epam
OPEN SOURCE UI AUTOMATION ON C#
2 MARCH 2017

# ROMAN IOVLEV

Chief QA Automation

In Testing more than 11 years

In Testing Automation 9 years

# JDI ?

UI Test Automation Framework

# JDI SETUP

https://github.com/epam/JDI

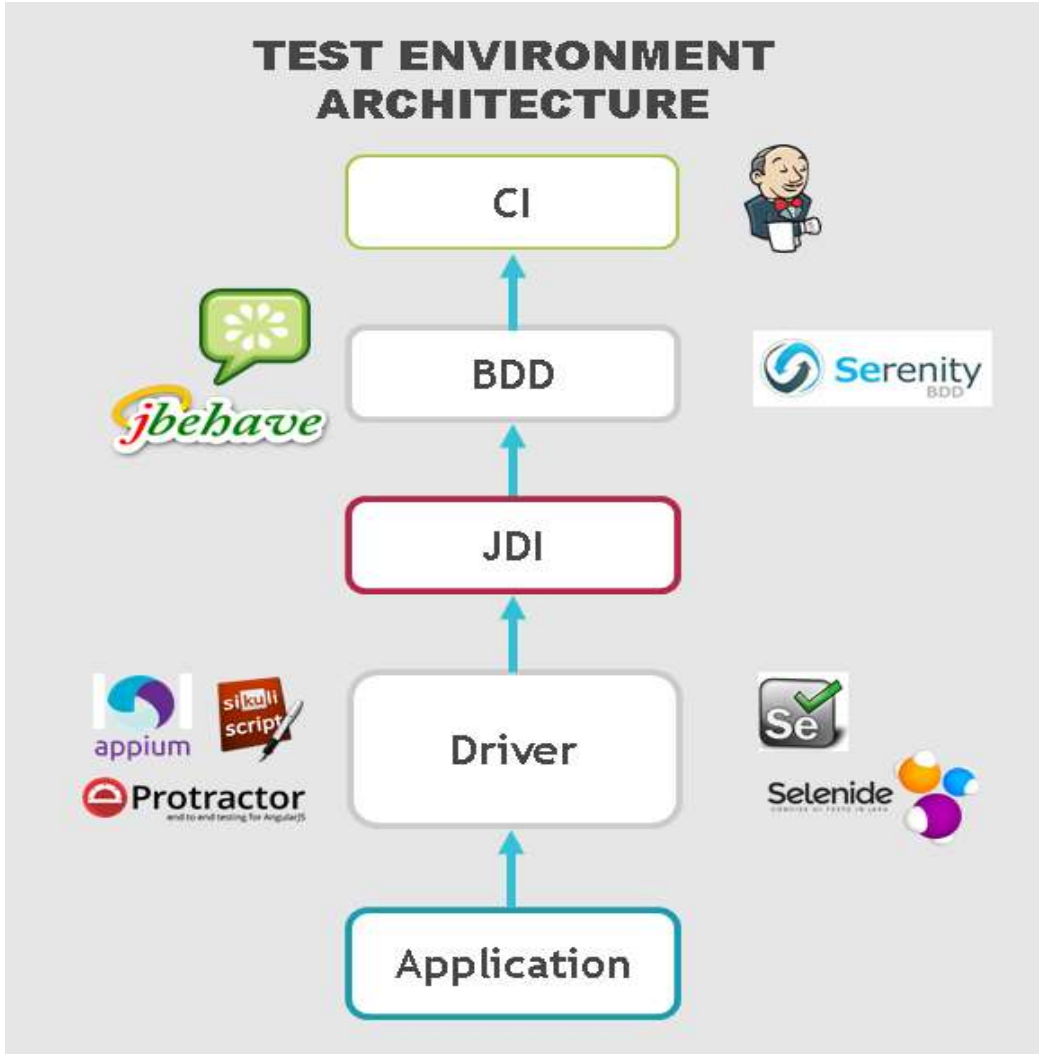
http://jdi.epam.com/

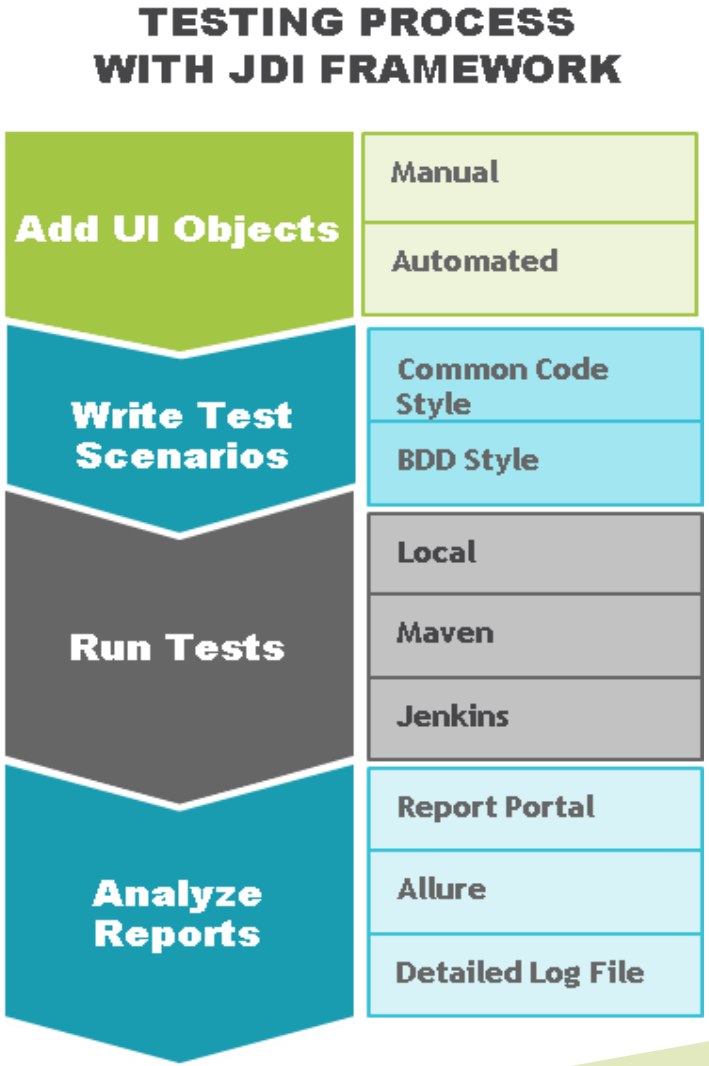
https://vk.com/jdi_framework

# README

JDI
UI Test Automation Framework

# FAQ


TEST ENVIRONMENT ARCHITECTURE
CI
BDD
JDI
Driver
Application
jbehave
Serenity BDD
appium
sikuli script
Protractor
Selenide


TESTING PROCESS WITH JDI FRAMEWORK
Add UI Objects
Manual
Automated
Write Test Scenarios
Common Code Style
BDD Style
Run Tests
Local
Maven
Jenkins
Analyze Reports
Report Portal
Allure
Detailed Log File

# SIMPLE TEST

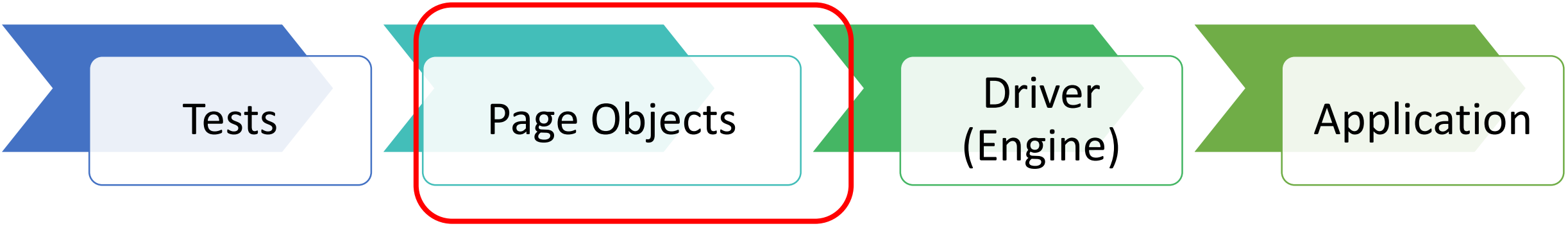
```
ProductPage.ProductType.Select(«jacket»);
ProductPage.Price.select(«500$»);
ProductPage.Colors.check(«black», «white»);
Assert.isTrue(ProductPage.Label.getText(),
                «Armani Jacket»)
```

# SIMPLE TEST

```
LoginPage.Open();
LoginPage.LoginForm.Login(admin);
SearchPage.Search.Find(«Cup»);
Assert.AreEqual(ResultsPage.Products.Count(), expected);
```

# PAGE ELEMENTS

# PAGE OBJECTS




**ELEMENTS**

```
private IWebElement UserName;
private IWebElement Password;
private IWebElement LoginButton;
```

- **ACTIONS**
  - EnterUserName(String name);
  - EnterPassword(String name);
  - ClickLoginButton();
- **BUSINESS ACTIONS**
  - Login(User user)

# PAGE ELEMENTS

## ELEMENTS

```
public TextField UserName;
public TextField Password;
public Button LoginButton;
```

- **ACTIONS**
- ---

- **BUSINESS ACTIONS**
- Login(User user)

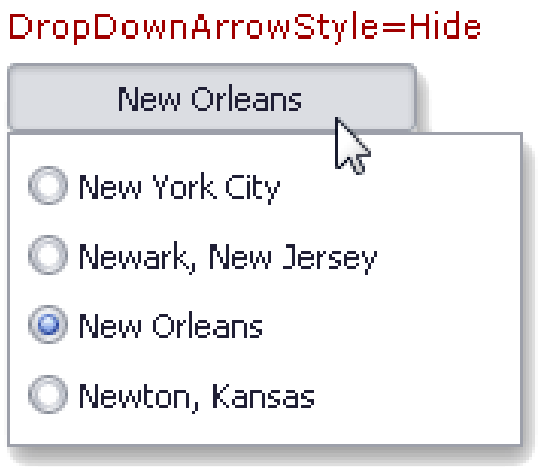
## ELEMENT

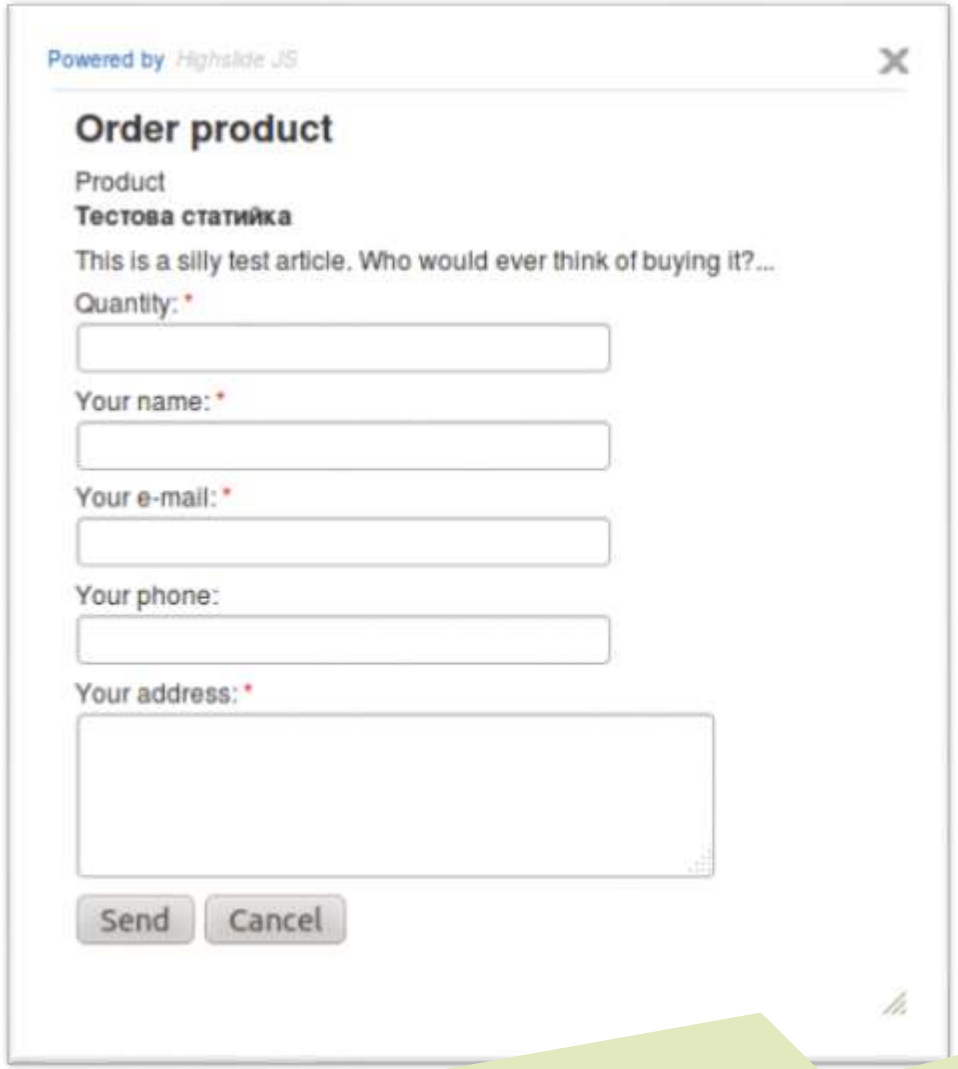
```
public DropDown UserStatus;
```

- **ACTIONS**
- Select(string option)
- boolean IsSelcted()
- List<String> AllOptions()
- string GetValue()

# ELEMENTS

- Simple
- Complex
- Composite

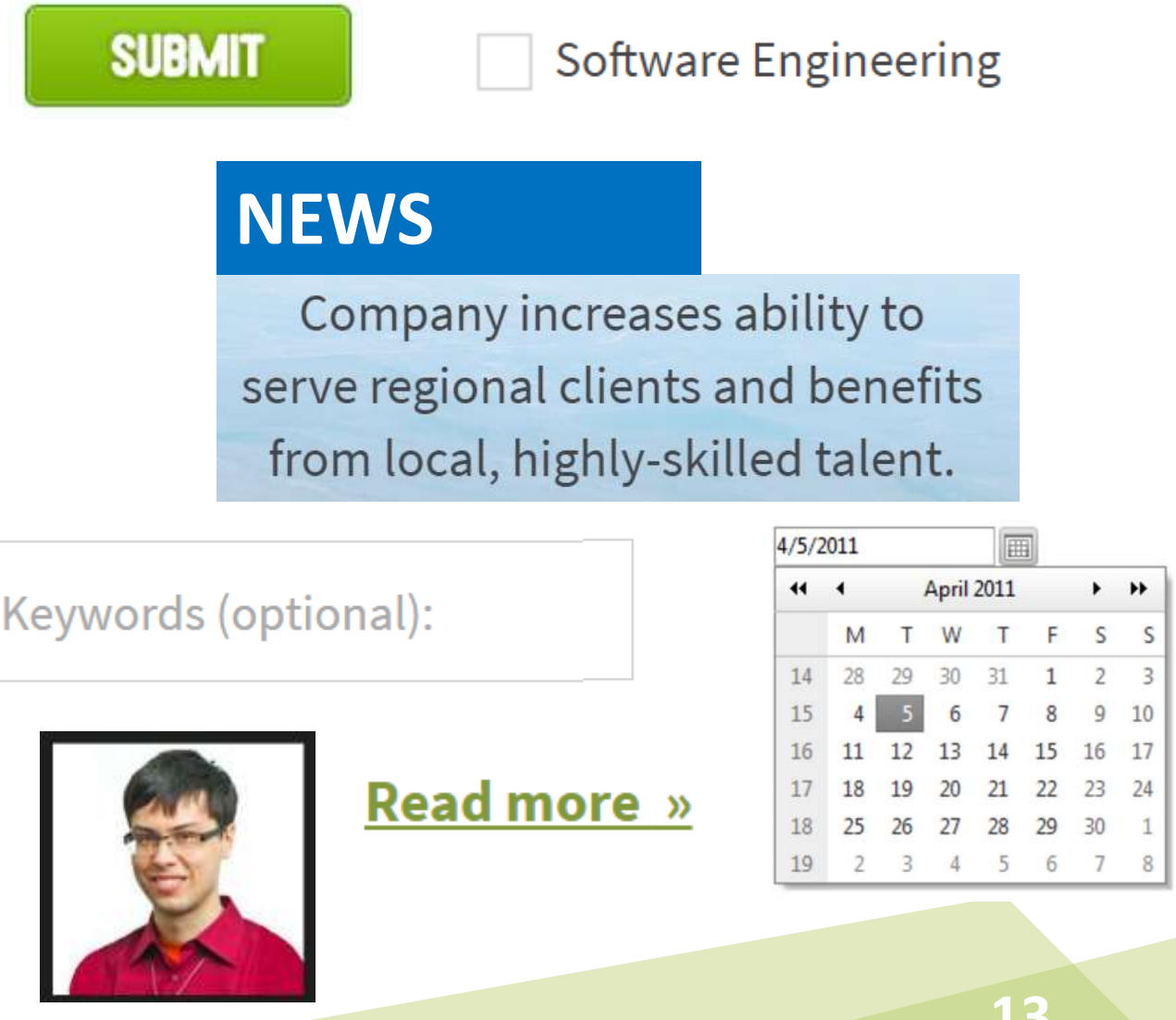


# SIMPLE ELEMENTS

# SIMPLE ELEMENTS

```
[FindBy (Css=".description")]
public Text Description;
public Button Submit;
public Label ProductName;
public Link FollowMe;
public TextField Password;
public TextArea Abuse;
public CheckBox RememberMe;
public DatePicker Date;
public FileInput Upload;
public Image Photo;
```



# SIMPLE ELEMENTS

```
[FindBy(Css=".btn") ]
 public Button submit;
```

```
[FindBy(Css=".btn") ]
 public Button submit =
        new Button(By.Css(".btn"));
```

```
[FindBy(Css=".btn") ]
 public IButton submit;
```

```
[FindBy(Css=".btn") ]
[FindBy(Xpath="//button") ]
[FindBy(Id="button-id") ]
[FindBy(Name="button") ]
```

# MULTILOCATORS

## Multi language testing

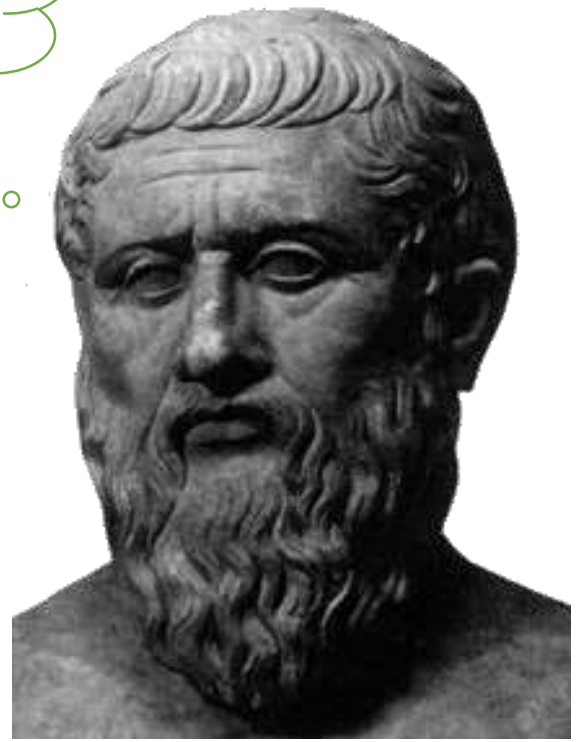
```
[JFindBy (text="Submit", group="en") ]
[JFindBy (text="Отправить" , group="ru") ]
public Button submit;
```

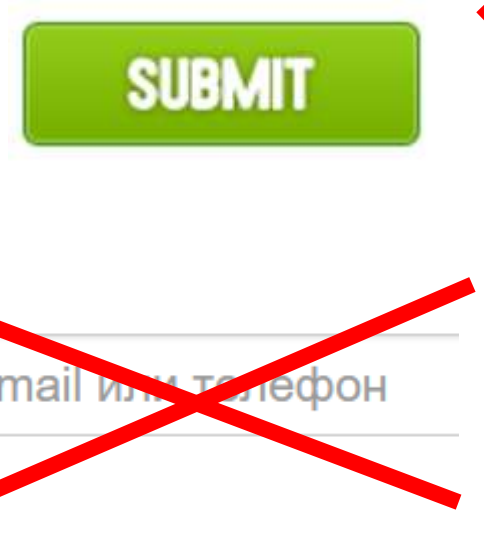
## Multi version testing

```
[JFindBy(text="Submit", group="1.7") ]
[JFindBy(value="Submit" , group="2.0") ]
public Button submit;
```

# PLATO'S THEORY OF FORMS

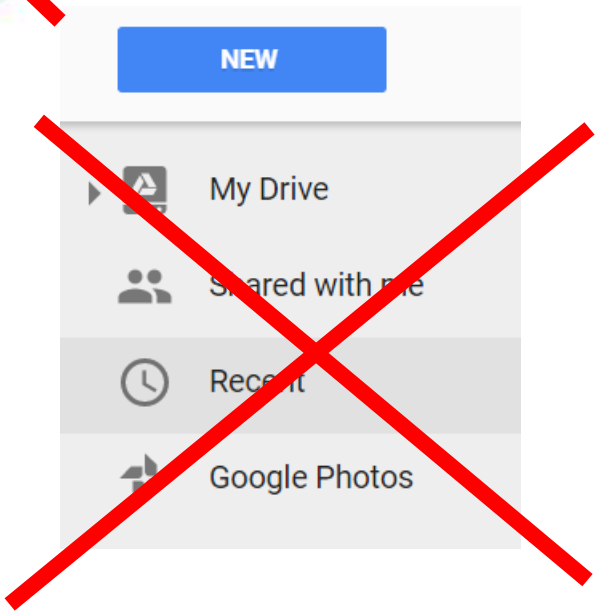
IButton












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

# COMPLEX ELEMENTS

# COMPLEX ELEMENTS

public **Dropdown** Colors;

public **Checklist** Settings;

public **ComboBox** Tags;

public **DropList** ShirtSizes;

public **List<*Element*>** SearchResults;

public **Elements** Reviews;

public **Table** Products;

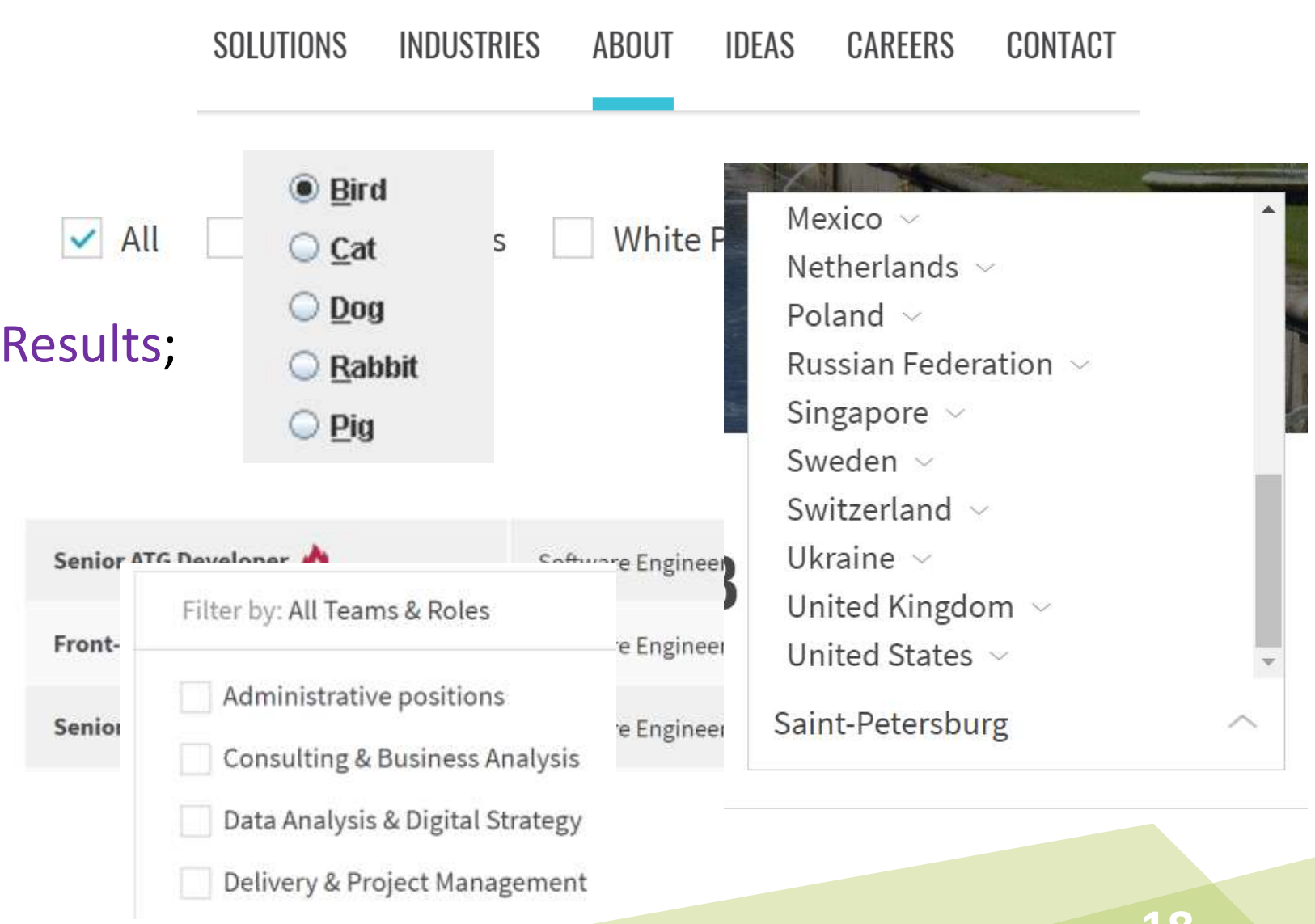
public **Menu** MainMenu;

public **Tabs** Areas;

public **Selector** Vote;

public **RadioButtons** Rating;

public **TextList** Chat;



# COMPLEX ELEMENTS

```
[FindBy (Css = “.colors") ]
public Dropdown colors = new Dropdown {
        Value = By.Css(".value"),
        List = By.TagName(“li")
};

[FindBy (Css = “.offers") ]
public Table offers = new Table {
        Row = By.Css(".value"),
        Column = By.TagName(“li"),
        Header = new [] {“ID", “Title", “Apply”}
};
```

# COMPLEX ELEMENTS

```
[FindBy (Css = ".colors") ]
public Dropdown Colors;

[FindBy (Css = ".table") ]
public Table Offers;


[FindBy (Css = ".menu li") ]
public Menu Navigation;
[FindBy (Css = ".menu ul") ]
public Menu Navigation;
[FindBy (Xpath = "//*[@class='menu']//li[text()='%s']") ]
public Menu Navigation;
```

# TYPIFIED ELEMENTS

- Code readability
- Clear behavior
- Union of all element’s locators
- Union of element and its actions
- Detailed logging

# COMPARE

```
Text Description;
Button Submit;
Label ProductName;
Link FollowMe;
TextField Password;
TextArea Abuse;
CheckBox RememberMe;
DatePicker Date;
FileInput Upload;
Image Photo;
```

```
IWebElement Description;
IWebElement SubmitButton;
IWebElement ProductName;
IWebElement FollowMeLink;
IWebElement PasswordField;
IWebElement Abuse;
IWebElement RememberMe;
IWebElement DatePicker;
IWebElement Upload;
IWebElement Photo;
```

# COMPARE

```
[FindBy(css = “.colors") ]
public Dropdown colors = new Dropdown {
    Value = By.Css(".value"),
    List = By.TagName(“li") };
```

---

```
[FindBy(Css = “.colors .value") ]
IWebElement ColorsValue;
[FindBy (Css = “.colors li") ]
List<IWebElement> ColorsList;

public string GetColor() {
    return ColorsValue.GetText();
}
```

```
public void SelectColor(String ColorName) {
    ColorsValue.Click();
    for (IWebElement Color : ColorsList)
        if (Color.getText().Equals(ColorName) {
            Color.Click();
            return;
    }
```

# COMPARE

```
[FindBy (Id = “trades") ]
public Table Colors;
```

```
[FindBy (Id = “…") ]
private List<IWebElement> ResultsColHeaders;
[FindBy (Id = “…") ]
private List<IWebElement> ResultsRowsHeaders;
[FindBy (Id = “…") ]
private List<IWebElement> ResultsCellsHeaders;
[FindBy (Id = “…") ]
private List<IWebElement> ResultsColumn;
[FindBy (Id = “…") ]
private List<IWebElement> ResultsRow;
```

ICell cell(Column column, Row row) { }
ICell cell(String columnName, String rowName) { }
ICell cell(int columnIndex, int rowIndex) { }
List<ICell> cells(String value) { }
List<ICell> cellsMatch(String regex) { }
ICell cell(String value) { }
ICell cellMatch(String regex) { }
MapArray<String, MapArray<String, ICell>> rows(String... colNameValues) { }
MapArray<String, MapArray<String, ICell>> columns(String... rowNameValues) { }
boolean waitValue(String value, Row row) { }
boolean waitValue(String value, Column column) { }
boolean isEmpty() { }
boolean waitHaveRows() { }
boolean waitRows(int count) { }
ICell cell(String value, Row row) { }
ICell cell(String value, Column column) { }
List<ICell> cellsMatch(String regex, Row row) { }
List<ICell> cellsMatch(String regex, Column column) { }
MapArray<String, ICell> row(String value, Column column) { }
MapArray<String, ICell> column(String value, Row row) { }
MapArray<String, ICell> row(int rowNum) { }
MapArray<String, ICell> row(String rowName) { }
List<String> rowValue(int colNum) { }
List<String> rowValue(String colName) { }
MapArray<String, ICell> column(int colNum) { }
MapArray<String, ICell> column(String colName) { }
List<String> columnValue(int colNum) { }
List<String> columnValue(String colName) { }
MapArray<String, SelectElement> header() { }
SelectElement header(String name) { }
List<String> headers() { }
List<String> footer() { }
List<ICell> getCells() { }
void clean() { }
void clear() { }
ITable useCache(boolean value) { }
ITable useCache() { }
Table clone() { }
Table copy() { }
ITable hasAllHeaders() { }
ITable hasNoHeaders() { }
ITable hasOnlyColumnHeaders() { }
ITable hasOnlyRowHeaders() { }
ITable hasColumnHeaders(List<String> value) { }
<THeaders extends Enum> ITable hasColumnHeaders(Class<THeaders> headers) { }
ITable hasRowHeaders(List<String> value) { }
<THeaders extends Enum> ITable hasRowHeaders(Class<THeaders> headers) { }
ITable setColumnsCount(int value) { }
ITable setRowsCount(int value) { }

# COMPOSITE ELEMENTS

# COMPOSITE ELEMENTS

```
public class Header : Section

[JPage (Url = "/index.html", Title = “Good site") ]
public class HomePage : WebPage

[JSite (Domain = “http://epam.com/") ]
public class EpamSite : WebSite

public class LoginForm : Form

public class SearchBar : Search

public class Alert : Popup

public class Navigation : Pagination
```

```
[JPage (domain = “http://epam.com/") ]
public class EpamSite : WebSite {
        [JPage (Url = "/index.html") ]
        public static HomePage homepage;
        [JPage (Url = "/login", Title = “Login page") ]
        public static LoginPage loginPage;
        [FindBy (Css=“.nav”) ]
        public static Menu navigation;
}

public static void SetUp() {
    WebSite.Init(typeof(EpamSite));
}
```

# WEB PAGE

```
[JPage (Url = "/main", Title = "Good site", UrlTemplate = “/main?\d{10}“,
        UrlCheckType = MATCH, TitleCheckType = CONTAINS) ]
public class HomePage : WebPage
```

## USAGE

```
homepage.Open();
homepage.CheckOpened();
homepage.IsOpened();
```

```
homepage.Refresh();
homepage.Back();
homepage.Forward();
homepage.AddCookie();
homepage.ClearCache();
```

# SECTION

```
public class Header : Section {
        [FindBy (Css=".submit") ]
        public Button submit;
        [FindBy (Css=".followMe") ]
        public Link followMe;
        [FindBy (Css=".navigation") ]
        public Menu navigation;

        public void openAbout() {
                followMe.Click();
                navigation.Select(ABOUT);
        }
}
```

## USAGE

```
header.Submit.Click();
header.Menu.IsSelected();
header.OpenAbout();
```

# ENTITY DRIVEN TESTING

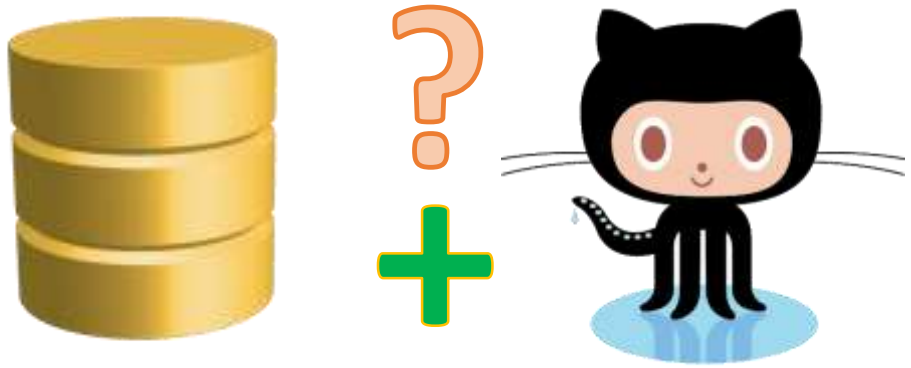
# EDT: DATA DRIVEN TESTING

- Provide List<*User*> for test

# EDT: PRECONDITIONS

- Provide List<*User*> for test

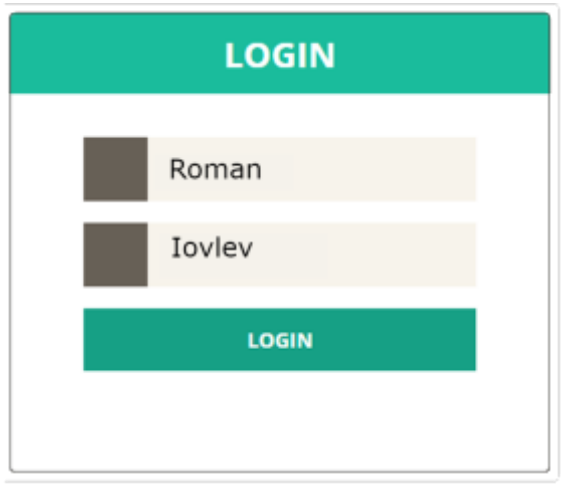
0. Have *DefaultUser* in DB



# EDT: FILL AND SUBMIT

- Provide List<*User*> for test

0. Have *DefaultUser* in DB
1. Login with *DefaultUser*



# EDT: FILL AND SEND

- Provide List<*User*> for test

0. Have *DefaultUser* in DB
1. Login with *DefaultUser*
2. Submit Contact Us Form for *DefaultUser*

**CONTACT DETAILS**

| | |
|---|---|
| Roman | Iovlev |
| roman_iovlev@epam.com | 89212345678 |
| Epam | Academic/Student |
| Russian Federation | |

# EDT: EXTRACT

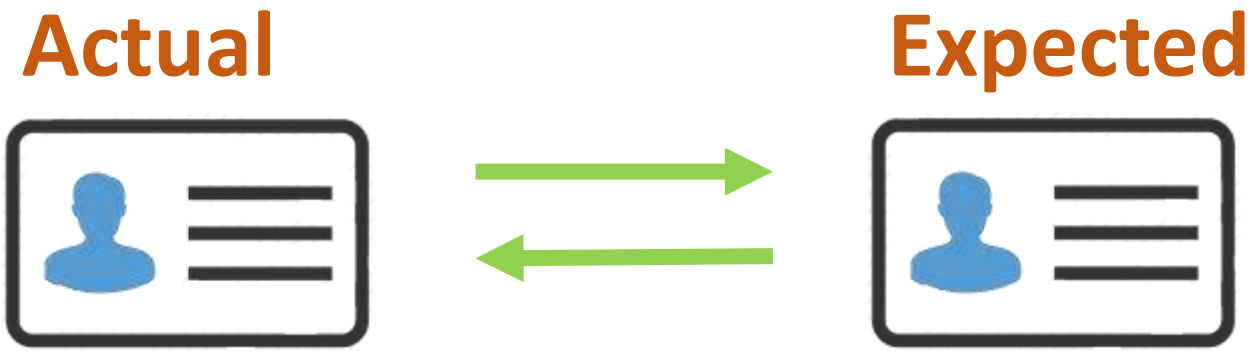
- Provide List<*User*> for test

0. Have *DefaultUser* in DB
1. Login with *DefaultUser*
2. Submit Contact Us Form for *DefaultUser*
3. Get *Act. Opening* from Vacancy table

# EDT: VALIDATE

- Provide List<*User*> for test

0. Have *DefaultUser* in DB
1. Login with *DefaultUser*
2. Submit Contact Us Form for *DefaultUser*
3. Get *Act. Opening* from Vacancy table
4. Assert *Act. Opening* equals to *Exp. Opening*

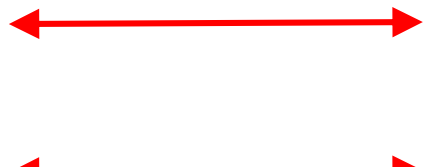


# FORM

```
public class LoginForm extends Form<User> {
        [FindBy (Css=".login")
        public TextField Login;
        [FindBy (Css=".psw")
        public TextField Password;

        [FindBy (Css=".submit")
        public Button Submit;
        [FindBy (Css=".cancel")
        public Button Cancel;

}
```



```
public class User {
        public String Login = "roman";

        public String Password = null;
}

[Test]
public class SimpleTest(User User) {
        loginForm.login(user);
        …
}
```

# ENTITY DRIVEN TESTING

```
[Test]
public void FormTest(User Admin) {
        LoginForm.LoginAs(Admin);
        Filter.Select(Admin.name);
        Assert.Each(Results).Contains(Admin.Name);
        Results.get(1);
        PayForm.Submit(Admin.CreditCard);
        Assert.AreEquals(DB.Transactions[1]),
                Admin.CreditCard);
}
```

## USAGE

```
LoginForm.Fill(User);
LoginForm.Submit(User);
LoginForm.Verify(User);
LoginForm.Check(User);

LoginForm.Cancel(User);
LoginForm.Save(User);
LoginForm.Publish(User);
LoginForm.Search(User);
LoginForm.Update(User);
…
```

```
[FindBy (Css = “.offers") ]
public Table<Job, JobRecord> Offers = new Table {
        Row = By.Css(".value"),
        Column = By.TagName(“li"),
        Header = new [] {“ID", “Title", “Apply”}
};
```

```
[FindBy (Css = ".list.ui-sortable") ]
public Elements<FilmCard> FilmCards;

FilmCards[name].Title.GetText();
```

# OTHER UI OBJECTS

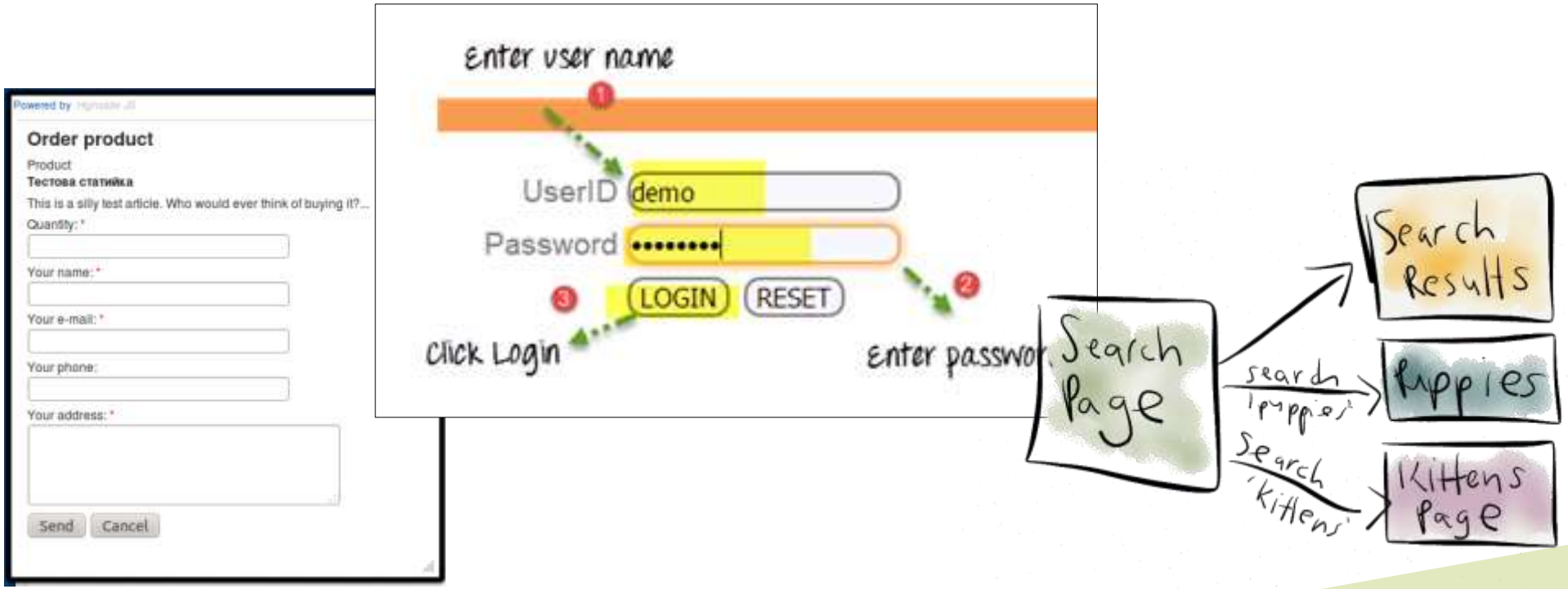
```
public class SearchBar : Search { }

public class Navigation : Pagination { }

public class Confirmation : Popup { }
…
public class MyCustom : CustomObject { }
// implements IComposite
```

# UI OBJECTS PATTERN

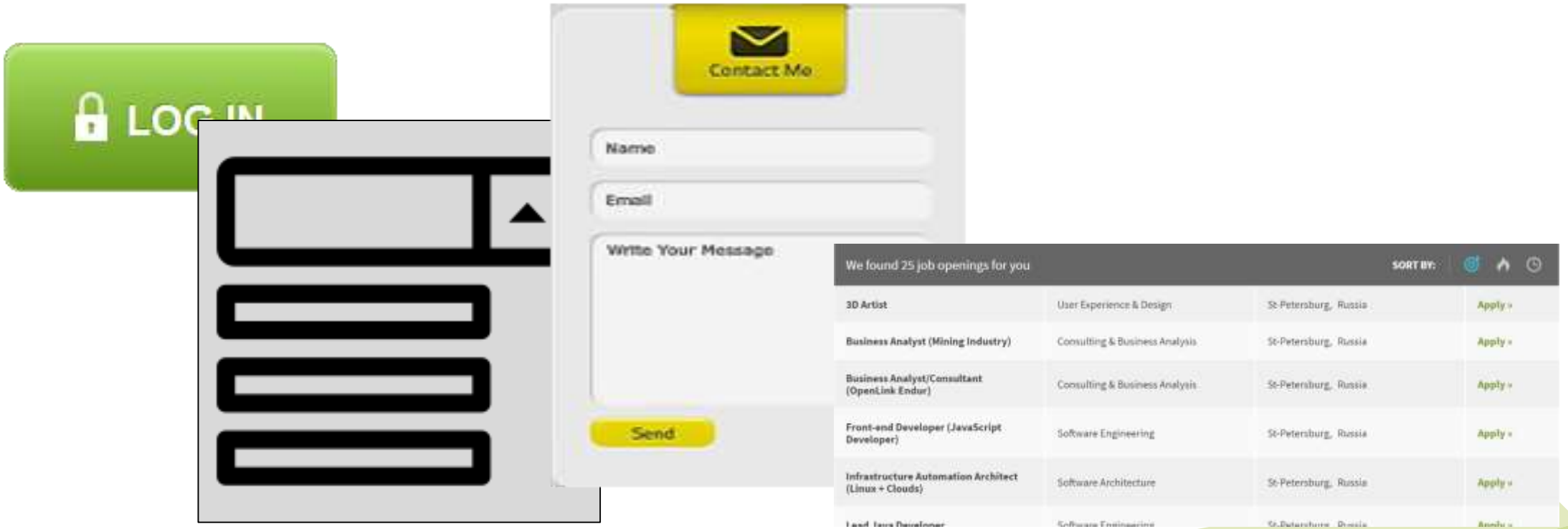
# Popular test pattern
## Page Objects



epam

# Useful test approach
# UI Elements

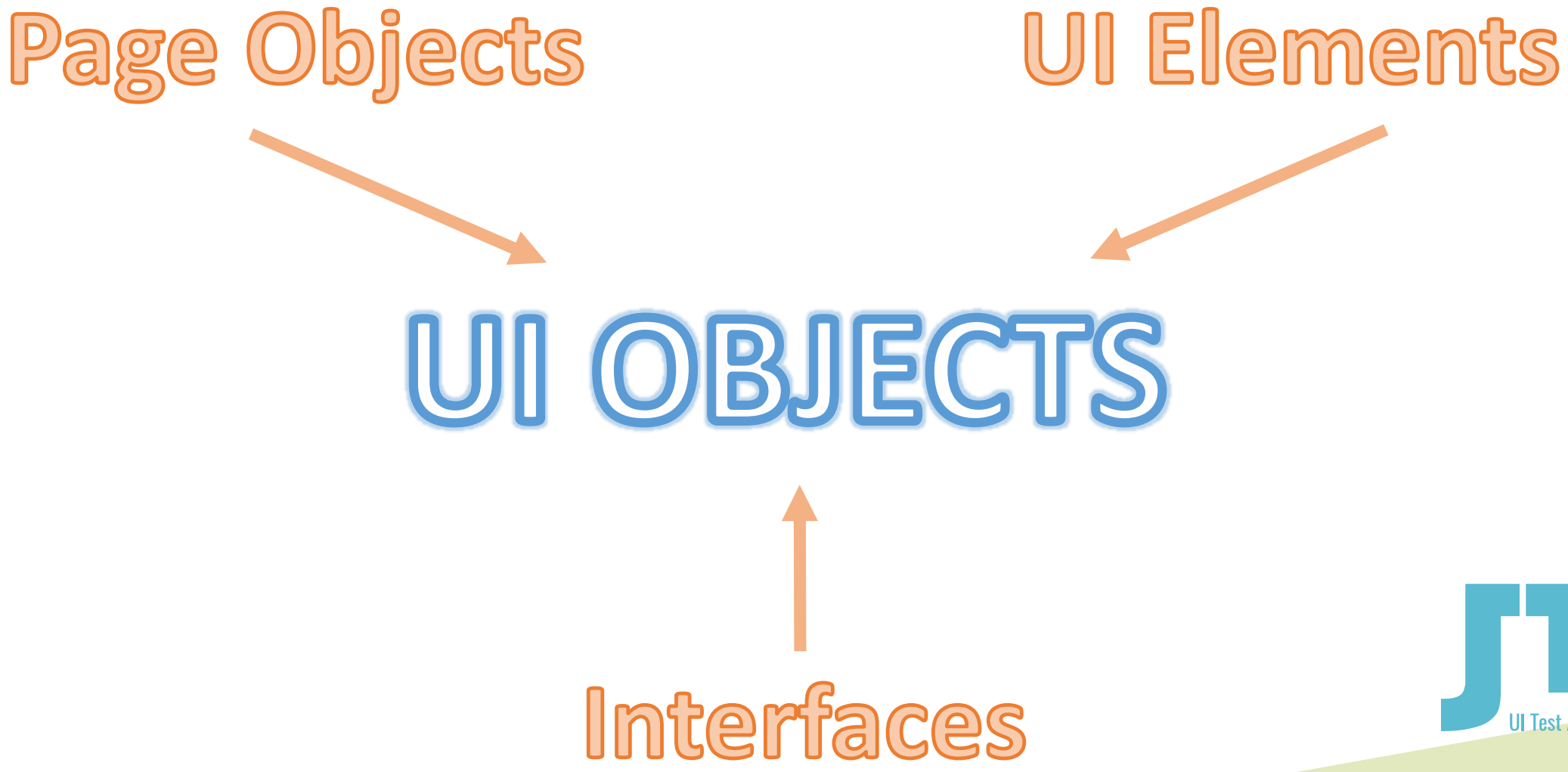
JDI
UI Test Automation Framework



# Flexible implementation

## Interfaces

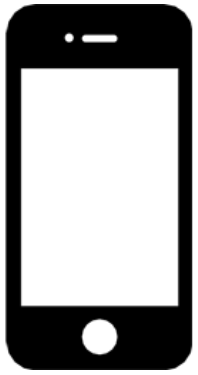
JDI
UI Test Automation Framework




JDI
UI Test Automation Framework

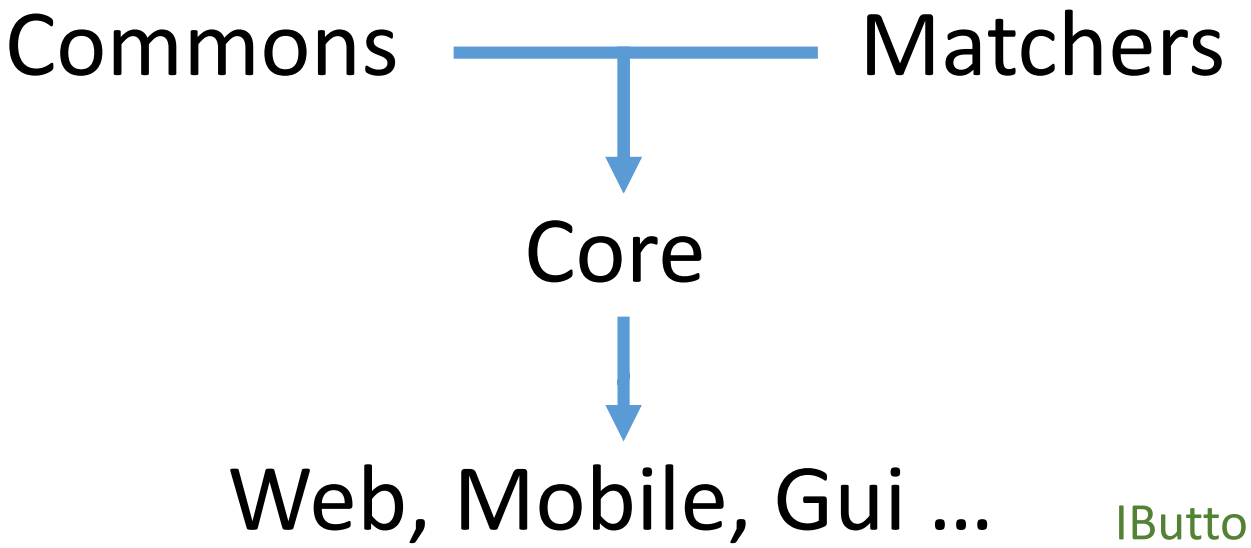
# INTERFACES

# TEST ANY UI



## Your Engine (Driver)

# JDI ARCHITECTURE




IElement
ISelector
IComposite
IPage

IHasValue
ISetValue

## INTERFACES

```
public interface ICheckBox
        extends IClickable, ISetValue {
    void Check();
    void Uncheck();
    boolean IsChecked();
}
```

IButton
ICheckBox
IDatePicker
IFileInput
IImage
ILabel
ILink
IText
ITextArea

ITextField
ICheckList
IComboBox
IDropDown
IDropList
IForm
IGroup
IMenu
IPage

IPagination
IPopup
IRadioButtons
ISearch
ISelector
ITabs
ITextList

# FOR BUSINESS

# JDI BENEFITS

- Write test code **faster up to 5 times**
  - Average result around 2.8 times
  - 4.7 times speedup on the project with standard implementation
- Produce **less** amount of **test code (loc) up to 3 times**
  - Average result around 2.1 times
  - 2.8 times reduction on the project with standard implementation
- Achieve **higher clearness of tests**
  - Decrease of support time for test projects
  - Lowering of project entry barrier for newcomers
- Complete projects with **higher quality**
  - Based on 70 % answers in survey

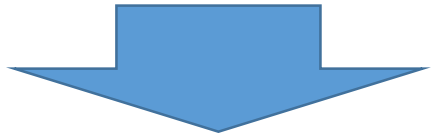
# JDI BENEFITS

- **Reuse investments** from one Project on another
  - Based on 5 years of work and more than 30 projects that already use JDI
- **Save up to 80% test effort** by migrating tests to other Platforms
  - Based estimated average scope reductions for all test process stages
  - Example: migrate Web tests to Mobile platform
- Can be **used in most of projects** with UI Automation
  - Actually we have no projects where JDI is not applicable. The only reason why not all of our projects use JDI is Client requirements
- **Save up to 30-40% money** from whole test process
  - Based on average calculation of scope reductions for all test process stages

# FUTURE

# LOG IN BDD STYLE

```
ProductPage.ProductType.select(«jacket»);
ProductPage.Price.select(«500$»);
ProductPage.Colors.check(«black», «white»);
```



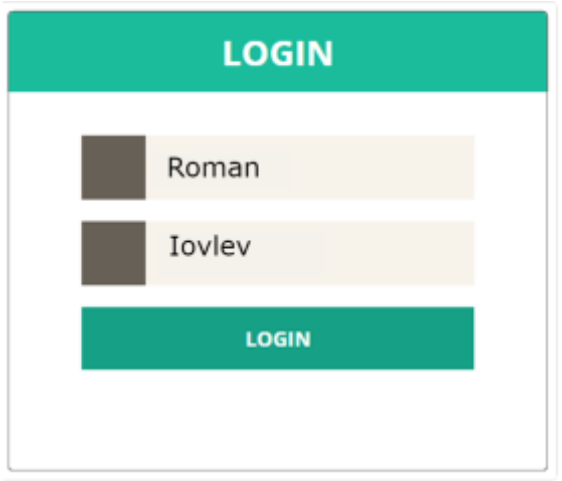
I select ‘jacket’ ProductType on ProductPage

I select ‘500$’ Price on ProductPage

I check ‘black’ and ‘white’ Colors on ProductPage

# GENERATED PAGE OBJECTS

- Chrome plugin
  - Get ui elements to page object in browser
  - Auto generation




```
public class LoginForm extends Form<User> {
        @FindBy (css=".login")
        public TextField login;
        @FindBy (css=".psw")
        public TextField password;

        @FindBy (css=".submit")
        public Button login;
}
```

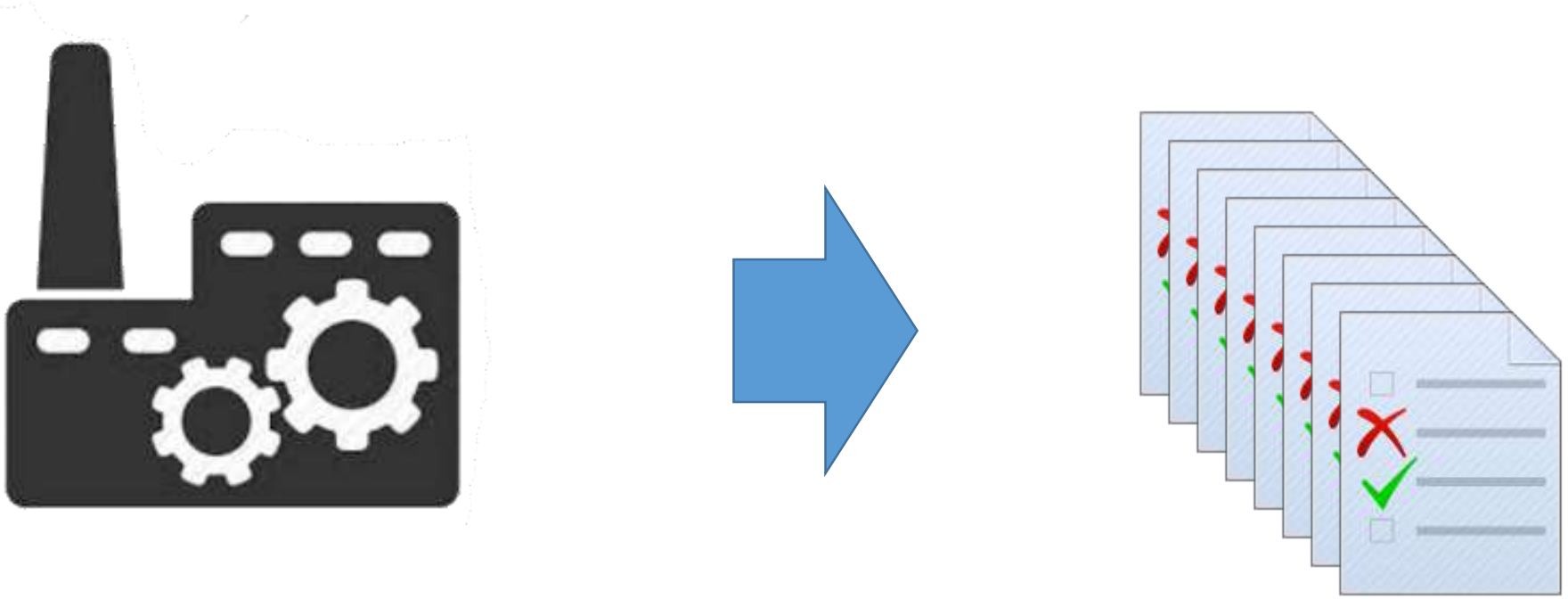
# SERVICES PAGE OBJECTS

```
@ServiceDomain("http://service.com")
public class ServiceExample {
        @GET ("/color/get")
        public RestMethod getColor;
        @POST ("/color/change/100")
        public RestMethod changeColor;
}
```

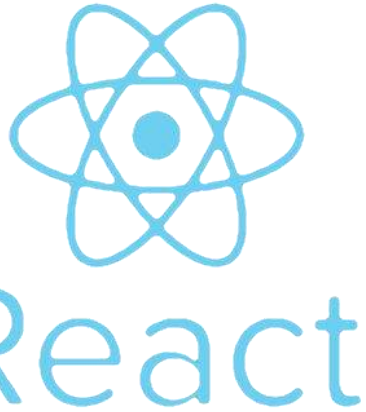
Auto generation by WSDL

# DB TESTING SUPPORT

# SIMPLE TESTS GENERATOR



# SUPPORT MAIN UI DEV FRAMEWORKS



# PERFORMANCE / STATISTIC

# SNIFF HTTP REQUESTS

# SUPPORT JS / PHYTON

enjoy
your
life

# CONTACTS

JDI
UI Test Automation Framework

roman.lovlev

roman_iovlev@epam.com

http://jdi.epam.com/

https://vk.com/jdi_framework

https://github.com/epam/JDI