Browser testing via Selenium

What is Selenium?

- Browser framework - automating user-level testing
- Has a method for saving scripts - but it's not very good
- Good for testing behaviour that can't be tested any other way - e.g. client-side sorting, that widgets are visible

Selenium directly

```
class DocManagementPageTest extends DriverSpec {
  def goToDocManagementPage(): DocManagementPage = loginAndNavigateTo(PageUrls.DocManagement)

  "The doc management page" should {
    "show nothing for a search for very old documents" in {
     goToDocManagementPage().search(SearchQuery(toDate = Some(new LocalDate(1850, 1,
   1)))).getDocuments.size mustEqual 0
    }
   "show all documents for a non-restrictive date search" in {
    val docManagementPage: DocManagementPage = goToDocManagementPage()
    val initialDocCount = docManagementPage.getDocuments.size
    docManagementPage.search(SearchQuery(toDate = Some(new LocalDate(2050, 1,
 1)))).getDocuments.size mustEqual initialDocCount
 }
 "find an existing document by date" in {
    val docManagementPage: DocManagementPage = goToDocManagementPage()
    val initialDoc = docManagementPage.getDocuments.head.document
    val initialTitle = initialDoc.title
    val foundDocs = docManagementPage.search(SearchQuery(fromDate=Some(initialDoc.releaseDate),
 toDate=Some(initialDoc.releaseDate))).getDocuments
    foundDocs.map(_.document.title) must contain(initialTitle)
 }
```

Selenium directly in Play

"displaying the example page should work" in new WithBrowser(webDriver = createWebDriver()) {
  browser.goTo(exampleUrl)
  browser.pageSource() must contain("Overview")
}

Directly accessing page

```
import browser._
implicit val b: TestBrowser = browser
goTo(exampleUrl)
$(".comment").size() mustEqual 0

// Create a comment
val commentText = randomGuid()
val div = findFirst("#editPolicyOwn")
addPlainTextComment(div, commentText)
div.find(".comment").size() mustEqual 1
div.find(".comment .commentText").first().getText mustEqual commentText

// Edit that comment
val newText : String = randomGuid()
editComment(div, expectedText = commentText, newText)
$(".comment").size() mustEqual 1
$(".comment .commentText").first().getText mustEqual newText

// Now add another normal (non-edit) comment
private val ordinaryCommentText: String = randomGuid()
addPlainTextComment(div, ordinaryCommentText)

val comments = div.find(".comment")
comments.size() mustEqual 2
comments.get(0).find(".commentText").first().getText mustEqual ordinaryCommentText
comments.get(1).find(".commentText").first().getText mustEqual newText
```

Create an API - preferred

```
browser.goTo(exampleUrl)
val commentsSection = new CommentsSection("#editPolicyOwn")(browser)
import commentsSection._
listComments must haveSize(0)

// Create a comment
val originalCommentText = randomGuid()
submitPlainTextComment(originalCommentText)

// Edit that comment
val editedCommentText = randomGuid()
listComments.head.editText(editedCommentText)
listComments must haveSize(1)
listComments.head.getText mustEqual editedCommentText

// Add a new comment, to check that adding new comments doesn't again edit the
existing comment
val newCommentText: String = randomGuid()
submitPlainTextComment(newCommentText)
listComments must haveSize(2)
listComments.map(_.getText) mustEqual List(editedCommentText, newCommentText
```

Why do they break?

- Timing problems - mitigate by waiting for specific elements, “waitForJqueryProcessing”

- Real genuine problems in the underlying code - good to check directly by going to the page in the browser

- UI changes - mitigate by using sensible class names and identifiers in tests

- Performance problems, leading to developers not running the tests, and makes it harder to debug them - mitigate by using a remote web driver, perhaps using Phantom, using a single browser for all tests (and perhaps something like Selenium grid)

- Mysterious things that go away when you try again (often meaning timing, really)

Using in IDEA

- Set up activator to build web:assets and test-web:assets

- Set up activator to compile

Selenium recommendations

- Useful but yeah… use sparingly

- Consider whether the tests can instead use server-side tests, or Jasmine