



# Selenium

Difference between  
POM(Page Object Model)  
and Page Factory.



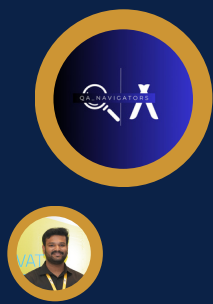
# Selenium

## POM (Page Object Model)

**Structure:** POM is a design pattern that creates an object repository for web elements. Each web page is represented as a class, and each class contains methods to interact with the web elements on that page.

**Elements:** In POM, you define web elements using **By locators**, and you have to initialize them using `driver.findElement()` or `driver.findElements()`.

```
public class LoginPage {  
    WebDriver driver;  
  
    By username = By.id("username");  
    By password = By.id("password");  
    By loginButton = By.id("loginButton");  
  
    public LoginPage(WebDriver driver) {  
        this.driver = driver;  
    }  
  
    public void enterUsername(String user) {  
        driver.findElement(username).sendKeys(user);  
    }  
  
    public void enterPassword(String pass) {  
        driver.findElement(password).sendKeys(pass);  
    }  
}
```



# Selenium

## Page Factory

**Structure:** Page Factory is an implementation of POM. It provides a more concise way to initialize web elements.

**Elements:** In Page Factory, you use annotations like **@FindBy** to define web elements, and Page Factory initializes them for you using **PageFactory.initElements()**.





# TestNG

## Page Factory

Example:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
```

```
public class LoginPage {
    WebDriver driver;
```

```
    @FindBy(id = "username")
    WebElement username;
```

```
    @FindBy(id = "password")
    WebElement password;
```

```
    @FindBy(id = "loginButton")
    WebElement loginButton;
```

```
    public LoginPage(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }
```

```
    public void enterUsername(String user) {
        username.sendKeys(user);
    }
```