Автоматизация тестирования

SMART Framework

План лекции

- Принципы построения фреймворка
- Необходимость фреймворка
- Структура фреймворка
- Описание основных классов, которые должны присутствовать в любом фреймворке
- Использование фреймворка

SMART Framework: концепции

- Фреймворк – это такая организация проекта, которая позволяет упростить разработку, поддержку и модификацию программного кода.

- ✔ Page Object Pattern
- ✔ Конфигурируемость
- ✔ Независимость и стабильность тестов
- ✔ Data-driven Testing

SMART Framework: технологии

- Java (1.6)
- Maven
- TestNG
- ReportNG

Преимущества SMART Framework

- Гибкая конфигурация параметров фреймворка
- Поддержка популярных браузеров (IE, FF, Chrome, Opera, Safari)
- Простая адаптация для большинства веб-приложений
- Поддержка взаимодействия с БД и почтовыми серверами
- Интегрируемость с CI (Jenkins, Bamboo)
- Удобный и простой формат отчетности

Код теста без фреймворка

```
IWebDriver driver = new FirefoxDriver();
driver.navigate().to("http://www.google.com/");
IWebElement query = driver.findElement(By.Name("q"));
query.sendKeys("Cheese");
query.submit();
WebDriverWait wait = new WebDriverWait(driver,
TimeSpan.FromSeconds(10));
wait.until((d) => { return
d.title.toLower().startsWith("cheese"); });
System.console.writeLine("Page title is: " + driver.title);
driver.quit();
```

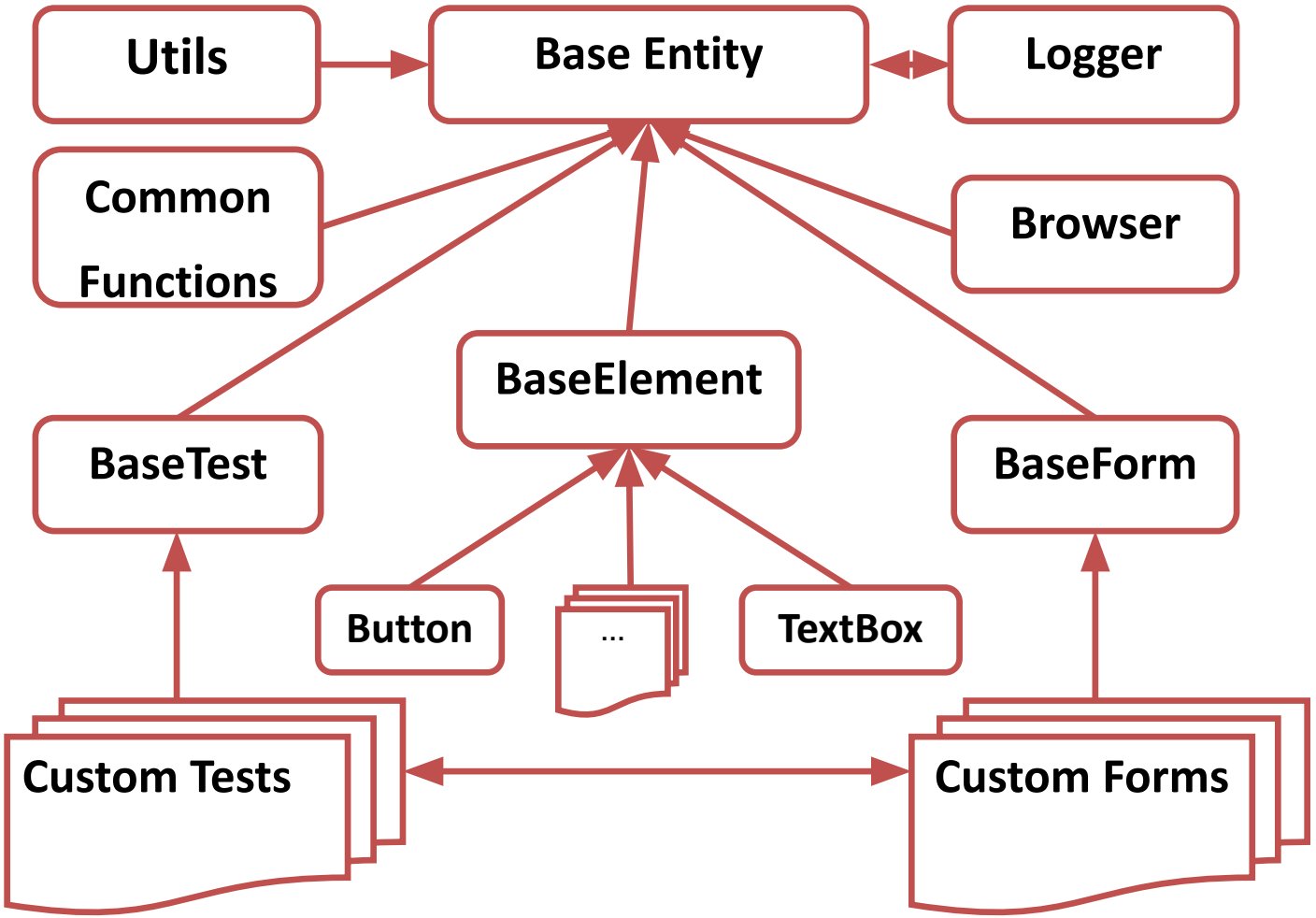
Код теста с фреймворком

```
logStep();
GooglePage search = new GooglePage();
 search. SearchByText(searchText);

logStep();
ResultsPage results = new ResultsPage();

logStep();
results. assertLinkIsPresent(searchText);
```

Структура SMART Framework



Java Framework: Base Entity

Базовый класс от которого наследуется большинство классов фреймворка. Описывает методы самого высокого уровня:

- ✔ @BeforeClass

public *void* **before**(ITestContext context)

- ✔ @AfterClass

public *void* **after**()

- ✔ protected *String* **makeScreen**(final Class<? extends BaseEntity> name*)*

Java Framework: Browser Factory

Класс выбора и инициализации экземпляра браузера.

Имплементирует Factory pattern

```
public static RemoteWebDriver setUp(final Browsers type){
...
case FIREFOX:
…
    FirefoxProfile ffProfile = new FirefoxProfile();
    try {
        JavaScriptError.addExtension(ffProfile);
        } catch (IOException e) {
        …
        }
    driver = new FirefoxDriver(new FirefoxBinary(),ffProfile,capabilitiesProxy);
    break;
    }
driver = new FirefoxDriver(capabilitiesProxy);
break;
```

Java Framework: Browser

Класс описывает взаимодействие с экземпляром браузера, расширяя возможности стандартного **Webdriver**

- ✔ *public static* **Browser getInstance**()
- ✔ *private static void* **initProperties**()
- ✔ *public RemoteWebDriver* **getDriver**()
- ✔ *public void* **navigate**(final String url)
- ✔ *public void* **waitForPageToLoad**()
- ✔ *public void* **refresh**()
- ✔ *public void* **selectNewWindow**()
- ✔ *public void* **exit**()

Java Framework: Logger

Класс применяется для реализации расширенного логирования.

Имплементирует Singleton pattern

- ✔ public static synchronized *Logger* ***getInstance()***
- ✔ public *void* **logTestName**(final String testName)
- ✔ public *void* **step**(final int step)
- ✔ public *void* **info**(final String message)
- ✔ public *void* **warn**(final String message)
- ✔ public *void* **error**(final String message)
- ✔ public *void* **fatal**(final String message)

Java Framework: Common Functions

Класс предоставляет набор статических функций общего назначения

- ✔ public static *String* **regexGetMatch**(*String* text, *String* regex)
- ✔ public static *String* **getCurrentDate**(*String* pattern)
- ✔ public static *String* **getTimestamp**()
- ✔ public static *String* **formatDate**(*Date* date, *String* pattern)
- ✔ public static *Date* **increaseDateByXDays**(final *Date* date, final *int* days)
- ✔ public static *String* **escapeMetaCharacters**(final *String* text)
- ✔ public static *void* **centerMouse()**
- ✔ public static *void* **awayMouse()**

Java Framework: Utils

Пакет Utils содержит набор вспомогательных классов:

- public **class DataBaseUtils** extends BaseEntity

✔ Описывает взаимодействие с базами данных

- public **class HttpUtils** extends BaseEntity

✔ Описывает работу с HTTP запросами

- public **class ImageMagicUtil** extends BaseEntity

✔ Описывает работы с графическими изображениями

- public **class MailUtils** extends BaseEntity

✔ Описывает взаимодействие с почтовыми серверами

Java Framework: Base Element

Абстрактный класс, описывающий базовые действия с элементом интерфейса приложения

- ✔ public *RemoteWebElement* **getElement()**
- ✔ public *boolean* **isEnabled()**
- ✔ public *By* **getLocator()**
- ✔ public *String* **getName()**
- ✔ public *void* **waitForIsElementPresent()**
- ✔ public *void* **sendKeys**(Keys key)
- ✔ public *void* **clickViaAction()**
- ✔ public *void* **clickExt()**
- ✔ public *void* **doubleClick()**

Java Framework: Base Form

Абстрактный класс, описывающий базовый методы работы с формами интерфейса приложения

```
/**
 * Contructor
 * @param locator Locator
 * @param formTitle Name
 */
protected BaseForm(final By locator, final String formTitle) {
init(locator, formTitle);
assertIsOpen();
}
```

Java Framework: Base Test

Абстрактный базовый класс теста, от которого наследуются все тесты. Класс содержит методы для старта и окончания теста.

public abstract *void* **runTest()**;

@Test
public void **xTest()** throws **Throwable**

- ✔ BaseTestParam
- ✔ BaseTestDataDriven

Использование фреймворка

```
public class LoginTest extends ProductiveBaseTest {

private UsersController users = UsersController.getInstance();

public void runTest() {
LoginForm loginForm = new LoginForm();
User admin = users.getUserByIndex(UserType.ADMIN, 2);

LogStep();
WelcomeForm welcomeForm =
loginForm.loginNewAccount(admin.getLogin(),
admin.getPassword());

LogStep();
welcomeForm.assertIsOpened();
}
```

Использование фреймворка

```
public class LoginForm extends BaseForm {

private final TextBox txbLogin = new
TextBox(By.id("inputUsername"), "Username");
private final TextBox txbPassword = new
TextBox(By.id("inputPassword"), "Password");
…
/**
 * Constructor by default
 */
public LoginForm() {
super(By.id("inputUsername"), "Login Page");
}
…
```

Использование фреймворка

```
/**
 * Performs login action for new accounts
 * @param user User Login
 * @param pwd User Password
 * @return Welcome Form
 */
public WelcomeForm loginNewAccount(final String user, final String pwd) {
    doLogin(user, pwd);
    return new WelcomeForm();
}
…
public void doLogin(final String user, final String pwd) {
    txbLogin.setText(user);
    txbPassword.setText(pwd);
    btnLogin.clickAndWait();
}
```

A1QA
Software Testing Company

Спасиб

О Contact us

5910 Countryard Drive, Ste. 170
Austin, TX 78731

info@a1qa.ru
www.a1qa.ru