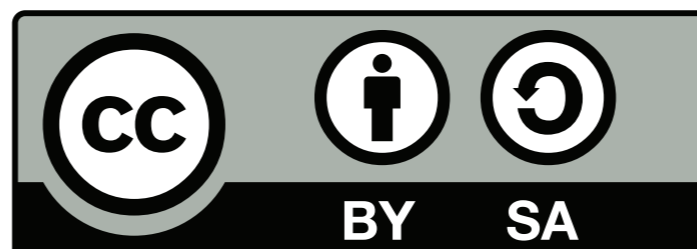


# Tecnologia e Applicazioni Internet 2011/12

Lezione 10 - Acceptance tests

Matteo Vaccari

<http://matteo.vaccari.name/>  
[matteo.vaccari@uninsubria.it](mailto:matteo.vaccari@uninsubria.it)



@Test

```
public void authorizeTranpipeRequest() throws Exception {
```

```
    String request = "" +  
        "<id>" + TerminalFactory.getTranpipeExplicitCaptureType().getId() + "</id>" +  
        "<password>" + TerminalFactory.getTranpipeExplicitCaptureType().getPassword() + "</password>" +  
        "<card>" + CCFactory.getVisaOnUsNotEnrolled().getPan() + "</card>" +  
        "<cvv2>" + CCFactory.getVisaOnUsNotEnrolled().getCVV2() + "</cvv2>" +  
        "<expYear>" + CCFactory.getVisaOnUsNotEnrolled().getExpirationYear() + "</expYear>" +  
        "<expMonth>" + CCFactory.getVisaOnUsNotEnrolled().getExpirationMonth() + "</expMonth>" +  
        "<action>4</action>" +  
        "<type>CC</type>" +  
        "<member>member&member</member>" +  
        "<amt>0.1</amt>" +  
        "<currencycode>978</currencycode>" +  
        "<trackid>2011IVR4189718</trackid>";
```

```
    String expectedResponse = "" +  
        "<response type=\"valid\">" +  
        "<result>APPROVED</result>" +  
        "<auth>DONTCARE</auth>" +  
        "<ref>DONTCARE</ref>" +  
        "<tranid>DONTCARE</tranid>" +  
        "<trackid>2011IVR4189718</trackid>" +  
        "<udf5>EQUAL2REF</udf5>" +  
        "<responsecode>000</responsecode>" +  
        "</response>";
```

```
    String actualResponse = sendTranpipeRequest(request);  
    assertXmlResponseEquals(expectedResponse, actualResponse);
```

```
}
```

# Expressive tests

```
TempConvPage page = TempConvPage.getTempConvPage();
assertEquals("", page.getDisplay());
page.setTemperature("0");
TempConvPage secondPage = page.submit();
assertEquals("0.0 C° = 32.0 F°", secondPage.getDisplay());
```

```
WebClient client = new WebClient();
HtmlPage page = client.getPage(BASE_URL);
assertEquals("", page.getElementById("display").getTextContent());
HtmlTextInput celsius = page.getElementByName("celsius");
celsius.type("0");
HtmlPage newPage = page.getElementById("submit").click();
assertEquals("0.0 C° = 32.0 F°",
    newPage.getElementById("display").getTextContent());
```

# Usare HtmlUnit

```
@Test
public void homePage() throws Exception {
    final WebClient webClient = new WebClient();
    final HtmlPage page = webClient.getPage("http://htmlunit.sourceforge.net");
    assertEquals("HtmlUnit - Welcome to HtmlUnit", page.getTitleText());

    final String pageAsXml = page.asXml();
    assertTrue(pageAsXml.contains("<body class=\"composite\">"));

    final String pageAsText = page.asText();
    assertTrue(pageAsText.contains("Support for the HTTP and HTTPS protocols"));
}
```

<http://htmlunit.sourceforge.net/>

# Looking for a specific element

```
@Test
public void getElements() throws Exception {
    final WebClient webClient = new WebClient();
    final HtmlPage page = webClient.getPage("http://some_url");
    final HtmlDivision div = page.getHtmlElementById("some_div_id");
    final HtmlAnchor anchor = page.getAnchorByName("anchor_name");
}
```

# Using XPath

```
@Test
public void xpath() throws Exception {
    final WebClient webClient = new WebClient();
    final HtmlPage page = webClient.getPage("http://htmlunit.sourceforge.net");

    //get list of all divs
    final List<?> divs = page.getByXPath("//div");

    //get div which has a 'name' attribute of 'John'
    final HtmlDivision div = (HtmlDivision) page.getByXPath("//div[@name='John']").get(0);
}
```

# Submitting a form

```
@Test
public void submittingForm() throws Exception {
    final WebClient webClient = new WebClient();

    // Get the first page
    final HtmlPage page1 = webClient.getPage("http://some_url");

    // Get the form that we are dealing with and within that form,
    // find the submit button and the field that we want to change.
    final HtmlForm form = page1.getFormByName("myform");

    final HtmlSubmitInput button = form.getInputByName("submitbutton");
    final HtmlTextInput textField = form.getInputByName("userid");

    // Change the value of the text field
    textField.setValueAttribute("root");

    // Now submit the form by clicking the button and get back the second page.
    final HtmlPage page2 = button.click();
}
```

# Must set up a server

```
@BeforeClass
```

```
public static void buildAndStartServer() throws Exception {  
    buildWar();  
    startServer();  
}
```

```
private static void buildWar() throws IOException, InterruptedException {  
    Process process = Runtime.getRuntime().exec("ant clean war");  
    process.waitFor();  
    assertEquals(0, process.exitValue());  
}
```

```
private static void startServer() throws IOException {  
    Map<String, String> args = new HashMap();  
    args.put("debug", "" + Logger.ERROR);  
    Launcher.initLogger(args);  
    args.put("httpPort", "8123");  
    args.put("ajp13Port", "-1");  
    args.put("warfile", WAR_PATHNAME);  
    new Launcher(args);  
}
```



**End-to-end tests must  
be expressive**

```

public void testContent() throws Exception {
    URL loginPageUrl = new URL("http://localhost/coffeeShop/login.html");
    File loginPageFile = new File(webContentDirectory, "login.html");
    WebClient webClient = new WebClient();
    FileSystemWebResponse webResponse = new FileSystemWebResponse(loginPageUrl, loginPageFile);
    webResponse.setContentType("text/html");
    FileSystemWebConnection fileSystemWebConnection = new FileSystemWebConnection(webClient);
    fileSystemWebConnection.setResponse(webResponse);
    webClient.setWebConnection(fileSystemWebConnection);

    HtmlPage loginPage = (HtmlPage) webClient.getPage(loginPageUrl);
    assertEquals("Login", loginPage.getTitleText());
    assertTrue(loginPage.asText().indexOf("Enter your user name and password") >= 0);
    HtmlForm loginForm = loginPage.getFormByName("loginForm");
    assertNotNull(loginForm);
    assertEquals("/coffeeShop", loginForm.getActionAttribute());
    assertTrue("post".equalsIgnoreCase(loginForm.getMethodAttribute()));
    HtmlInput usernameInput = loginForm.getInputByName("username");
    assertNotNull(usernameInput);
    assertEquals(12, Integer.parseInt(usernameInput.getSizeAttribute()));
    assertTrue(usernameInput instanceof HtmlTextInput);
    assertEquals("", usernameInput.getValueAttribute());
    HtmlInput passwordInput = loginForm.getInputByName("password");
    assertNotNull(passwordInput);
    assertTrue(passwordInput instanceof HtmlPasswordInput);
    assertEquals(12, Integer.parseInt(passwordInput.getSizeAttribute()));
    assertEquals("", passwordInput.getValueAttribute());
    HtmlInput loginInput = loginForm.getInputByName("login");
    assertNotNull(loginInput);
    assertTrue(loginInput instanceof HtmlSubmitInput);
    assertEquals("Login", loginInput.getValueAttribute());
}

```

# End-to-end tests must be expressive

```
@Test  
public void failedLogin() throws Exception {  
    loginAs("invalid user", "invalid password");  
    assertNotLoggedIn();  
}
```

```
@Test  
public void validLogin() throws Exception {  
    loginAs("admin", "secret");  
    assertLoggedIn();  
}
```

# End-to-end tests must be expressive

```
@Test
public void insertACourse() throws Exception {
    loginAs("admin", "secret");

    visit("/app/courses/list");
    int numberBefore = numberOfCoursesListed();
    clickOnNewCourseButton();
    insertCourseTitle("Course Title");
    insertCourseDescription("A Description");
    submitCourseForm();
    assertEquals(numberBefore+1, numberOfCoursesListed());
}
```