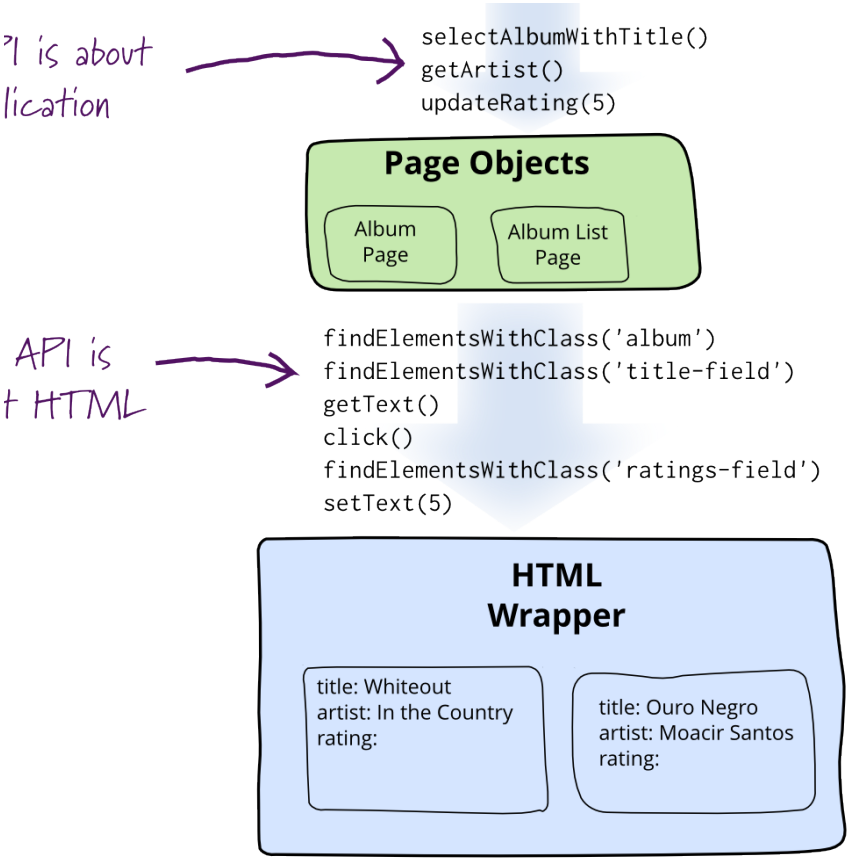
: Object

mber 2013

'owler

TESTING
ENCAPSULATION
WEB DEVELOPMENT

u write tests against a web page, you need to refer to elements within that
: in order to click links and determine what's displayed. However, if you write
t manipulate the HTML elements directly your tests will be brittle to changes
. A page object wraps an HTML page, or fragment, with an application-specific
ving you to manipulate page elements without digging around in the HTML.



: rule of thumb for a page object is that it should allow a software client to do
and see anything that a human can. It should also provide an interface that's
rogram to and hides the underlying widgetry in the window. So to access a

you should have accessor methods that take and return a string, check boxes
se booleans, and buttons should be represented by action oriented method
he page object should encapsulate the mechanics required to find and
ıte the data in the gui control itself. A good rule of thumb is to imagine
the concrete control - in which case the page object interface shouldn't

he term “page” object, these objects shouldn't usually be built for each page,
r for the significant elements on a page [1]. So a page showing multiple
ould have an album list page object containing several album page objects.
uld probably also be a header page object and a footer page object. That said,
he hierarchy of a complex UI is only there in order to structure the UI - such
te structures shouldn't be revealed by the page objects. The rule of thumb is to
e structure in the page that makes sense to the user of the application.

if you navigate to another page, the initial page object should return another
ct for the new page [2]. In general page object operations should return
ntal types (strings, dates) or other page objects.

e differences of opinion on whether page objects should include assertions
es, or just provide data for test scripts to do the assertions. Advocates of
assertions in page objects say that this helps avoid duplication of assertions
ripts, makes it easier to provide better error messages, and supports a more
Ask style API. Advocates of assertion-free page objects say that including
ıs mixes the responsibilities of providing access to page data with assertion
l leads to a bloated page object.

ıving no assertions in page objects. I think you can avoid duplication by
assertion libraries for common assertions - which can also make it easier to
good diagnostics. [3]

cts are commonly used for testing, but should not make assertions
es. Their responsibility is to provide access to the state of the underlying
up to test clients to carry out the assertion logic.

ribed this pattern in terms of HTML, but the same pattern applies equally well
technology. I've seen this pattern used effectively to hide the details of a Java
and I've no doubt it's been widely used with just about every other UI
rk out there too.

ncy issues are another topic that a page object can encapsulate. This may
iding the asynchrony in async operations that don't appear to the user as
may also involve encapsulating threading issues in UI frameworks where you
orry about allocating behavior between UI and worker threads.

cts are most commonly used in testing, but can also be used to provide a
interface on top of an application. Usually it's best to put a scripting interface
ath the UI, that's usually less complicated and faster. However with an
on that's put too much behavior into the UI then using page objects may make

of a bad job. (But look to move that logic if you can, it will be better both for and the long term health of the UI.)

non to write tests using some form of DomainSpecificLanguage, such as er or an internal DSL. If you do this it's best to layer the testing DSL over the ects so that you have a parser that translates DSL statements into calls on the ect.

that aim to move logic out of UI elements (such as Presentation upervising Controller, and Passive View) make it less useful to test the UI and thus reduce the need for page objects.

If you have WebDriver APIs in you methods, You're Doing It Wrong. - Simon Stewart.

ects are a classic example of encapsulation - they hide the details of ructure and widgetry from other components (the tests). It's a good rinciple to look for situations like this as you develop - ask yourself “how can I ıe details from the rest of the software?” As with any encapsulation this yields efits. I've already stressed that by confining logic that manipulates the UI to a ace you can modify it there without affecting other components in the system. quential benefit is that it makes the client (test) code easier to understand the logic there is about the intention of the test and not cluttered by UI

r Reading

ribed this pattern under the name Window Driver. However since then the term “page object” was d by the Selenium web testing framework and that's become the generally used name.

wiki strongly encourages using page objects and provides advice on how they should be used. It assertion-free page objects.

asured the times to update two versions of a suite of selenium tests after a software upgrade. They version with page object took a bit longer for the first test case, but much faster for the rest. For ls see Leotta et al, “Improving test suites maintainability with the page object pattern”, ICSTW

vledgements

ler, Pete Hodgson, and Simon Stewart gave particularly useful comments on drafts of this post - s usual I owe much to various denizens of Thoughtworks's internal software development list for stions and corrections.

n argument here that the name “page object” is misleading because it makes you think you should ne page object per page. Something like “panel object” would be better - but the term “page hat's become accepted. Another illustration of why naming is one of the TwoHardThings.

age objects be responsible for creating other page objects in response to things like navigation is dvice. However some practitioners prefer that page objects return some generic browser context, ts control which page objects to build on top of that context based on the flow of the test y conditional flows). Their preference is based on the fact that the test script knows what pages ed next and this knowledge doesn't need to be duplicated in the page objects themselves. They eir preference when using statically typed languages which usually reveal page navigations in type

n of assertions is fine even for people like me who generally favor a no-assertion style. These
are those that check the invariants of a page or the application at this point, rather than specific
a test is probing.