Авивов Артем Константинович

магистрант

Обломов Игорь Александрович

канд. техн. наук, доцент

ФГБОУ ВО «Чувашский государственный

университет им. И.Н. Ульянова»

г. Чебоксары, Чувашская Республика

РЕАЛИЗАЦИЯ ШАБЛОНА MVP ДЛЯ ПОСТРОЕНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ WINDOWS FORMS

Аннотация: *в статье рассмотрены особенности реализации шаблона построения пользовательских интерфейсов MVP (Model-View-Presenter) с использованием технологий Windows Forms. Показан подход построения приложений, облегчающий автоматическое модульное тестирование и обеспечивающий разделение ответственности в презентационной логике.*

Ключевые слова: *пользовательский интерфейс, Windows Forms, Net, MVP.*

При разработке любого проекта следует предварительно утвердить архитектуру разрабатываемой системы. Если при создании достаточно простого приложения [1; 3; 4] не возникает критичных проблем при объединении кода, отвечающего за пользовательский интерфейс, и кода бизнес-логики, то при разработке сложной системы [2] такой подход приводит к избыточности, сложности тестирования и увеличению риска появления ошибок.

Разрабатываемая система использует интерфейс программирования настольных приложений Windows Forms [5]. На рис. 1 показано разделение интерфейса, бизнес-логики и данных при применении шаблона MVP.

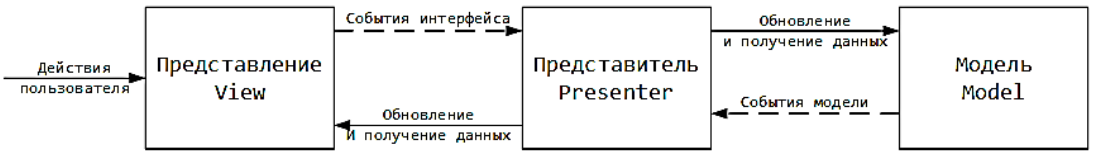

Рис. 1. Архитектура шаблона MVP

При проектировании приложения следует полагать, что существует некий интерфейс представления (IView), который представляет собой некоторый контракт для отображения данных. Представление (View) – это конкретная реализация интерфейса IView, которая способна отображать себя в некотором интерфейсе и не хранит никакой информации о том, откуда приходят команды управления. Моделью является сущность, которая представляет некоторую бизнес-логику (доступ к базе данных, сервисам и т. п.). Сам же представитель (Presenter) взаимодействует с представлением через интерфейс (IView), управляет им, подписывается на его события, производит простую валидацию введенных данных; также содержит ссылку на модель или на ее интерфейс, передавая в нее данные из представления и запрашивая обновления.

При построении приложений с использованием данной архитектуры целесообразно разделить слои представления, представителя и модели на слабосвязанные компоненты – это способствует построению системы с высокой связностью.

Представитель, реализация которого на языке C# приведена на рис. 2, получает ссылки на реализацию соответствующих интерфейсов модели и представления. События представления передаются для обработки в представитель, который имеет средства для обеспечения работы с моделью данных, которая не имеет прямой зависимости от слоя отображения данных.

Для реализации переключения между оконными интерфейсами был задействован шаблон «Application Controller» (рис. 3), который содержит IoC-контейнер, знающий, как по интерфейсу получить объект реализации. При инициализации каждого представителя ему также передается экземпляр контроллера для получения возможности взаимодействовать с другими представителями, не нарушая изолированность от явных реализаций представлений.

```
public class LoginPresenter : IPresenter
{
    private readonly ILoginView _view;

    private readonly ILoginModel _model;

    private readonly IApplicationController _controller;

    public LoginPresenter(IApplicationController controller, ILoginView view, ILoginModel model)
    {
        _view = view;
        _model = model;
        _controller = controller;

        // Подписка на события представления
        _view.Login += () => Login(_view.Username, _view.Password);
    }

    private void Login(string username, string password)
    {
        if (username == null)
            throw new ArgumentNullException("username");
        if (password == null)
            throw new ArgumentNullException("password");

        _model.Authentication(username, password);

        var user = new User { Name = username, Password = password };
        _controller.Run<MainPresener, User>(user);
        _view.Close();
    }

    public void Run()...
}
```

Рис. 2. Реализация представителя

```
public interface IApplicationController
{
    IApplicationController RegisterView<TView, TImplementation>()
        where TImplementation : class, TView
        where TView : IView;

    IApplicationController RegisterService<TService, TImplementation>()
        where TImplementation : class, TService;

    void Run<TPresenter>()
        where TPresenter : class, IPresenter;

    Other
}
```

Рис. 3. Реализация контроллера

При инициализации приложения Windows Forms (рис. 4) сначала настраиваются зависимости между интерфейсами и объектами реализации, затем через начальный представитель запускается основной цикл работы приложения.

```
internal static class Program
{
    public static readonly ApplicationContext Context = new ApplicationContext();

    private static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);

        var controller = new ApplicationController(new LightInjectAdapder())
            .RegisterView<ILoginView, LoginForm>()
            .RegisterView<IMainView, MainForm>()
            .RegisterView<IChangeUsernameView, ChangeUsernameForm>()
            .RegisterService<ILoginService, StupidLoginService>()
            .RegisterInstance(new ApplicationContext());

        controller.Run<LoginPresenter>();
    }
}
```

Рис. 4. Инициализация приложения Windows Forms

В данной статье были рассмотрены основные необходимые концепции построения приложения Windows Forms с использованием шаблона проектирования MVP. Этот подход позволяет достаточно свободно менять логику работы компонентов, не изменяя работу всей системы, а также подходит для ведения разработки с использованием модульного тестирования.

Список литературы

1. Албутов К.В. Помощник в сборке кубика Рубика / К.В. Албутов, И.С. Федоров, А.Л. Симаков // Информатика и вычислительная техника: Сб. науч. трудов. – Чебоксары: Изд-во Чуваш. ун-та, 2016. – С. 12–15.

2. Албутов К.В. Исследование технологий построения высоконагруженных приложений / К.В. Албутов, О.А. Лобастова // Сборник научных трудов молодых ученых и специалистов. – Чебоксары: Изд-во Чуваш. ун-та, 2016. – С. 80–85.

3. Арсентьев С.А. Файловый менеджер / С.А. Арсентьев, А.Л. Симаков // Информатика и вычислительная техника: Сб. науч. трудов. – Чебоксары: Изд-во Чуваш. ун-та, 2016. – С. 27–31.

4. Мамонтов С.В. Разработка масштабируемого пользовательского интерфейса / С.В. Мамонтов, И.А. Обломов // Информатика и вычислительная техника: Сб. науч. трудов. – Чебоксары: Изд-во Чуваш. ун-та, 2015. – С. 80–85.

5. Фаулер М. Шаблоны корпоративных приложений / М. Фаулер [и др.]. – М.: Вильямс, 2016. – 544 с.