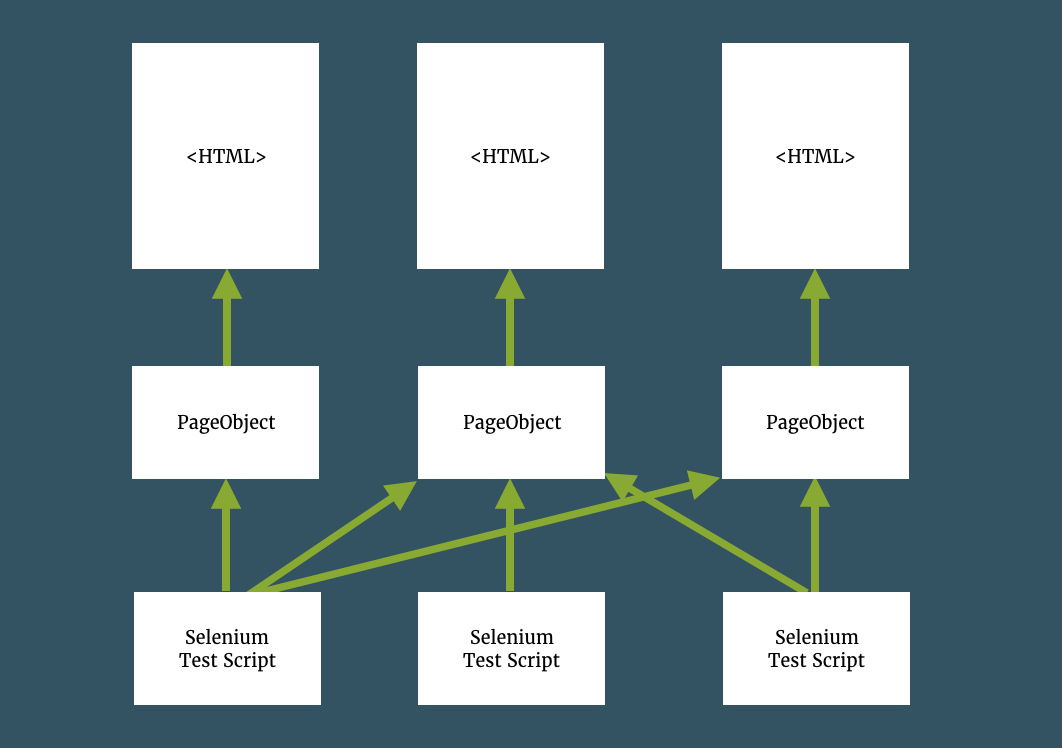
Page Object широко используется в автоматизированном тестировании и позволяет разделять логику выполнения тестов от их реализации. Изначальный смысл Page Object состоял в том, чтобы инкапсулировать (то есть прятать) логику работы с элементами. Тесты не должны ничего знать о веб-элементах, не должны оперировать напрямую с XPath или другими селекторами. Тесты должны использовать публичные методы Page объекта.

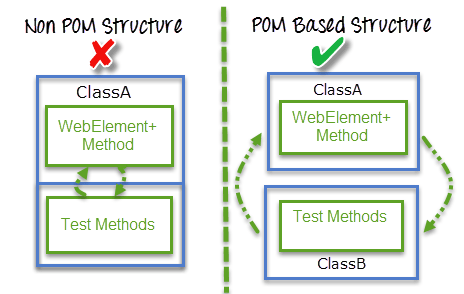
**Объектная модель страницы или POM**   − это шаблон проектирования или framework, который мы используем в Selenium, с помощью которого можно создать репозиторий объектов различных веб-элементов в приложении. Простыми словами **POM - это наиболее широко используемый шаблон проектирования в сообществе Selenium, в котором каждая веб-страница (или важные страницы) рассматривается как отдельный класс**. Для каждого из этих классов страниц (то есть объектов страницы) вы можете определить элементы этой страницы и определенные методы для этой страницы. Каждый объект страницы представляет страницу веб-страницы или приложения. Это слой между тестовыми сценариями и пользовательским интерфейсом, который инкапсулирует функции страницы. Он помогает сделать код **более читабельным, поддерживаемым** и **повторно используемым.**

В структуре объектной модели страницы мы создаем файл класса для каждой веб-страницы. Этот файл состоит из различных веб-элементов, присутствующих на веб-странице. Более того, сценарии тестирования затем используют эти элементы для выполнения различных действий. Поскольку веб-элементы каждой страницы находятся в отдельном файле класса, код становится простым в обслуживании и сокращается дублирование кода.

Page Object моделирует страницы тестируемого приложения в качестве объектов в коде. В результате его использования у вас получатся отдельные классы, отвечающие за работу с HTML каждой конкретной веб-страницы. Такой подход значительно уменьшает объем повторяющегося кода, потому что одни и те же объекты страниц можно использовать в различных тестах. Основное преимущество Page Object заключается в том, что в случае изменения пользовательского интерфейса, можно выполнить исправление только в одном месте, а не исправлять каждый тест, в котором этот интерфейс используется.  
Класс PageObject не обязательно должен представлять собой всю страницу. Он может быть частью страницы, которая часто используется на сайте (или даже на одной странице). Это может быть, например, пагинатор или меню. Основной принцип состоит в том, что есть только один класс в Вашем проекте, который знает о структуре HTML конкретной страницы или ее части.

**Преимущества объектной модели страницы**

К преимуществам объектной модели страницы можно отнести:

* ***Легко обслуживать.***После того, как вы внесли изменения в любой локатор или функциональные изменения на веб-странице, вы точно знаете, где изменить свой код.
* ***Чистый и понятный код.***Вместо того, чтобы использовать один и тот же код во всей тестовой среде, многие методы определены в объектах страницы. Следовательно, количество строк написанного кода сокращается, и ваш код становится более понятным и читабельным.
* ***Помогает указать на пользовательский интерфейс.***Поскольку вы создаете классы страниц почти для каждой страницы на своей веб-странице, легко понять функциональность и модель вашей веб-страницы даже с первого взгляда на ваши тестовые фреймворки.
* [](https://coderlessons.com/wp-content/uploads/images/gur/6d452ae3351f544d9a98f1cd9959fe1e.png)***Тестовые сценарии лучше.***Ваши тестовые сценарии будут содержать описательные имена методов (например, homepage.goToLoginPage ()) вместо непонятного и жестко запрограммированного кода (например, driver.findElement (By.id («login»)). Click ()).

***Разделение логики и реализации***

Существует большая разница между логикой тестирования (что проверить) и его реализацией (как проверить). Пример тестового сценария: «Пользователь вводит неверный логин или пароль, нажимает кнопку входа, получает сообщение об ошибке». Этот сценарий описывает логику теста, в то время как реализация содержит в себе такие действия как поиск полей ввода на странице, их заполнение, проверку полученной ошибки и т.д. И если, например, изменится способ вывода сообщения об ошибке, то это никак не повлияет на сценарий теста, все также нужно будет ввести неверные данные, нажать кнопку входа и проверить ошибку. Но это напрямую затронет реализацию теста — необходимо будет изменить метод получающий и обрабатывающий сообщение об ошибке. При разделении логики теста от его реализации автотесты становятся более гибкими и их, как правило, легче поддерживать.

***Page Object в Selenium***

Паттерн Page Object в Selenium реализован с помощью библиотеки PageFactory и класса страницы. Page Object представляет собой отдельный класс, содержащий локаторы элементов, методы для работы с ними и конструктор принимающий в качестве параметра объект WebDriver. Методы класса Page Object могут возвращать объекты других Page Object классов. С помощью этого можно воссоздать копию переходов и поведения веб-приложения. Например, метод успешной регистрации в классе RegistrationPage должен возвращать экземпляр HomePage, потому что после регистрации на сайте пользователя перенаправляет на домашнюю страницу. Одним из следствий такого подхода является то, что необходимо моделировать как успешные, так и неуспешные методы. Или, например, в случае если нажатие на элемент может открывать различные страницы в зависимости от условий, то также необходимо создавать разные методы для каждого необходимого случая:

|  |
| --- |
| public class RegistrationPage {        private WebDriver driver;        public RegistrationPage(WebDriver driver) {          this.driver = driver;      }        public HomePage registerUserSuccess(User user) {          // успешная регистрация и переход на домашнюю страницу      }        public RegistrationPage registerUserError(User user) {          // регистрация пользователя с неверно заполненными полями          // вывод ошибки, остаемся на той же странице      }  } |

### **Page Object Model VS Page Factory in Selenium**

|  |  |
| --- | --- |
| ***Page Object Model*** *(паттерн/шаблон проектирования, который используется в автоматизированном тестировании)* | ***Page Factory*** *(класс из библиотеки Selenium, который используют для инициализации веб-элементов, определенных в объекте страницы )* |
| Finding web elements using **By** | Finding web elements using **@FindBy** |
| POM does not provide lazy initialization | Page Factory does provide lazy initialization |
| Page Object Model is a design pattern | PageFactory is a class that provides the implementation of the Page Object Model design pattern |
| In POM, one needs to initialize every page object individually | In PageFactory, all page objects are initialized by using the **initElements()** method |

Page Object Model (POM) в тестировании — концепция не новая. И если в одних случаях она целесообразна, то в других приводит к дополнительным расходам.

**Основная идея POM**

Концепция основана на том, что каждый UI-элемент приложения имеет отдельную реализацию методов. Это позволяет повторно использовать код и легко менять его случае необходимости.

Кроме того, наличие тестов, написанных согласно этому шаблону, позволит нам создавать легко читаемый код, который будут хорошо понимать новые члены команды, что сэкономит время.

В конце концов, при правильном использовании концепции POM автоматизированное тестирование становится доступным для нетехнических членов команды, что тоже немаловажно. Нельзя также не упомянуть и то, что существует большой потенциал при совершенствовании тестовой инфраструктуры.