Текст речи для выступления на КМУ

Кислюк Игорь

Добрый день, коллеги, сегодня хотел бы рассказать об особенностях использования среды исполнения для реализации библиотеки анимации представлений в рамках модели mvc.

Передо мной стояла следующая задача: разработать библиотеку для анимирования с учетом условий того, что придется создавать свойства для неизменяемых объектов. В тот же момент, необходимо было получить более компактный синтаксис

В моем случае был только один способ выполнить данную цель - воспользоваться средой исполнения языка Objective-C.

План доклада будет следующий: сперва расскажу о том, что такое среда исполнения, как она работает, какой функционал предоставляет и в чем ее главное назначение. После чего расскажу про главные особенности, которые наиболее активно используются разработчиками. Впоследствии перейду к тому, какие приемы пришлось использовать, чтобы добавить свойства к неизменяемому объекту и получить наиболее удобный для разработчиков синтаксис

Как известно, в начале 1980-х популярным было структурное программирование, однако постоянно увеличивающаяся сложность привела к тому, что Брэд Кокс пытался решить проблему повторного использования кода, как и многие другие параллельно с ним.

Таким образом, его целью было создание языка, в котором классы являлись бы полноценными объектами, поддерживалась бы интроспекция (возможность определения типа объекта в момент исполнения программы) и динамическая обработка сообщений. В тот же момент, он хотел использовать уже знакомый разработчикам язык - это был C. Собственно, именно так и появился язык Objective-C

Динамичность считается достоинством языка, которая выражается в том, что большинство решений принимаются непосредственно в момент выполнения, а не в момент компиляции. Отдельно стоит упомянуть ориентированность языка на работу с сообщениями вместо таблиц методов, а также поддержка работы с метаинформацией, с помощью которой можно получить название класса, список его методов, переменных класса

Наибольшей популярностью пользуется так называемый "метод swizzling" - замена методов(функций) в процессе исполнения программы. Однако данным функционалом пользуются нечасто, по понятной причине - его сложно поддерживать, однако если необходимо внести исправления в сторонние библиотеки или в те участки кода, доступа к которым не имеется - среда исполнения - единственный выход.

Не менее интересная особенность языка, которая вытекает из его динамичности - это категории классов. С их помощью легко добавлять функциональность даже в недоступные для изменения классы, правда нельзя создавать свойства, но и это решается динамичностью языка

Для своей библиотеки мне понадобилась работа с метаинформацией класса. Причиной тому была цель получения удобство пользования библиотекой.

Зачем? Чтобы отойти от концепции, где самому нужно создать объект нужного класса, выполнить преобразования над ним, а также следить за жизненным циклом данного объекта. Необходимо было создать свойство в момент выполнения программы, заполнить необходимой фабрикой объектов, после чего присвоить значение в соответствующем момент выполнения программы.

Вторым узким местом и проблемой, которую необходимо было решить - это отход от стандартного синтаксиса языки и получении более читаемого для удобства пользователей библиотеки.

Как известно, синтаксис языка выглядит примерно следующим образом. (Показать волшебные квадратные скобки)

Хотелось достичь следующего результата. (Показать текущий синтаксис библиотеки)

В других языках такое довольно просто осуществить, но не в данном случае. Решить проблему получилось при использовании блоков в качестве возвращающих значений. Известно, что блок - это указатель на анонимную функцию, аналог замыкания.

Как же выглядит блок в нашем коде? (Следующим образом, а вот так выглядит его вызов)

Как раз то, что необходимо нам. Здесь приходит в голову идея, что мы можем возвращать блок в качестве результирующего значения из нашей функции (сообщения), но все же это далеко от необходимого нам синтаксиса.

Мы можем использовать свойства, а именно методы доступа для получения более удобочитаемого синтаксиса. Тут стоит обратиться к тому, как реализованы свойства в языке, согласно конвенции языка - Cocoa. Свойство имеет метод доступа, соответствующий имени свойству, и метод записи, соответствующий имени типа "set" + имя свойства.

Таким образом нам остается только переопределить методы доступа, и мы сможем обращаться к нашему блоку, который будет отдаваться в качестве свойства в наиболее читаемой форме, именно той, которую мы и хотели получить.

Таким образом, были продемонстрированы решения наиболее интересных мест для библиотеки - получения компактного синтаксиса, создание свойств для неизменяемых объектов, и оформление функциональности в родительском неизменяемом классе. Вопросы?