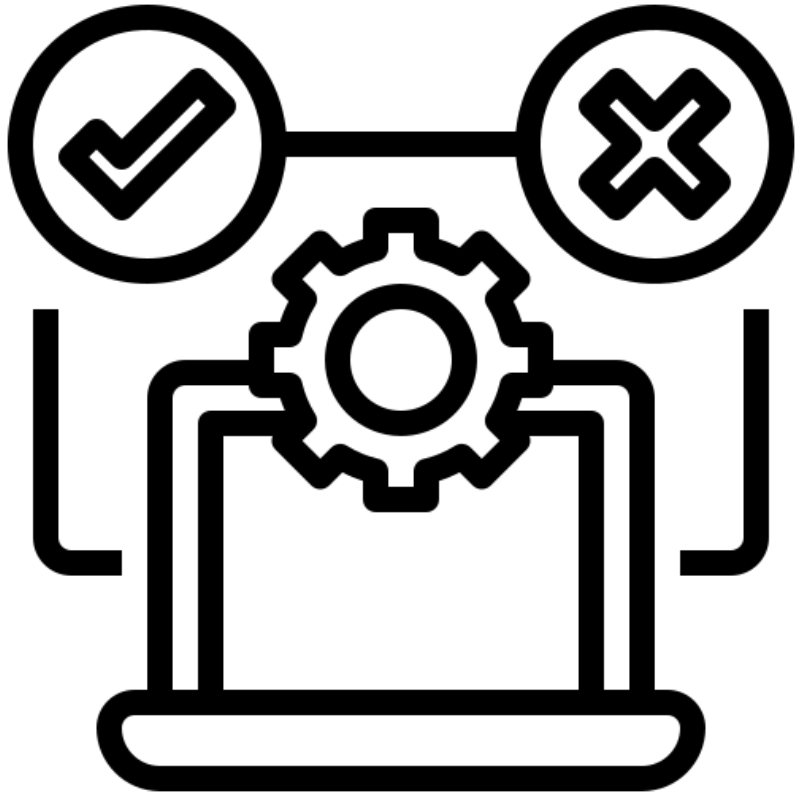
АВТОМАТИЗИРОВАННОЕ ТЕСТИРОВАНИЕ

Лекция 4: Шаблоны проектирования



БАШАРИНА ЕКАТЕРИНА АЛЕКСАНДРОВНА

Шаблоны проектирования

- Что такое шаблоны (паттерны) проектирования тестов? Зачем они нужны?
- Паттерны данных
- Структурные паттерны
- Паттерны взаимодействия

Шаблоны (паттерны)

- *наборы повторно используемых подходов и методов создания тестов*

- **Упрощение процесса тестирования**
- **Улучшение качества тестов**
- **Повышение производительности тестирования**

Как появилась идея?

- **Много тестов – много проблем**
- **Зачастую проблемы типовые!**
- **Для их решения требуется структурировать тесты и привести их к единообразию**
- **Одна проблема – один паттерн**

Зачем нужны паттерны?

- Сделать тесты более гибкими
- Сделать тесты более читаемыми
- Сделать тесты более надёжными
- Упростить поддержку тестов

Какие бывают паттерны?

- Паттерны данных (Data patterns)
- Структурные (Structural patterns)
- Паттерны взаимодействия (Involvement patterns)

Паттерны данных

- **Отделение работы с данными от тестовой логики**

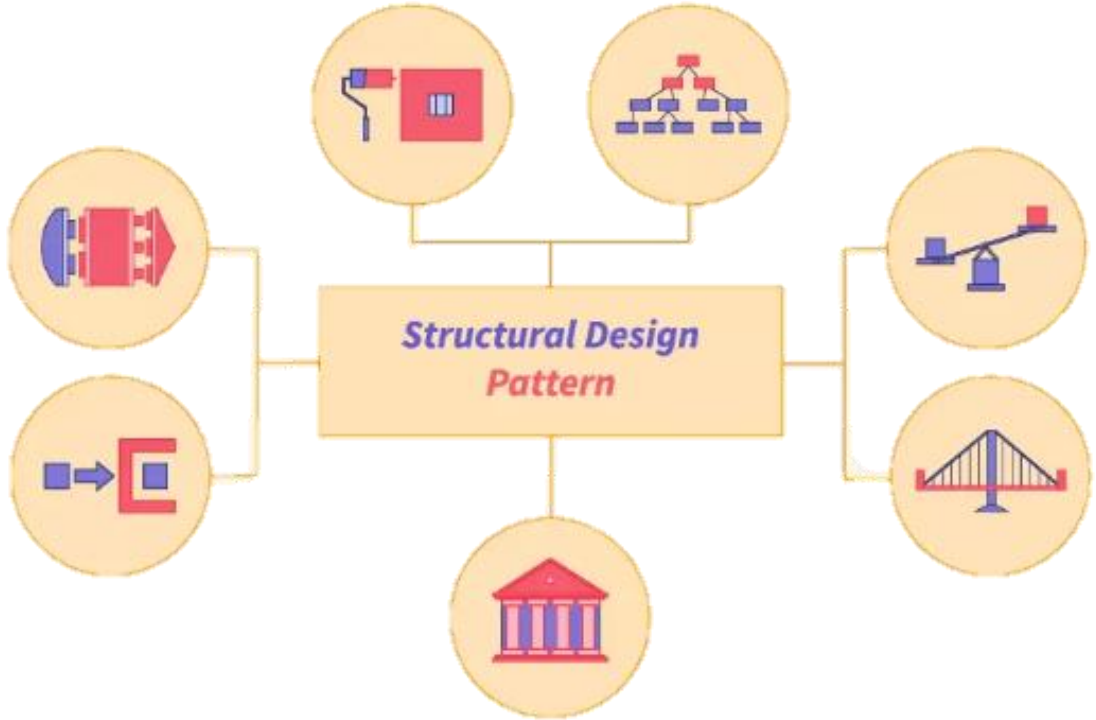
- **Value Object,**
  Builder,
  DataProvider,
  Assert Object/Matchers,
  …

```
public void registerUser(String name, String surname, int age,
                         Set<Role> roles, boolean paid) {
    enter(name, into( path: "name"));
    enter(surname, into( path: "surname"));
    enter( text: "" + age, into( path: "age"));
    chooseRoles(roles, paid);
}
```

```
public void registerUser(User user) {
    enter(user.getName(), into( path: "name"));
    enter(user.getSurname(), into( path: "surname"));
    enter(user.getAgeAsText(), into( path: "age"));
    chooseRoles(user.getRoles(), user.isPaid());
}
```

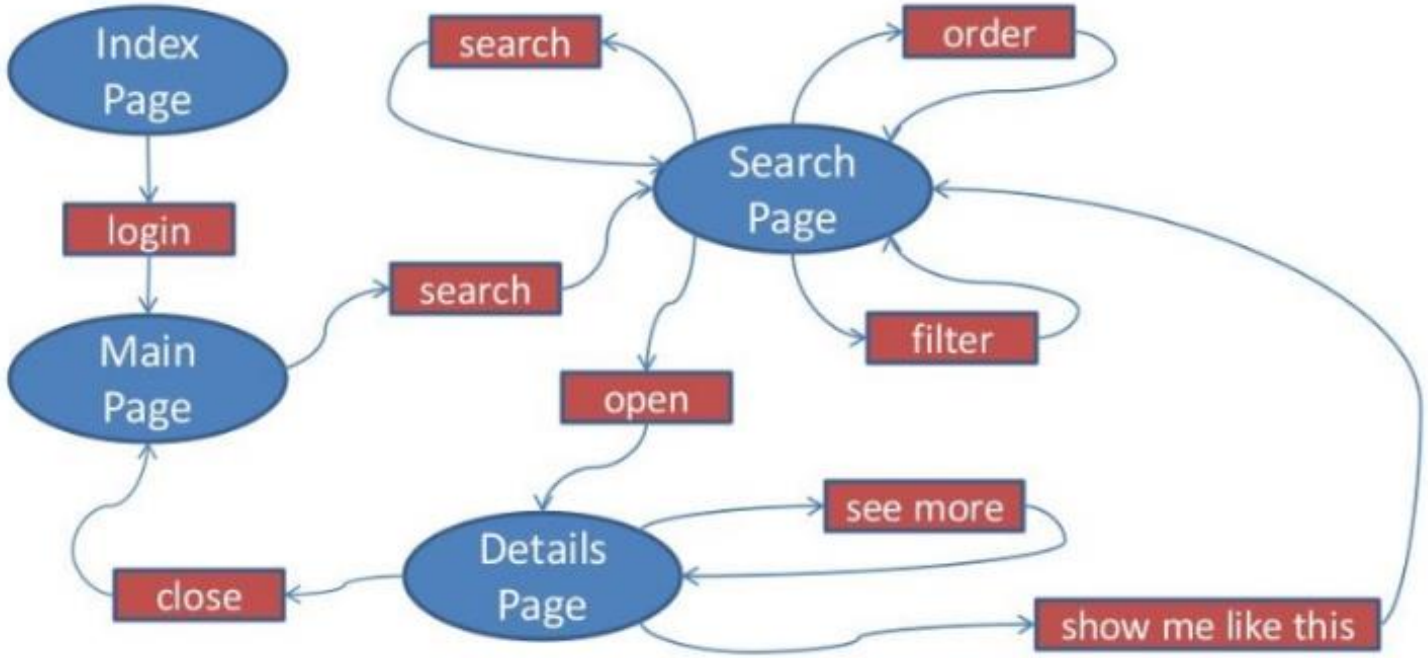
Структурные паттерны

- Структурирование кода тестов (привет, ООП!)
- Убираем дубликаты
- Инкапсулируем
- ~~Правим миром~~



PageObject

- **Идея: синхронизация кода с UI**
- **Разделение технических действий с логикой тестов**
- **Снова инкапсуляция!**



PageObject: код

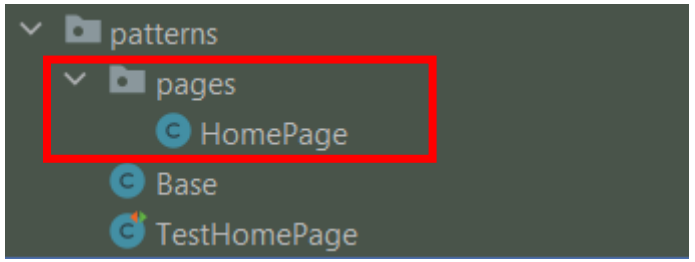
- **Аннотация @FindBy: отдельно находим веб-элементы, отдельно работаем с этими элементами**

- **Код самих тестов сокращается в разы**

```
public class testPatterns {
    private WebDriver driver;

    @FindBy(id = "username")
    private WebElement usernameInput;

    @FindBy(id = "password")
    private WebElement passwordInput;

    @FindBy(id = "login-button")
    private WebElement loginButton;

    public testPatterns(WebDriver driver) {    this.driver = driver;    }

    public String getUserName() {    return driver.findElement(By.id("user-name")).getText();    }

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

    public void enterPassword(String password) {    passwordInput.sendKeys(password);    }

    public void clickLoginButton() {    loginButton.click();    }

    @Test
    public void logintest() {
        enterUsername("some_name");
        enterPassword("qwerty");
        clickLoginButton();
        Assert.assertEquals(getUserName(), "some_name");
    }
}
```

PageObject: принцип

- **Каждый тип сущностей (например, веб-страница) выносится в отдельный пакет**



- **Каждая сущность внутри пакета (например, страница Home Page) описывается отдельным классом**

```
public class HomePage {

    private final WebDriver driver;
```

- **Для поиска элементов на странице используется аннотация @FindBy**

```
@FindBy(id = "name")
private WebElement name;

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

Паттерны взаимодействия

- **Приспосабливание тестов к восприятию неподготовленным человеком**

- **Переводим код в человеческий язык**

- **Разделяем код на атомарные элементы**

```
Feature: Addition
  In order to avoid silly mistakes
  As a math idiot
  I want to be told the sum of two numbers

  Scenario: Add two numbers
    Given I have entered 50 into the calculator
    And I have entered 70 into the calculator
    When I press add
    Then the result should be 120 on the screen
```

Основная идея

- **При написании тестов составляется логический сценарий, состоящий из понятных человеку шагов**

- **На каждом шаге описывается действие, которое необходимо совершить**

```
@Step("Navigate driver to {url}")
public void navigate(String url) {    home = new Home(driver, url);   }

@Step("Perform login with {name} and {password}")
public void login(String name, String password) {    home.login(name, password);   }

@Step("Switch to default content")
public void switchToDefaultContent() {    driver.switchTo().defaultContent();  }
```

Спасибо за внимание!