Паттерны проектирования: паттерн Стратегия в TypeScript

Как использовать и реализовать паттерн Стратегия в TypeScript для решения реальных проблем в веб-проектах.

Добро пожаловать в серию «Паттерны проектирования в TypeScript», в которой будет представлен ряд полезных паттернов проектирования в веб-разработке с использованием TypeScript.

Вот ссылки на опубликованные статьи:

Паттерн Стратегия в TypeScript

Паттерн Цепочка Обязанностей в TypeScript

Паттерн Наблюдатель в TypeScript

Паттерн Шаблонного метода в TypeScript

Паттерн Адаптер в TypeScript

Паттерн Фабричного метода в TypeScript

Паттерн Абстрактной фабрики в TypeScript

Паттерны проектирования очень важны для веб-разработчиков, которые освоив паттерны становятся способными улучшить качество написания кода. В этой статье я буду использовать TypeScript, чтобы рассказать о паттерне Стратегия.

Важным функционалом веб-приложений является регистрация и вход в систему. Для регистрации в веб приложении наиболее распространенным способом является использование учетной записи/пароля, электронной почты или номера мобильного телефона.

Регистрация и вход в систему являются важными функциями веб-приложений. При регистрации веб-приложения более распространенным способом регистрации является использование учетной записи/пароля, электронной почты или мобильного телефона. И после успешной регистрации пользователь может использовать соответствующий метод для входа (аутентификации).

Иногда веб -приложение может поддерживать другие методы входа, например, в дополнение к методу входа по электронной почте, страница входа на Medium также поддерживает методы входа сторонних платформ, таких как Google, Facebook, Apple и Twitter.

В дальнейшем, для поддержки новых сторонних методов входа нам нужно будет снова и снова изменять и дополнять существующую функцию, которая обслуживает процесс входа в веб-приложение:

Если мы продолжим и далее добавлять или изменять методы входа в систему, мы обнаружим, что функцию входа в систему становится все труднее поддерживать. И для решения данной проблемы можно использовать паттерн Стратегия, в котором мы сможем инкапсулировать различные методы входа в веб-приложение в разные стратегии входа.

Чтобы лучше понять код, который я приведу далее, давайте сначала посмотрим на соответствующую диаграмму UML:

На приведенном выше рисунке мы определяем интерфейс Strategy, а затем на основе этого интерфейса Strategy реализуем две стратегии входа - через Twitter и через учетную запись/пароль.

Интерфейс Strategy

Twitter Strategy класс

LocalStrategy класс

После того как мы описали различные стратегии входа в систему, мы определяем класс Authenticator для переключения между различными стратегиями входа и выполнения соответствующих операций аутентификации.

Класс Authenticator

После этого мы можем использовать различные методы входа для аутентификации пользователя с помощью следующего кода:

Когда вы запустите приведенный выше код, вы получите соответствующий ему вывод, показанный ниже на рисунке:

Кроме задачи аутентификации при входе паттерн Стратегия можно использовать при решении задач проверки полей формы. А также - для оптимизации проблем со слишком большим количеством ветвей if else.

Если вы используете Node.js для разработки сервисов аутентификации, обратите внимание на модуль passport.js:

Этот модуль очень мощный и в настоящее время поддерживает до 538 стратегий аутентификации:

Итак, подытожим сценарии использования паттерна Стратегия:

Когда системе необходимо динамически выбирать один из нескольких алгоритмов, каждый алгоритм может быть инкапсулирован в отдельный, созданный для него, класс стратегии.

Когда несколько классов отличаются только поведением, и вы сможете использовать паттерн Стратегия для динамического выбора конкретного поведения, которое будет выполняться во время выполнения.

Если у вас есть какие-либо вопросы, пожалуйста, пишите мне. В дальнейшем я продолжу знакомить вас с другими паттернами, и если вам интересно, подпишитесь на меня в Medium или Twitter.