DOI: 10.17117/na.2015.12.02.058

Поступила (Received): 03.12.2015

<http://ucom.ru/doc/na.2015.12.02.058.pdf>**Гонаров В.А.****Методологический подход к совершенствованию паттернов проектирования архитектуры приложения****Goncharov V.A.****The methodological approach to improve the application architecture design patterns**

Статья посвящена демонстрации разработки приложения используя известные паттерны проектирования и их модификацию за счет использования других паттернов проектирования, в статье показано как лучше строить архитектуру приложения используя методологию проектирования архитектуры SOLID

Ключевые слова: паттерны, программирование, SOLID

The article is devoted to demonstrating the application development using known design patterns and their modification through the use of other design patterns, the article shows how to build a better application architecture using SOLID architecture design methodology

Key words: pattern, programming, SOLID

Гончаров Валентин Александрович

Студент

Поволжский государственный университет сервиса

г. Тольятти, ул. Гагарина, 4

Goncharov Valentin Aleksandrovich

Student

Volga region state university of service

Togliatti, Gagarin st., 4

Современные требования к процессам разработки приложений определяют важность не только функциональной полноты и соответствию техническому заданию, но и возможность эффективной модернизации приложения при необходимости изменения существующего или добавления нового функционала. Построение приложений по принципу “оно же работает” означает, изменение функционала будет стоить больших усилий, поскольку код очень сильно связан между собой, изменение малой части системы может повлечь за собой неожиданные последствия в другой части системы. Для примера возьмем проектирование архитектуры приложения типа “Онлайн магазин” [1]. Проектирование базируется на применении различных паттернов проектирования, таких как MVC (Модель-Отображение-Контроллер), EventBus (Шина событий) [2] и Dependency Injection (Инъекция зависимостей) [3]. Отметим, сами паттерны структур не являются чем то незыблемым, и вопрос о методологической организации проектирования паттернов структур является актуальным и рассмотрен в настоящей работе.

Очевидно, что, как при проектировании приложений, так и при проектировании паттернов структур эффективным будет использование общих методологий проектирования, в частности [4]:

1. Принцип единой обязанности (Single responsibility principle), говорящий, что класс должен существовать только для одной цели и ни какой другой.

2. Принцип открытости и закрытости (Open/Closed principle), говорящий, что код программы должен быть открыт для расширения, но закрыт для изменения.

3. Принцип Барбары Лисков (Liskov substitution principle), говорящий, что функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа, не зная об этом, например, абстракция, измеряющая температуру, не должна идти от абстракции “Устройство изменяющее температуру”.

4. Принцип разделения интересов (Interface segregation principle), говорящий, что много маленьких абстракций лучше, чем одна большая.

5. Принцип инверсии зависимостей (Dependency inversion principle), служащая для уменьшения зависимости классов друг от друга.

Проектируя приложение, часто прибегают к паттернам проектирования, таким как MVC (Модель-Отображение-Контроллер), EventBus (Шина событий), Dependency Injection (Инъекция зависимостей) и др.

Объединив все выше сказанное можно построить модель проектирования MVCP, где большинство букв уже знакомы, добавляет только Presenter (Предъявитель) у нас его роль будет играть EventBus (Шина событий). Presenter (Предъявитель) это слой между отображением и его логикой работы (Например гигантские функции в методах onClick на кнопках). Подобная зависимость в отображение необходима чтобы избавиться от бизнес логики из View (Представления) в контроллер, тогда представление будет содержать минимальную логику и может быть легко протестировано.

Рассмотрим взаимодействие в системе MVCP (рис. 1) применительно к моделированию деятельности “Онлайн магазина”. Поддерживаемые операции это отображение товаров и добавление товаров в корзину.

Запуск приложения начинается с Lifecycle (жизненного цикла), который подтягивает зависимости для компонента и выполняет начальную инициализацию данных. Реализуем поддержку двух операций просмотр товаров и добавление товаров.

За взаимодействие между пользователем и системой отвечает элемент называемый контроллером. Необходимо два контроллера LoadOfferController и AddOfferController, которые будут отвечать на реакцию загрузки и добавления товаров. Для структурирования пользовательских данных отвечает модель, она и же хранит в себе данные. Модель на данный момент, будет одна OffersModel, в ней будут храниться все товары. Получив модель от контроллера, будет необходимо отобразить их в отображение (View), еще их можно называть формами на данный момент их будет форма для отображения товаров и добавления товаров. Для того чтобы отображение содержало мало бизнес логики эту логику делегируют в презентер. Presenter (Предъявитель) создает событие, например

AddOfferEvent и LoadOfferEvent, контроллер ждет эти события и в случай поступления исполняет свою бизнес логику, например, вызывает контроллер.

Рис. 1. Схема MVCP архитектуры

Подобная архитектура имеет ряд достоинств:

1. Логика разделена на отдельные классы, и можно легко догадаться, за что отвечает класс.

2. Разделение логики позволяет легко тестировать код

3. Для добавления нового функционала надо не переписывать, а добавлять, например, при удалении товара, появится новый Контроллер и Событие.

Из изложенного следует, что хорошо продуманный код помогает при написании и поддержке приложения, он помогает структурировать и группировать логику. Если следовать общим принципам архитектуры то большинство приложений будет написаны в одном стиле, а в таком коде гораздо проще разбираться. Так же подобный подход позволяет не модифицировать код, а расширять его.

Список используемых источников:

1. Панюков Д.И., Хрипунов Н.В. Компьютерные технологии в науке и производстве: практикум. Тольятти: Изд-во ТГУ, 2013. 106 с.
2. Джесс, Ч. и др. ASP.NET MVC 4: разработка реальных веб-приложений с помощью ASP.NET MVC. М.: Вильямс, 2013. 432 с.
3. Симан М. Внедрение зависимостей в .NET. Санкт-Петербург: Питер, 2014. 624 с.
4. Мартин Р.С. и др. Быстрая разработка программ. Принципы, примеры, практика. М.: Вильямс, 2004.