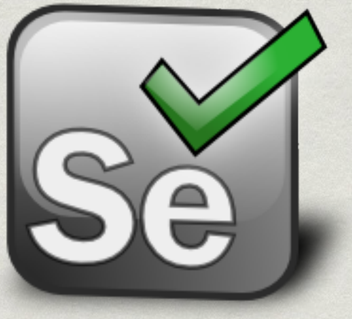
QA Battle

Типизированные элементы vs. нетипизированных

Alexei Vinogradov

Контекст

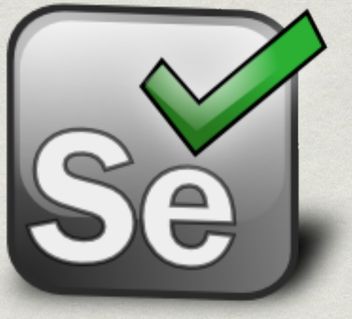


Selenium / WebDriver

не фреймворк для тестирования

… и никогда им не станет.

Контекст



Selenium / WebDriver

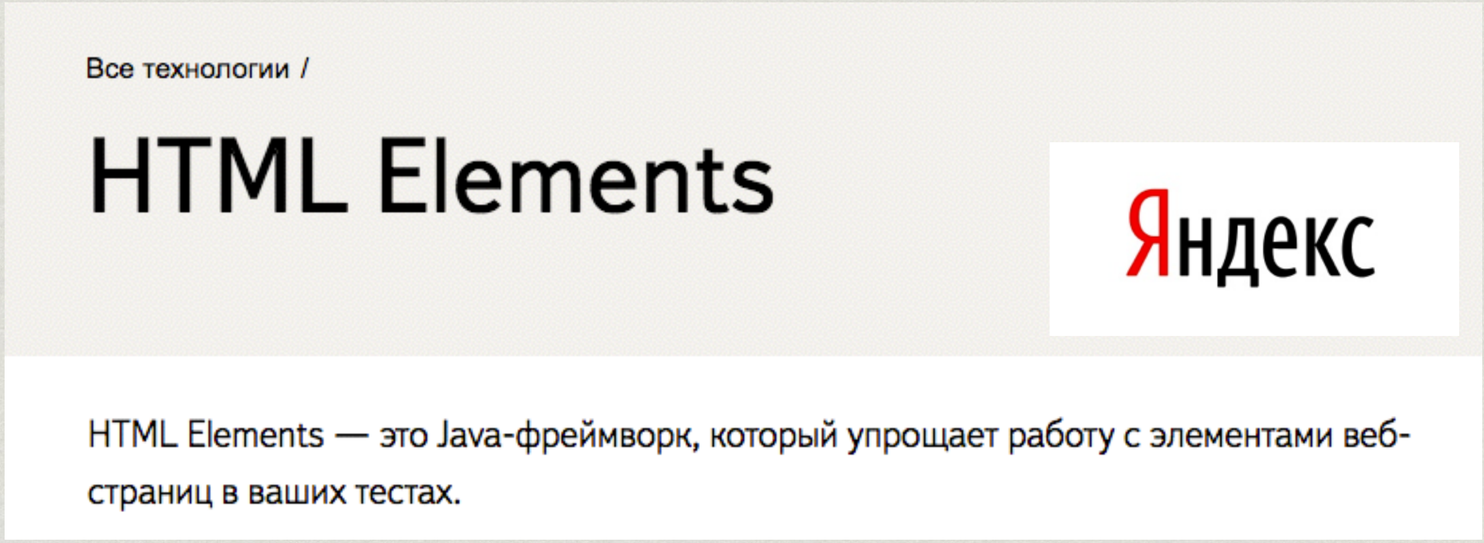
Набор инструментов для управления браузером

Контекст

Для эффективного написания и
исполнения UI тестов нам необходим
фреймворк для тестирования

Контекст

Например:



Контекст

Например:

<epam> | JDI

AUTOMATE THE WAY YOU TEST YOUR APPLICATIONS

JDI is a flexible, easily customizable framework that simplifies website implementation by extending the PageObjects design pattern with a variety of additional UI elements.

Extended PageObjects Pattern

More than 30 UI elements ready to use

Cross-platform & Cross-browser

Supports web, mobile, and desktop testing

Rich Test Run Information

Detailed logs and statistical reports

Multilanguage

Supports testing in Java and C#

Контекст

Например:

webtester

> This is the Java 6/7 optimized version of WebTester (v1.x), for the Java 8 version see 2.x.

testIT WebTester is a web-application UI test automation framework based on Selenium (http://www.seleniumhq.org).

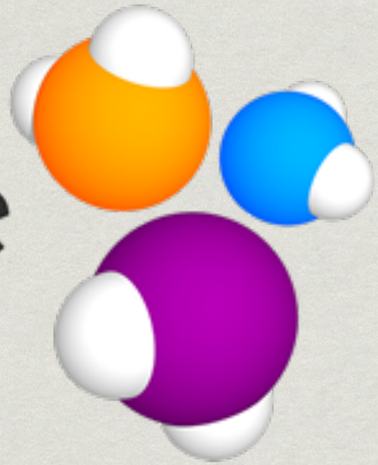
It is the product of years of consulting experience in various projects and aims at providing a very intuitive, declarative and extendable API for writing programmatic UI tests in Java.

Контекст

Например:

Selenide

CONCISE UI TESTS IN JAVA

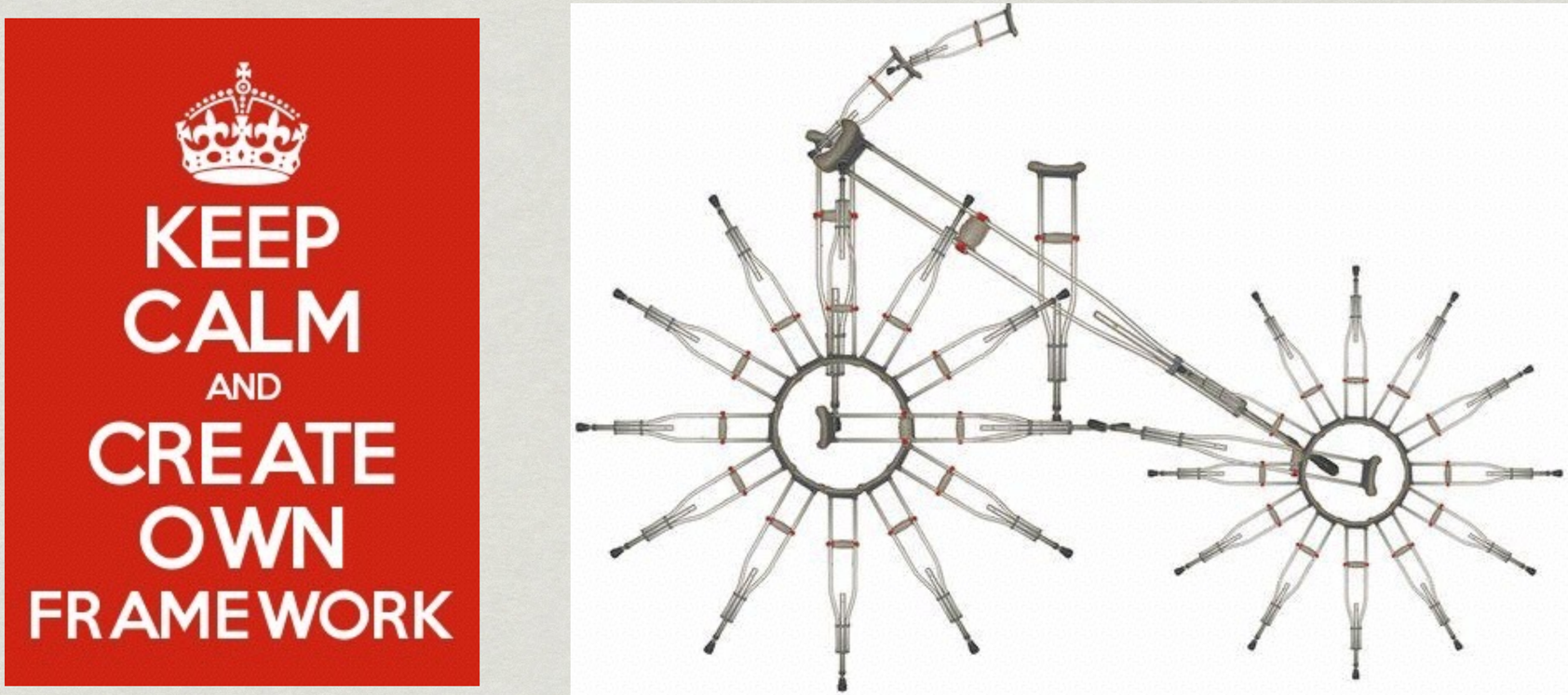


Selenium inside

http://selenide.org

Контекст

Наконец:



Контекст

Упор на веб-элементы:

HTML Elements — это Java-фреймворк, который упрощает работу с элементами веб-страниц в ваших тестах.

Features

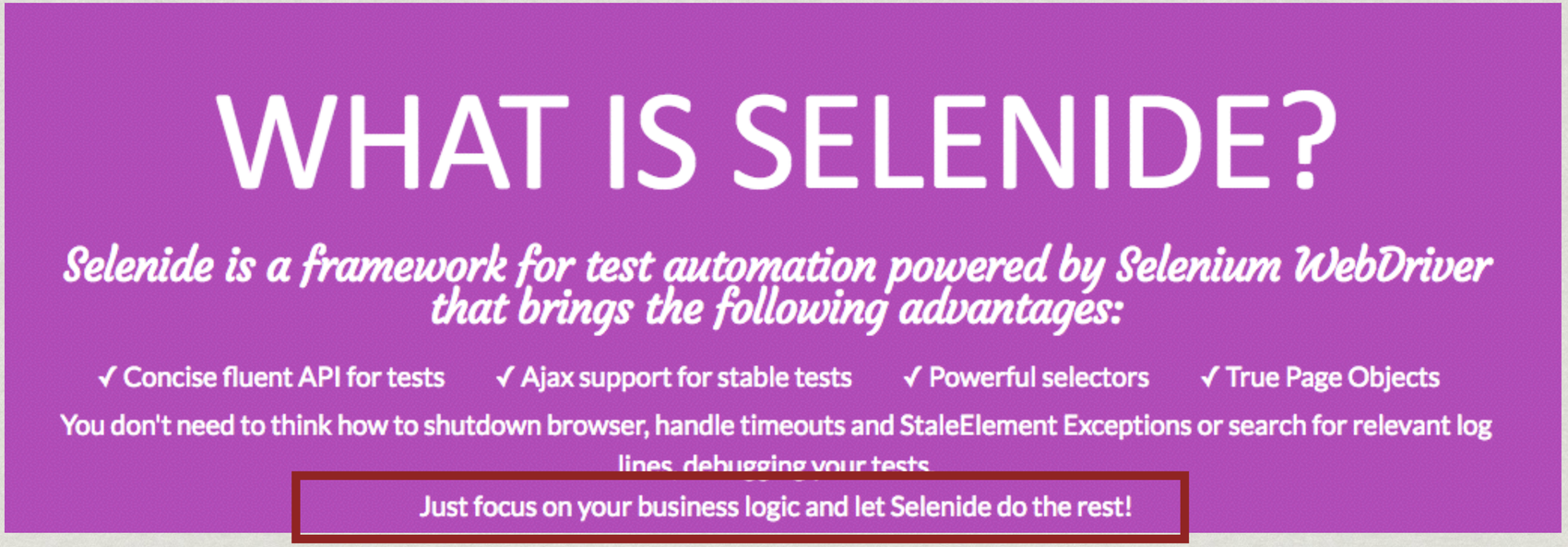
- Page Object Pattern as it's main focus
- Functional page elements instead of generic WebElement API
- Script style testing support using Ad-Hoc element identification API
- Event System for traceability and custom action

AUTOMATE THE WAY YOU TEST YOUR APPLICATIONS

JDI is a flexible, easily customizable framework that simplifies website implementation by extending the PageObjects design pattern with a variety of additional UI elements.

Контекст

За исключением



Just focus on your business logic and let Selenide do the rest!

Контекст

Нужны нам:

- Button
- Link
- Image
- Checkbox

или нет?

На примере тестирования Web & Java

The Battle

Many Types vs. one / few Supertypes

Краткая версия

Типы не нужны!

Типы приносят больше вреда, чем пользы!

Аргументы

Разработка типов стоит времени и денег

Аргументы

Взамен использование типов тоже стоит дороже

Проверка HTML кода

Идея - автоматически проверять, соответствие HTML-кода типу элемента

Например, если Button находится на странице, как <img>

Практика

Это что за элемент?

<a class=„fancy“ href=„javascript:doSmth()“>

Практика

А это что за элемент?

<input type=„button“>

Практика

А это? :)

Проверка соответствия элементов типу

Идея - как можно раньше замечать, если тип вызываемого элемента не соответствует определяемому

Unit Tests:

```
  @Test
  public void someActions(){
    Type1 withParam1=…;
    Type2 param2=…;
    callMethod(withParam1,param2);
  }
}
```

Работает! Часто ошибка заметна еще на стадии компиляции кода. Если Runtime - запуск быстрый.

Проверка соответствия элементов типу

UI Tests:

```
  @Test
  public void someActions(){
    Image picture=…;
    picture.getSource();
  }
}
```

Не работает! Ошибка только Runtime. Запуск - медленный.

Проверка соответствия элементов типу

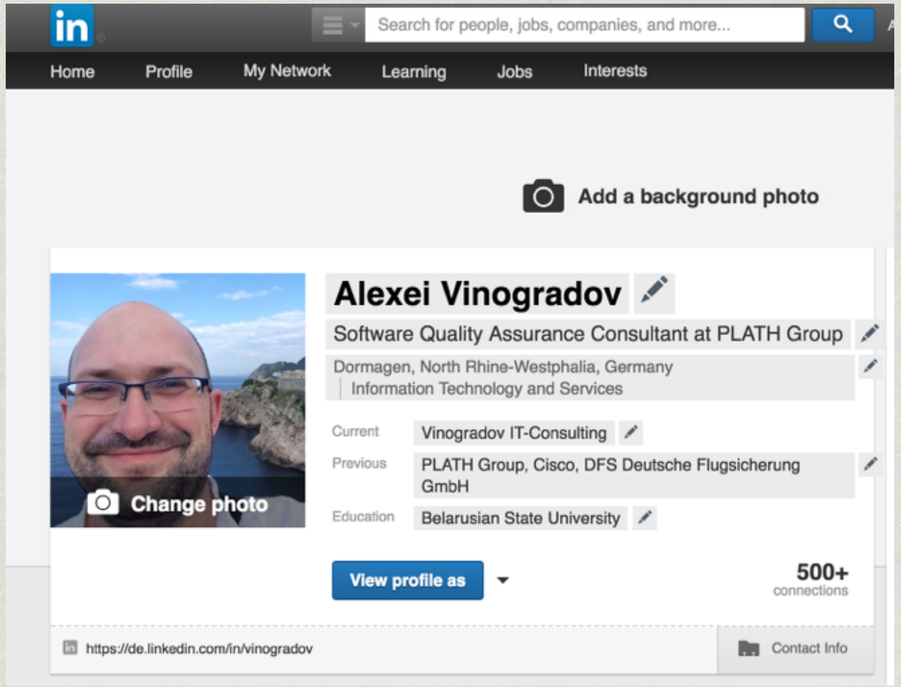
UI Tests:

```
  @Test
  public void someActions(){
    Button button=…;
    button.click();
  }
}
```

Совсем не работает! Click будет выполнен даже, если элемент не кнопка.

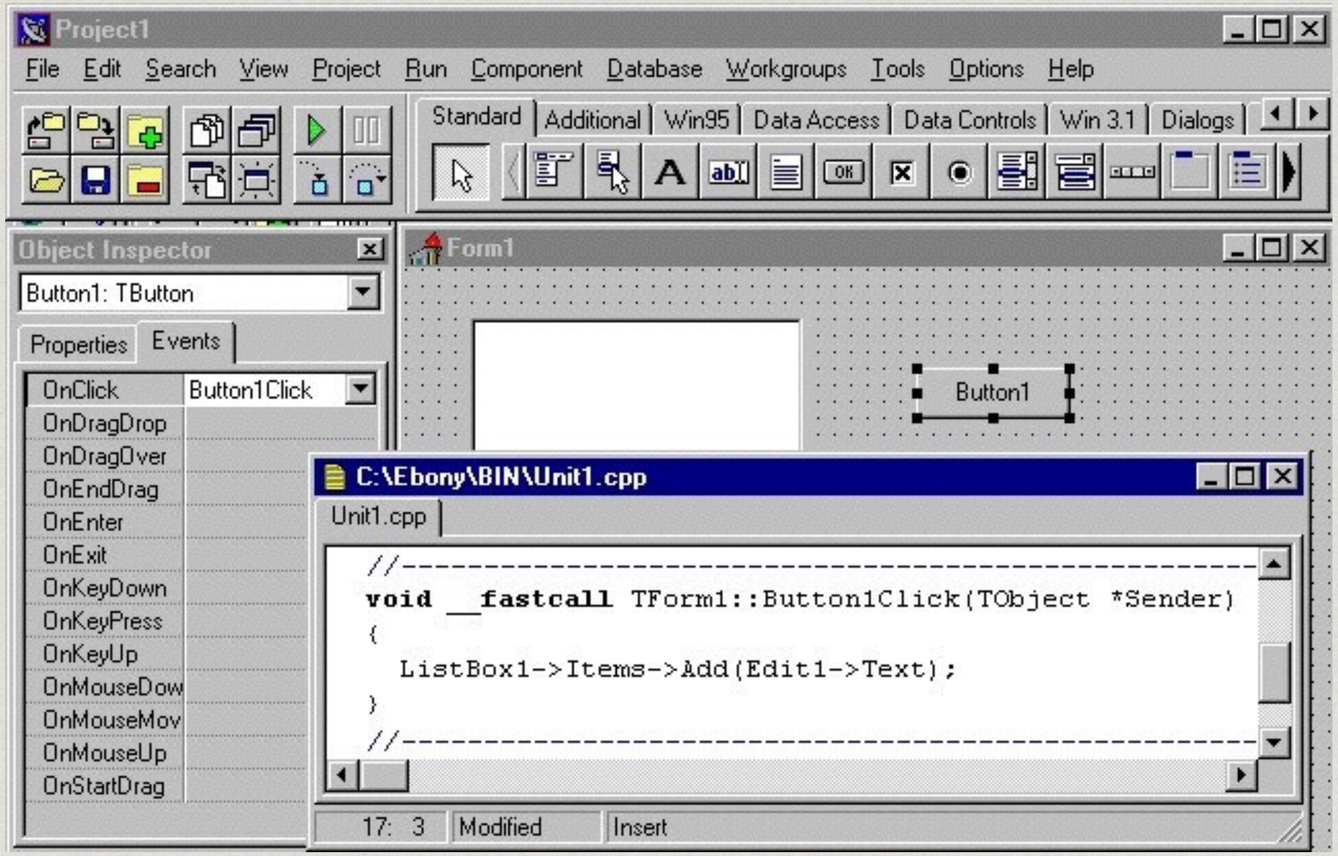
Элементы - гибриды

Элемент одновременно и Button/Link, и Image.
Нужен новый тип?



Боль выбора

Какой тип подходит моему элементу лучше?



Боль выбора

Какой тип подходит моему элементу лучше?

Link vs. Button (везде)

Dropdown vs. Droplist vs. Combobox vs Selector (JDI)

Функциональное UI тестирование

Все операции:

- кликнуть мышкой, ввести текст (95%)
- кликнуть правой клавишей мышки (4%)
- moveMouse/drag&drop/keyboard shortcuts (1%)

Плюс проверки:

- visible/invisible
- style/size
- text/attributes

Зачем тут типы?!


K.I.S.S.
PRINCIPLE


Keep It Simple Stupid

Функциональное UI тестирование

Нет разницы для функционального теста:

Link followMe;

или

Button followMe;

Тестируйте функциональность, а не техническую реализацию элементов:

```
  // method in PageObject
  public void followAccount(){
    followMe.click();
  }
}
```

Если очень хочется

```
Button submit;
Menu navigation;
…
Link followMe;
```

```
SelenideElement submitButton,
        navigationMenu,
        …
        followMeLink;
```

Без всякой потери,
даже наоборот:

```
@Test
public void someTest(){
    assert(myPage.submit)
          .isVisible();
}
```

```
@Test
public void someTest(){
    assert(myPage.submitButton)
          .isVisible()
}
```

визуально понятнее

Как писать UI тесты

Как писать UI тесты

```
// PageObjects and PageElements
public class SomePageElement {

  //UI Elements with Locators
  public static SelenideElement thing=$(locator),
                                …
                                something=$(locator),
                                anything=$(locator);


  //Actions for Page Objects
  public static void someAction(…){
    …
    something.setValue(..);
    thing.click();
    …
  }
…
}
…
```

Как писать UI тесты

```
public class SomeTest {

   @Test
   public void someChecking(){
      //actions
      SomePageObject.doSmth();
      …
      SomePageElement.doSmthElse();

      //asserts
      assert(SomePageElement.element).isVisible();
      …
   }
…
}
…
```

Как писать UI тесты

„Писать простые тесты легко и приятно“ (с)

4 стадии развития

1. Знаю как сделать
2. Знаю как сделать эффективно
3. Знаю как сделать изящно
4. Знаю как не делать

Вывод

Типизирование элементов - overengineering!

Тотальное типизирование элементов - зло!

Но…

Поддержка стандартных ситуаций - польза:

- Uploads
- Select/Dropdown
- Tables
- …

Поддержка составных элементов

- Uploads
- Select/Dropdown
- Tables
- …

В разных проектах - разные реализации (таблиц, пагинаторов и т.п.)

Действительно ли нужен вам общий тип для всех проектов?

Типизированные элементы

Дополнительный уровень абстракции - дополнительный источник ошибок

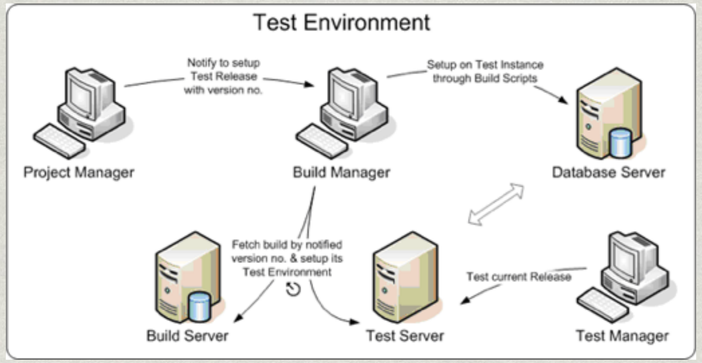
Баги в слоях UI тестов

Selenium Core

WebDrivers

Browsers

Ваш код тестов



Environment

Баги в слоях UI тестов

Вам мало?

Список популярных, всеми любимых фреймворков успешно реализовавших типизированные элементы:

The End.
Questions?

skype: alexejv
email: alexei@vinogradov-it.de
twitter: @vinogradoff

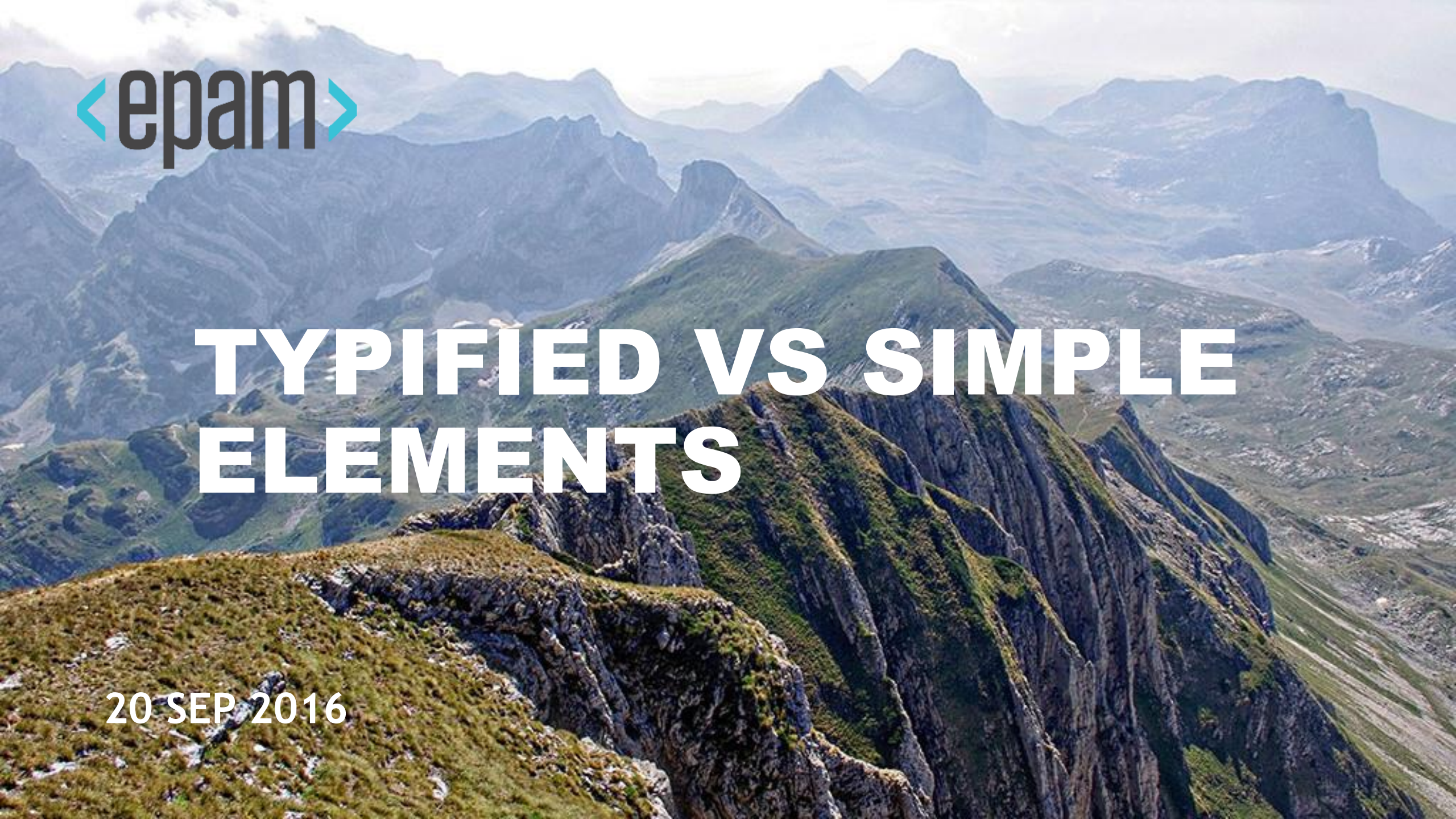
Маленькая поправка

WIND OF CHANGE

It is the very same project

<epam>

	Small Projects (< 4 months)	Big Projects (6+ months)
	+	−
	−	+


epam
TYPIFIED VS SIMPLE ELEMENTS
20 SEP 2016

ROMAN IOVLEV

Chief QA Automation

In Testing more than 11 years

In Testing Automation 9 years

TYPIFIED ELEMENTS VS SUPER ELEMENT

VS

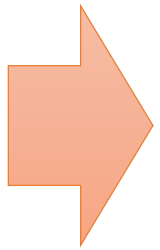
WebElement

We found 25 job openings for you — SORT BY:

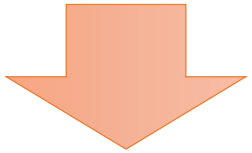
3D Artist	User Experience & Design	St-Petersburg, Russia	Apply »
Business Analyst (Mining Industry)	Consulting & Business Analysis	St-Petersburg, Russia	Apply »
Business Analyst/Consultant (OpenLink Endur)	Consulting & Business Analysis	St-Petersburg, Russia	Apply »
Front-end Developer (JavaScript Developer)	Software Engineering	St-Petersburg, Russia	Apply »
Infrastructure Automation Architect (Linux + Clouds)	Software Architecture	St-Petersburg, Russia	Apply »
Lead Java Developer	Software Engineering	St-Petersburg, Russia	Apply »

SIMPLE BUT…

```
$(“#submit”).click();
$(“#submit”).click();
$(“#submit”).click();
$(“#submit”).click();
$(“#submit”).click();
$(“#submit”).click();
$(“#submit”).click();
$(“#submit”).click();
```



```
@FindBy(how = How.CSS, using = “#submit")
private SelenideElement submitButton;
```



```
@FindBy(how = How.CSS, using = “#color-arrow")
private SelenideElement colorsArrow;
@FindBy(how = How.CSS, using = “#colors li")
private List<SelenideElement> colorsValues;
@FindBy(how = How.CSS, using = “#colors")
private SelenideElement colorsValue;

public void selectColor(String color) {
   colorsArrow.click();
   colorsValues.filter(text(color)).first().click();
}
public String getColor() {
   colorsValue.text();
}
```

3 DROPDOWNS

```
@FindBy(how = How.CSS, using = “#color-arrow")
private SelenideElement colorsArrow;
private List<SelenideElement> colorsValues = $$(“#colors li");
@FindBy(how = How.CSS, using = “#colors")
private SelenideElement colorsValue;
@FindBy(how = How.CSS, using = “#type-arrow")
private SelenideElement typesArrow;
@FindBy(how = How.CSS, using = “#types li")
private List<SelenideElement> typesValues;
@FindBy(how = How.CSS, using = “#types")
private SelenideElement typesValue;
@FindBy(how = How.CSS, using = “#shapes-arrow")
private SelenideElement shapesArrow;
@FindBy(how = How.CSS, using = “#shapes li")
private List<SelenideElement> shapesValues;
@FindBy(how = How.CSS, using = “#shapes")
private SelenideElement shapesValue;
```

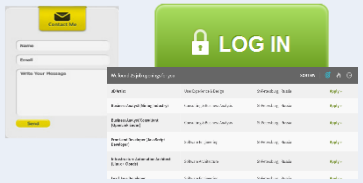
```
public void selectColor(String color) {
   colorsArrow.click();
   colorsValues.filter(text(color)).first().click();
}
public String getColor() {
   colorsValue.text();
}
public void selectType(String type) {
   typesArrow.click();
   typesValues.filter(text(type)).first().click();
}
public String getType() {
   typesValue.text();
}
public void selectShape(String shape) {
   shapesArrow.click();
   shapesValues.filter(text(shape)).first().click();
}
public String getShape() {
   shapesValue.text();
}
```

SO OBVIOUS

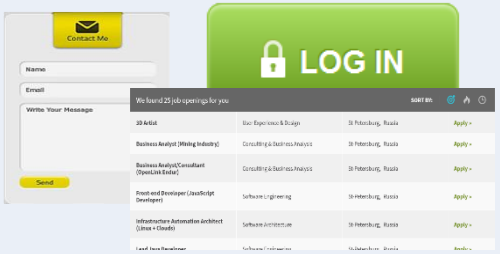
```
@JDropdown(expand= “#color-arrow“
                  list = “#colors li"
                  value = “#colors")
public Dropdown colors;

@JDropdown(expand= “#type-arrow“
                  list = “#types li"
                  value = “#types")
public Dropdown types;

@JDropdown(expand= “#shape-arrow“
                  list = “#shapes li"
                  value = “#shapes")
public Dropdown shapes;
```

	Actions
	Control
	Control Business

WIND OF CHANGE

	Small Projects (< 4 months)	Big Projects (6+ months)
	+	−
	−	+

SHORT & OBVIOUS

```
public class Filter {
    public Button submit;
    public Link followMe;
    public Menu navigation;
    public Dropdown colors;
    public Tabs areas;
    public Checklist settings;
    public ComboBox tags;
    public DropList shirtSizes;
    public Selector vote;
    public RadioButtons rating;

}
```

```
public class Filter {
    public WebElement submit;
    public WebElement followMe;
    public WebElement navigation;
    public WebElement colors;
    public WebElement areas;
    public WebElement settings;
    public WebElement tags;
    public WebElement shirtSizes;
    public WebElement vote;
    public WebElement rating;
}
```

ONLY USABLE ACTIONS

Button submitButton.

- click()
- getText()

SelenideElement submitButton.

click()	findElement()
isDisplayed()	getText()
is()	getCoordinates()
sendKeys()	$()
attr()	$$()
shouldHave()	append()
should()	clickContext()
getLocation()	closest()
find()	…

LOGS & REPORTING

- Click on Login **Button**
- Get Row with “Type=Approved” from Products **Table**
- Select “Blue” in Color **Dropdown**
- Login with “CorrectUser”
- …

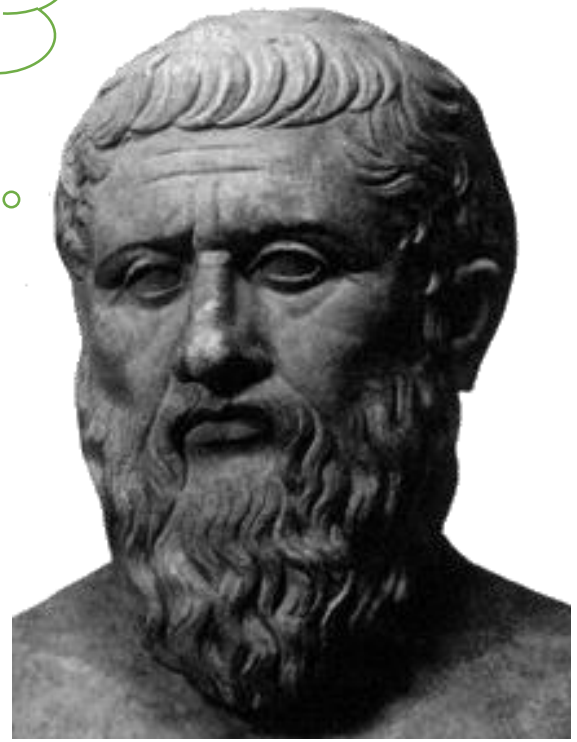
WORK WITH OBJECTS

- registerForm.save(**vipClient**);
- searchForm.find(**product**);
- List<**Product**> products = products.entities(p -> p.Type == “shoes”);
- **Card** products = cards.entity(c ->
  c.Status == *APPROVED* && c.Name.contains(“Duke”));
- …

```
ITable products;
ITable cards;
```

PLATO'S THEORY OF FORMS

IButton



SUBMIT

e-mail или телефон

NEW

My Drive

Shared with me

Recent

Google Photos

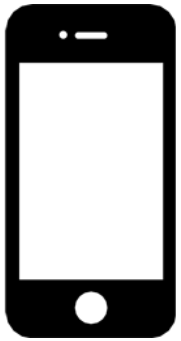
No application but you can write UI Objects (Page Objects)

INTERFACES

IButton submit;

IDropdown cities;

ITable products;

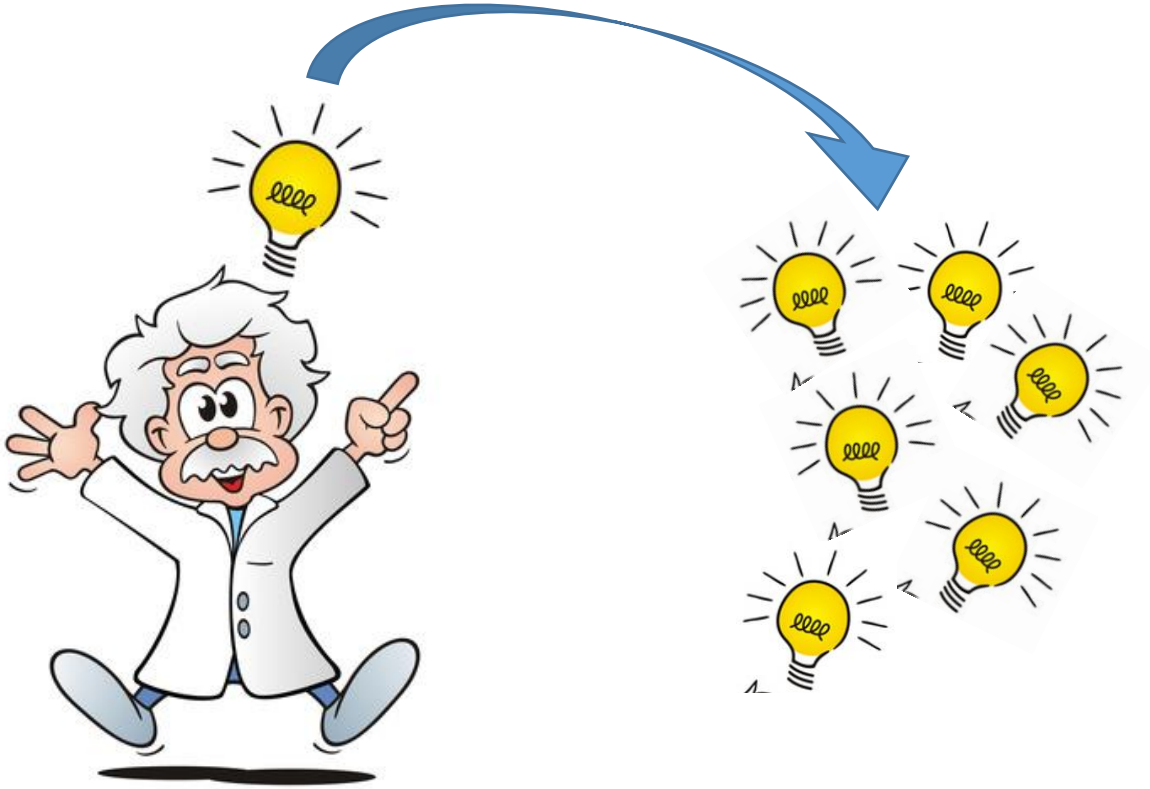


```
MapInterfaceToElement.update(
    new Object[][]{
        { IDropDown.class, MyDropDown.class},
        { IButton.class, MyButton.class},
        { ITable.class, CustomTable.class}
    });
```

EVERYBODY DO IT

1. Save a lot of time - money
2. More obvious test scenarios, page objects, logs, reports – easier develop & support
3. Operate with real business Objects
4. Write UI Objects (Page Objects) without app
5. Easier reuse scenarios on different platforms

BE LAZY



COME ∞ TOGETHER

JDI
UI Test Automation Framework

QUESTIONS

https://github.com/epam/JDI

INFO

http://jdi.epam.com/

https://vk.com/jdi_framework

CONTACTS

roman.Iovlev

roman_iovlev@epam.com