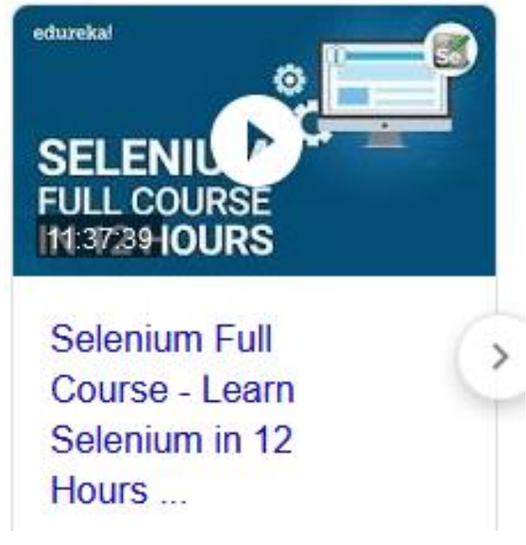
Integration and System Testing

- Java library HtmlUnit

slides: João Neto

HtmlUnit

- Do we need users using browsers for integration tests in web applications?
  - Even with tools like Selenium, Katalon or Cypress?



- How to test web applications programmatically in Java code?
- The library HtmlUnit (`htmlunit.sourceforge.net/`) includes a non-GUI browser
  - It provides an API to access webpages, fill forms, click links…
  - Also provides methods to access elements of HTML pages
  - Simulates Firefox, Chrome, IE, Safari
  - Integrates with JUnit4 for testing
  - However, it allows for arbitrary web scraping

Web Scraping

Areas that form integral parts of sovereign states, such as the countries of the United Kingdom, are counted as part of the sovereign states concerned. Not included are other entities, such as the European Union,[Note 1] that are not sovereign states, and independent territories that do not have permanent populations, such as various countries' claims to Antarctica.

Sovereign states and dependencies by population [edit]

Note: All dependent territories or constituent countries that are parts of sovereign states are shown in *italics*.

Rank	Country (or dependent territory)	Population	Date	% of world population	Source
1	China[Note 2]	1,388,950,000	January 30, 2018	18.3%	Official population clock
2	India[Note 3]	1,327,250,000	January 30, 2018	17.5%	Official population clock
3	United States[Note 4]	326,542,000	January 30, 2018	4.3%	Official population clock
4	Indonesia	261,890,900	July 1, 2017	3.45%	Official annual projection
5	Pakistan	210,421,000	January 30, 2018	2.77%	Official population clock
6	Brazil	208,594,000	January 30, 2018	2.75%	Official population clock
7	Nigeria	193,392,500	March 21, 2016	2.55%	Annual official estimate
8	Bangladesh	163,911,000	January 30, 2018	2.16%	Official population clock
9	Russia[Note 5]	146,877,088	January 1, 2018	1.93%	Official estimate
10	Japan	126,590,000	January 1, 2018	1.67%	Monthly provisional estimate
11	Mexico	123,675,351	October 1, 2017	1.63%	Official projection
12	Philippines	105,364,000	January 30, 2018	1.39%	Official population clock
13	Egypt	96,434,200	January 30, 2018	1.27%	Official population clock
14	Ethiopia	94,352,000	July 1, 2017	1.24%	Official projection
15	Vietnam	93,700,000	July 1, 2017	1.23%	Annual official projection
16	Germany	82,521,653	December 31, 2016	1.09%	Official annual data

en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population

- We wish to get the information of all the countries names and population

```
...
<p>Note: All dependent territories or constituent countries that are parts of sovereign
states are shown in <i>italics</i>.</p>
<table class="wikitable sortable" style="text-align:right">
<tr>
<th data-sort-type="number">Rank</th>
<th>Country<br />(or dependent territory)</th>
<th>Population</th>
<th>Date</th>
<th>% of world<br />
population</th>
<th class="unsortable">Source</th>
</tr>
<tr>
<td>1</td>
<td align="left"><span class="flagicon" style="display:inline-block;width:25px;"><img
alt="" src=… <a href="/wiki/China" title="China">China</a><td>
<td>1,390,570,000</td>
<td>April 26, 2018</td>
<td>18.3%</td>
<td align="left"><a rel="nofollow" class="external text"
href="http://data.stats.gov.cn/english/">Official population clock</a></td>
</tr>
...
```

Web Scraping

- We wish to get the information of all the countries' names and population
- Using HtmlUnit:

```
public class CountriesWikipedia {

  private static final String url =
     "https://en.wikipedia.org/wiki/List_of_countries...";

  public static void main(String[] args) throws Exception {

    HtmlPage page;

    try (final WebClient webClient =
              new WebClient(BrowserVersion.getDefault())) {
      webClient.getOptions().setCssEnabled(false);
      webClient.getOptions().setJavaScriptEnabled(false);

      page = webClient.getPage(url);
  }
```

Web Scraping

```
// get table using a unique feature on the webpage via XPath
final HtmlTable countriesTable = (HtmlTable)
    page.getByXPath("//table[@class='wikitable sortable']")
        .toArray()[0];

for(final HtmlTableRow row : countriesTable.getRows()) {
  // this table has six columns, we need the 2nd and 3rd
  List<HtmlTableCell> infoCountry = row.getCells();
  if (infoCountry.get(2).asText().contains("Population"))
     continue; // skip header
  System.out.println(infoCountry.get(1).asText() + "  " +
                     infoCountry.get(2).asText());
  }
} // main()
```

```
China[Note 2]  1,388,950,000
India[Note 3]  1,327,250,000
United States[Note 4]  326,542,000
Indonesia  261,890,900
Pakistan  210,421,000
Brazil  208,594,000
Nigeria  193,392,500
…
```

Word of Warning

- Websites can detect you are web scraping and blacklist you
  - Unusual download rate, repeated operations or honeypots can be markers of automatic web scraping
  - Honeypots herein are hidden links for humans, but crawlers will click them
- Check robots.txt (at root directory) for this pattern:
  - Eg: https://facebook.com/robots.txt

```
User-agent: *
Disallow: /
```

- Since Google is the über-crawler, and almost everybody likes publicity, most sites allow access to crawlers
  - Eg: https://www.publico.pt/robots.txt

```
User-agent: *
Allow: *
```

- Cf. `www.scrapehero.com/how-to-prevent-getting-blacklisted-while-scraping/`

Testing with HtmlUnit

- How to use it to test web applications?
  - We'll use `vvs_webapp` maven project
- For a first test, let's just go to the main page and check if it's working

```
private static final String APPLICATION_URL =
                                "http://localhost:8080/VVS_webappdemo/";

private static HtmlPage page;

@BeforeClass
public static void setUpClass() throws Exception {
  try (final WebClient webClient =
                    new WebClient(BrowserVersion.getDefault())) {

    page = webClient.getPage(APPLICATION_URL);
    // OK status?
    assertEquals(200, page.getWebResponse().getStatusCode());
  }
}
```

Get information from HtmlPage

- `HtmlPage` represents the page returned by the server
- We can convert it to a string on text or xml format
- This class has a rich API that allows for complex queries
  - Includes DOM manipulation and XPath search

```
@Test
public void indexTest() throws Exception {
  assertEquals("WebAppDemo Menu", page.getTitleText());

  final String pageAsXml = page.asXml();
  assertTrue(pageAsXml.contains("<div class=\"w3-container w3-
blue-grey w3-center w3-allerta\" id=\"body\">"));

  final String pageAsText = page.asText();
  assertTrue(pageAsText.contains("WebAppDemo Menu"));
}
```

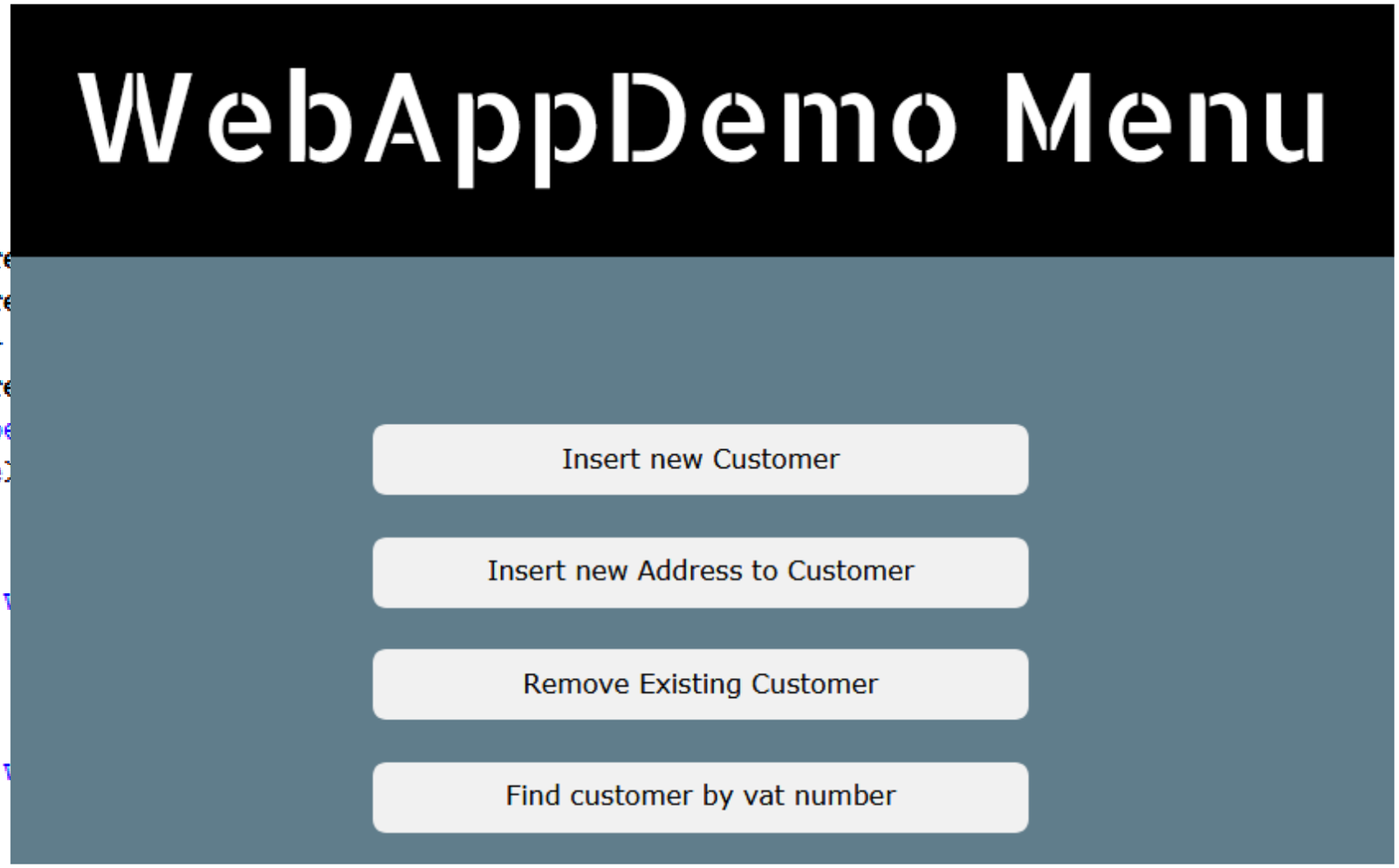
Using DOM

- HTML DOM is a standard for how to get, change, add, or delete HTML elements
- In the following test, the library returns all elements from the webpage with id `botao2`
- At `index.html`, use cases are activated via HTML elements with that id

```
private static final int APPLICATION_NUMBER_USE_CASES = 11;

@Test
public void numberOfOptionsTest() throws Exception {
    // get list of case uses
    List<DomElement> inputs = page.getElementsById("botao2");

    assertTrue(inputs.size()==APPLICATION_NUMBER_USE_CASES);
}
```

Using DOM

```
<html>
<head>
    <link rel="stylesheet" hre
    <link rel="stylesheet" hre
/css?family=Allerta+Stencil">
    <link rel="stylesheet" hre
<meta http-equiv="Content-Type
<title>WebAppDemo Menu</title>
</head>
<body>
    <div class="w3-container w
    <p>WebAppDemo Menu</p>
    </div>

    <div class="w3-container w
        <form>
        <br>
        <br>
        <br>
        <br>
        <a id="botao2" class="w3-button w3-light-grey w3-round-large w3-allerta"
href="addCustomer.html">Insert new Customer</a>
        <br>
        <br>
        <a id="botao2" class="w3-button w3-light-grey w3-round-large w3-allerta"
href="addAddressToCustomer.html">Insert new Address to Customer</a>
```



Integration Tests

- With this functionality it's possible to test two or more components
  - Scripted automation allows for flexibility
- A test can tell a narrative and check if the application is behaving well
  - Testing based on the model's expected behavior
  - Important to determine what to test
- Let's see a test where we insert and remove a user
  - We also include the removal to keep the database unchanged
  - An alternative would be to create a database only for testing purposes

```
@Test
public void insertAndRemoveClientTest() throws IOException {
        final String NPC = "503183504";
        final String DESIGNATION = "FCUL";
        final String PHONE = "217500000";
```

Integration Tests

```
// get a specific link
HtmlAnchor addCustomerLink = page.getAnchorByHref("addCustomer.html");
// click on it
HtmlPage nextPage = (HtmlPage) addCustomerLink.openLinkInNewWindow();
// check if title is the one expected
assertEquals("Enter Name", nextPage.getTitleText());

// get the page first form:
HtmlForm addCustomerForm = nextPage.getForms().get(0);

// place data at form
HtmlInput vatInput = addCustomerForm.getInputByName("vat");
vatInput.setValueAttribute(NPC);
HtmlInput designationInput = addCustomerForm.getInputByName("designation");
designationInput.setValueAttribute(DESIGNATION);
HtmlInput phoneInput = addCustomerForm.getInputByName("phone");
phoneInput.setValueAttribute(PHONE);
// submit form
HtmlInput submit = addCustomerForm.getInputByName("submit");
```

Integration Tests

```
// check if report page includes the proper values
HtmlPage reportPage = submit.click();
String textReportPage = reportPage.asText();
assertTrue(textReportPage.contains(NPC));
assertTrue(textReportPage.contains(DESIGNATION));
assertTrue(textReportPage.contains(PHONE));

// at index, goto Remove case use and remove the previous client
HtmlAnchor removeCustomerLink =
          page.getAnchorByHref("RemoveCustomerPageController");
nextPage = (HtmlPage) removeCustomerLink.openLinkInNewWindow();
assertTrue(nextPage.asText().contains(NPC));

HtmlForm removeCustomerForm = nextPage.getForms().get(0);
vatInput = removeCustomerForm.getInputByName("vat");
vatInput.setValueAttribute(NPC);
submit = removeCustomerForm.getInputByName("submit");
submit.click();
```

Integration Tests

```
  // now check that the new client was erased
  HtmlAnchor getCustomersLink =
     page.getAnchorByHref("GetAllCustomersPageController");
  nextPage = (HtmlPage) getCustomersLink.openLinkInNewWindow();
  assertFalse(nextPage.asText().contains(NPC));
}
```

- The test uses three use cases to validate the application behavior during the narrative "*customer is added, then removed, then checked that is no longer among all customers*"

Testing a GET request

- The previous narrative included two POST requests (insert, remove) and a GET request (getAllCustomers)
- HtmlUnit allows to send GET request directly by code, without loading html pages

```
@Test
public void parametersGetTest() throws IOException {

  HtmlPage reportPage;

  // Build a GET request
  try (final WebClient webClient =
                  new WebClient(BrowserVersion.getDefault())) {
    java.net.URL url = new java.net.URL(APPLICATION_URL +
                                        "GetCustomerPageController");
    WebRequest requestSettings =
                          new WebRequest(url, HttpMethod.GET);
```

Testing a GET request

- The previous narrative included two POST requests (insert, remove) and a GET request (getAllCustomers)
- HtmlUnit allows to send GET request directly by code, without loading html pages

```
    // Set the request parameters
    requestSettings.setRequestParameters(
                                new ArrayList<NameValuePair>());
    requestSettings.getRequestParameters()
        .add(new NameValuePair("vat", "197672337"));
    requestSettings.getRequestParameters()
        .add(new NameValuePair("submit", "Get+Customer"));

    reportPage = webClient.getPage(requestSettings);
  } // try

  assertTrue(reportPage.asXml().contains("JOSE FARIA"));
}
```

Exercises

- Test the following webapp demo features:
  - The ‘update customer contacts’ use case (check by listing all customers)
  - Insert two sales for a given customer, close one and keep another open. Then show customer’s sales to verify if the information is correct

Using XPath

- Used to navigate through elements and attributes in an XML document.
- Uses path expressions to select nodes or node-sets in an XML document.
  - XML documents are treated as trees of nodes. The topmost element of the tree is called the root element.
  - Nodes can have children, parents, siblings (nodes that share the same parent)

Using XPath

- Path expressions ease node selection, egs:
  - `user`, select all nodes with name ‘user’
  - `/user`, select the root node called ‘user’
  - `user/phone`, select all phone nodes children of user nodes
  - `//user`, select all nodes with name ‘user’ from the current node
  - `//@lang`, select all attributes named ‘lang’
  - `user/phone[1]`, select the first phone of user nodes
  - `user/phone[last()-1]`, select the penultimate phone
  - `user/phone[position()<4]`, select the first three phones
  - `user[@lang='en']`, select the users with attribute lang = ‘en’
  - `user/credit[value>=50 and value=<75]/deadline`, select the deadlines for users with credits with element value in [50,70]
  - `//user/debit | //user/credit`, select all credit and debit nodes from current user

Using XPath

- When writing HTML be nice to yourself and write in a way to allow simple XPath expressions
  - This means coherent HTML code
  - Add information (id's, classes) to ease tests

- Exercise: web-scrape the highest mountains on Earth (select name, height and url) at wikipedia