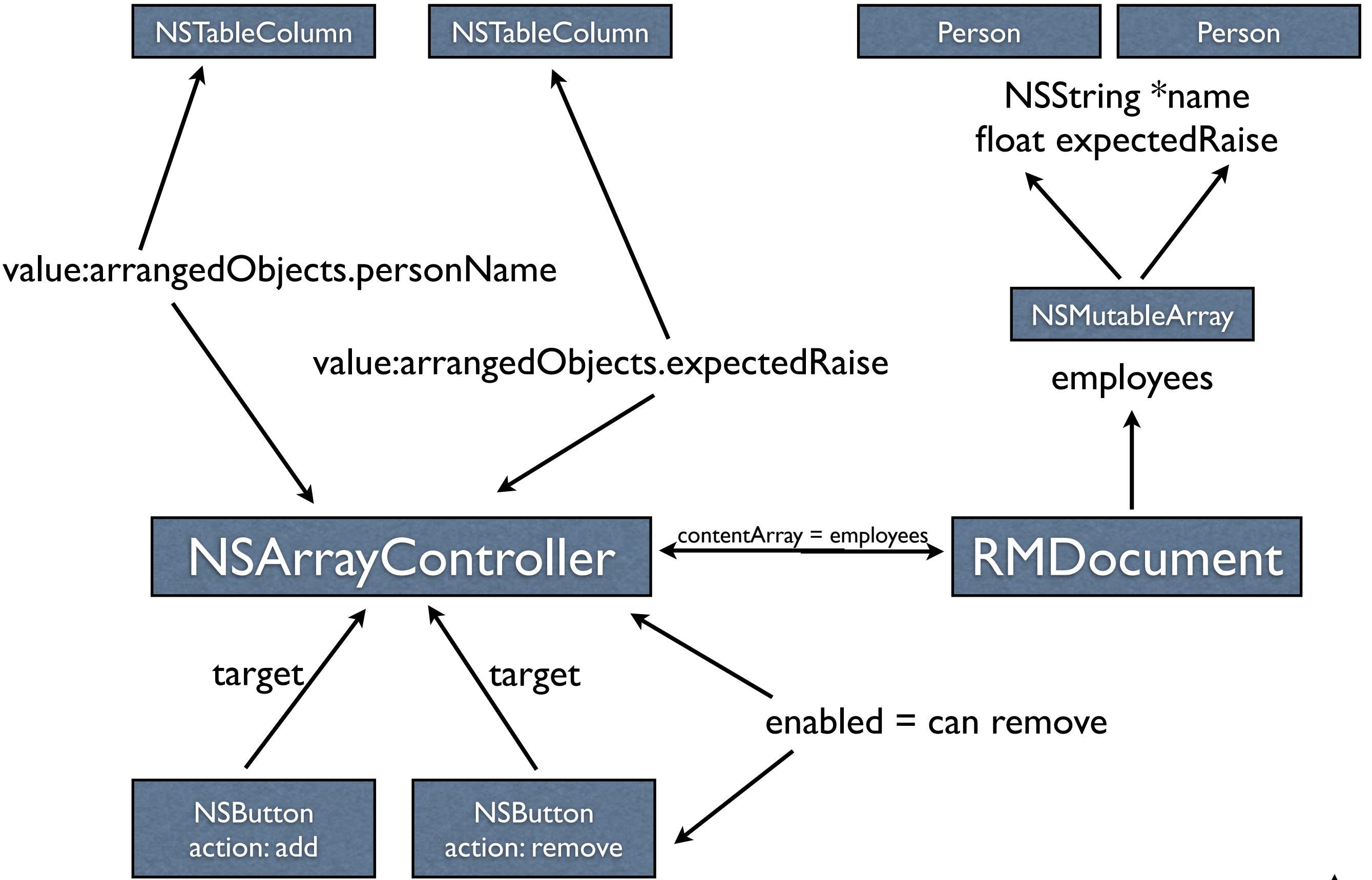
# Key-Value Observing

```
[self addObserver:self  
      forKeyPath:@"myDict"  
      options:0  
      context:@"dictChanged"];
```



# Document-based app

- Приложение, работающее с несколькими документами (обычно в отдельных окнах)
- MVC!
- Документ (подкласс `NSDocument`) – контроллер



# NSUndoManager

- Хорошие приложения позволяют пользователям отменять и повторять последние действия (в отличие от реальной жизни, чёрт побери!)
- Мы можем реализовать свой механизм или воспользоваться NSUndoManager

# Идея: изменения — в стек

- Сохранить все изменения куда-то
- Изменения — результат сообщений
- Так что давайте сохранять сообщения
- Но сообщения это не объекты... хотя...

# NSInvocation

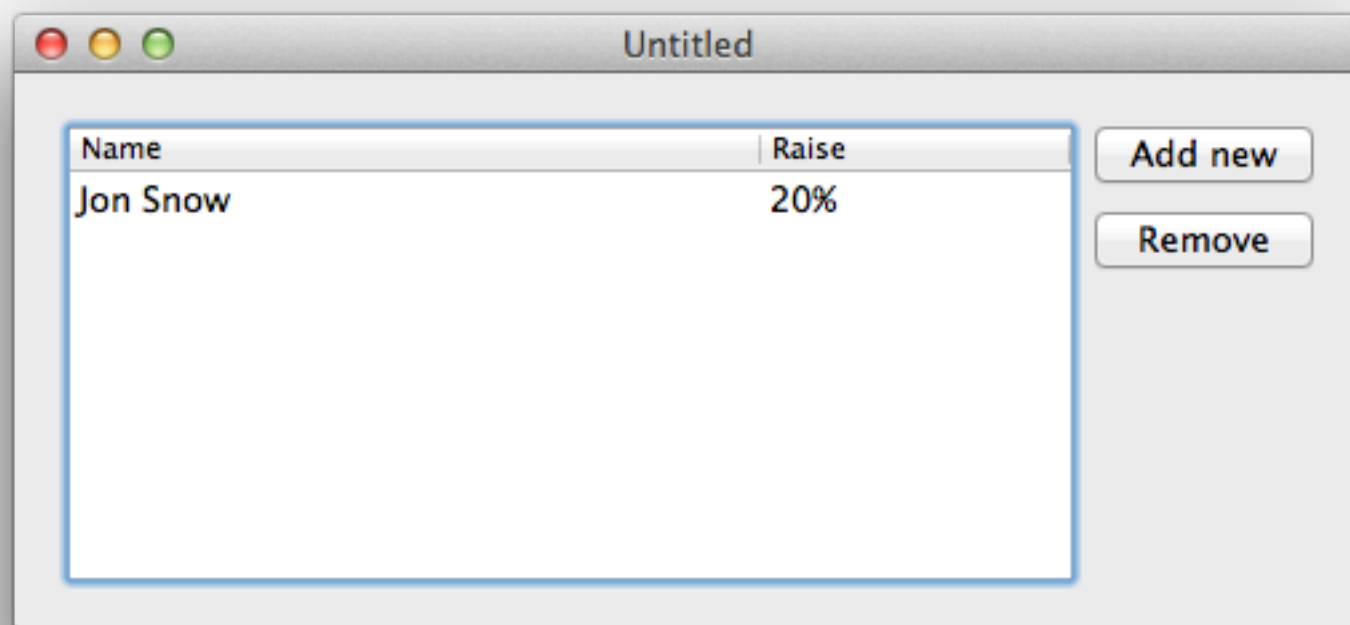
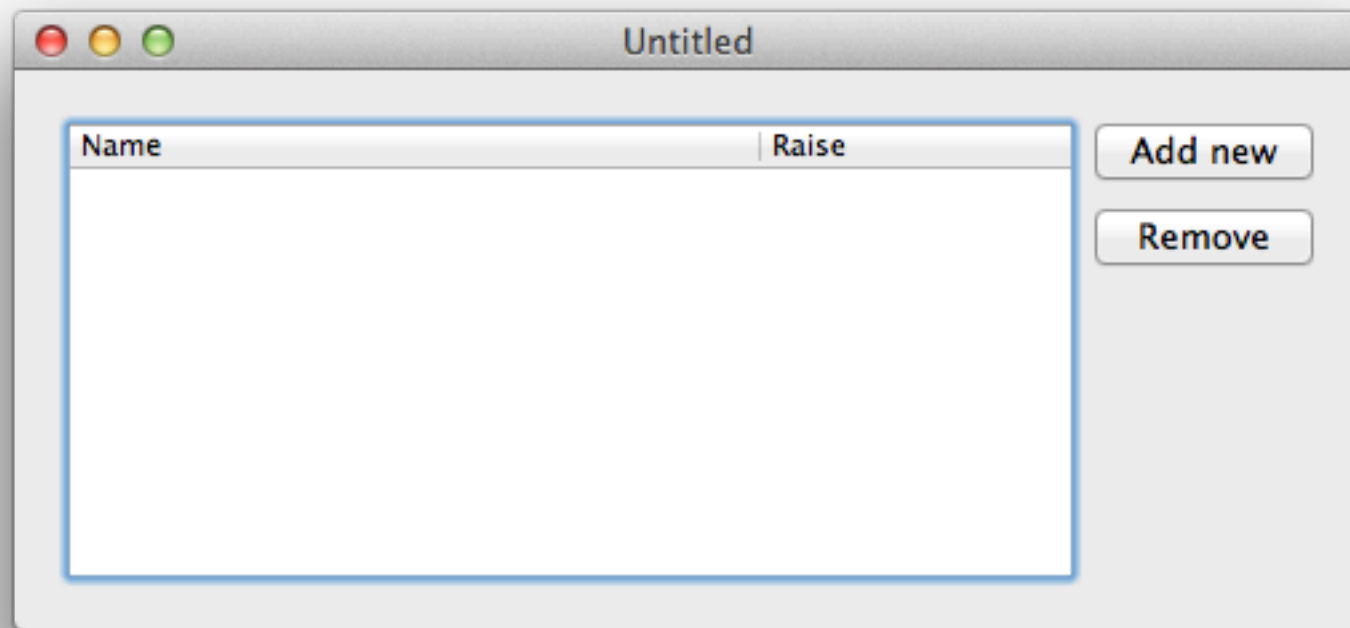
- “Упаковка” сообщения – селектора (названия), получателя и аргументов – в единый объект
- Может использоваться для перенаправления сообщений

# Message Forwarding

- Если объект получает сообщение, которое не может обработать, то перед обработкой исключения система проверяет наличие такого метода:
  - `(void)forwardInvocation:(NSInvocation *)x`
- если он существует, то сообщение запаковывается в `NSInvocation` и передается в `forwardInvocation`



# NSUndoManager



- Добавить запись
- Изменить “Unnamed Person” на “Jon Snow”
- Изменить Raise с 30% на 20%

- Программа запущена

Undo stack

Redo stack

- Добавить запись

Undo stack

NSInvocation  
Delete record #1

Redo stack



- Изменить “Unnamed Person” на “Jon Snow”

Undo stack

**NSInvocation**

Set name to “Unnamed Person”

**NSInvocation**

Delete record #1

Redo stack

- Изменить Raise с 30% на 20%

Undo stack

NSInvocation

Set raise to 30%

NSInvocation

Set name to “Unnamed Person”

NSInvocation

Delete record #1

Redo stack



- UNDO

## Undo stack

**NSInvocation**

Set name to “Unnamed Person”

**NSInvocation**

Delete record #1

## Redo stack

**NSInvocation**

Set raise to 20%

- UNDO

Undo stack

**NSInvocation**  
Delete record #1

Redo stack

**NSInvocation**  
Set name to “Jon Snow”

**NSInvocation**  
Set raise to 20%

- UNDO

Undo stack

Redo stack

**NSInvocation**  
Add new record

**NSInvocation**  
Set name to “Jon Snow”

**NSInvocation**  
Set raise to 20%



# NSUndoManager

- Следит за стеками
- Наша задача – предоставить ему обратные операции

# Пример

```
- (void)makeItHotter
{
    temperature = temperature + 10;
    [[undoManager prepareWithInvocationTarget:self] makeItColder];
}

- (void)makeItColder
{
    temperature = temperature - 10;
    [[undoManager prepareWithInvocationTarget:self] makeItHotter];
}
```

# NSUndoManager

- Вызовы сгруппированы по действиям. Одно действие может генерировать несколько сообщений: все они будут включены в один undo/redo
- Текст в меню Edit->Undo можно поменять так:  
`[undoManager setActionName:@"Insert"];`



# Добавим undo/redo

- Нужно создать новый **NSUndoManager**, но в нашем **NSDocument** он уже есть

# Back to our app...

- У нас есть **array controller** и мы связали его **contentArray** с массивом **employees** в RMDocument
- Array controller использует key-value coding для добавления и удаления объектов
- Мы используем это для undo/redo



# Undo для текста

- KVO!
- Нужно сообщать документу о каждом изменении текста
  - `(void)addObserver:(NSObject *)observer  
forKeyPath:(NSString *)keyPath  
options:(NSKeyValueObservingOptions)options  
context:(void *)context;`
  - `(void)observeValueForKeyPath:(NSString *)keyPath  
ofObject:(id)object  
change:(NSDictionary *)change  
context:(void *)context;`