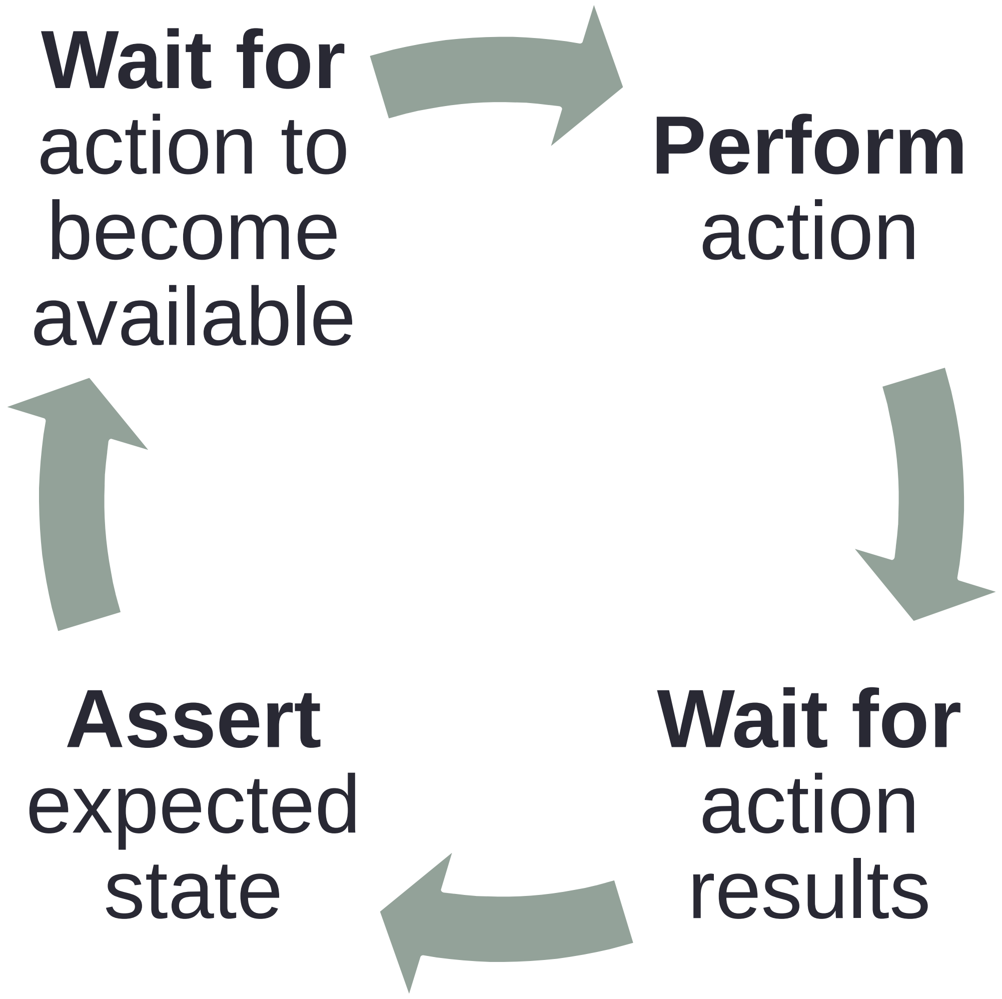
AJAX and Performance

Modern applications are not easy to test

AJAX typical workflow


Wait for action to become available
Perform action
Wait for action results
Assert expected state

You always need to wait for…

- Page loading
- Element appearance
- Element visibility
- Element disappearance
- Alert or popup dialog
- New window
- Text changes
- Element style changes

WebDriver waits for page loading

- By default page is loaded synchronously
- No need to use *waitForPageToLoad*

BUT

- Waiting is broken for long polling and some other techniques
- http://code.google.com/p/selenium/issues/detail?id=687#c4

You can avoid waiting manually

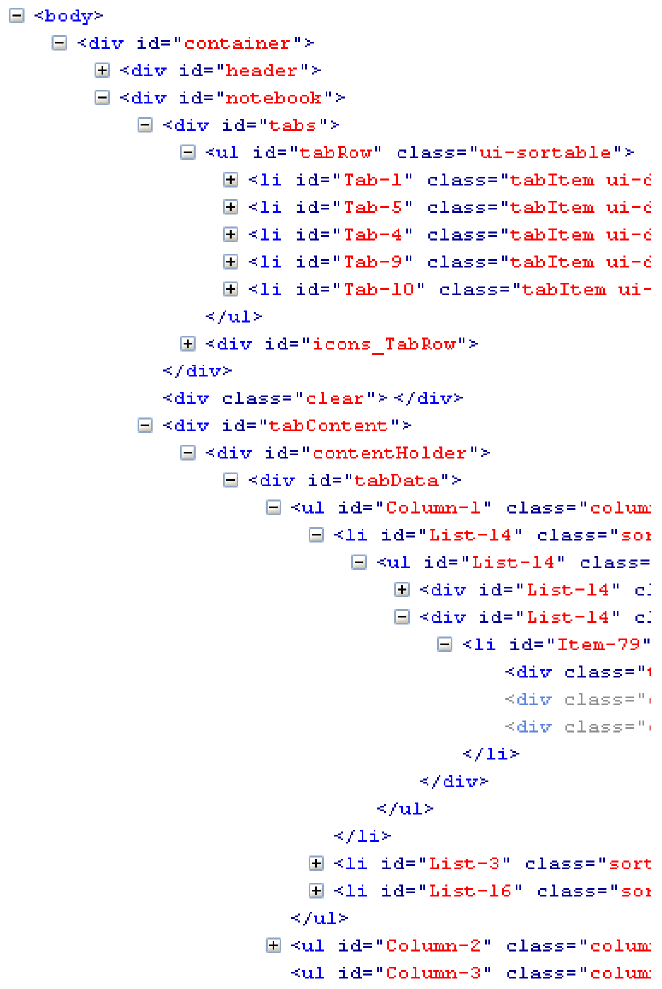
HtmlUnit

```
new HtmlUnitDriver() {
    @Override
    protected WebClient modifyWebClient(WebClient client) {
        WebClient noWaitClient = super.modifyWebClient(client);
        noWaitClient.setRefreshHandler(new WaitingRefreshHandler());
        return noWaitClient;
    }
};
```

mozilla Firefox

```
FirefoxProfile fp = new FirefoxProfile();
//"fast" before 2.19
fp.setPreference("webdriver.load.strategy", "unstable");
```

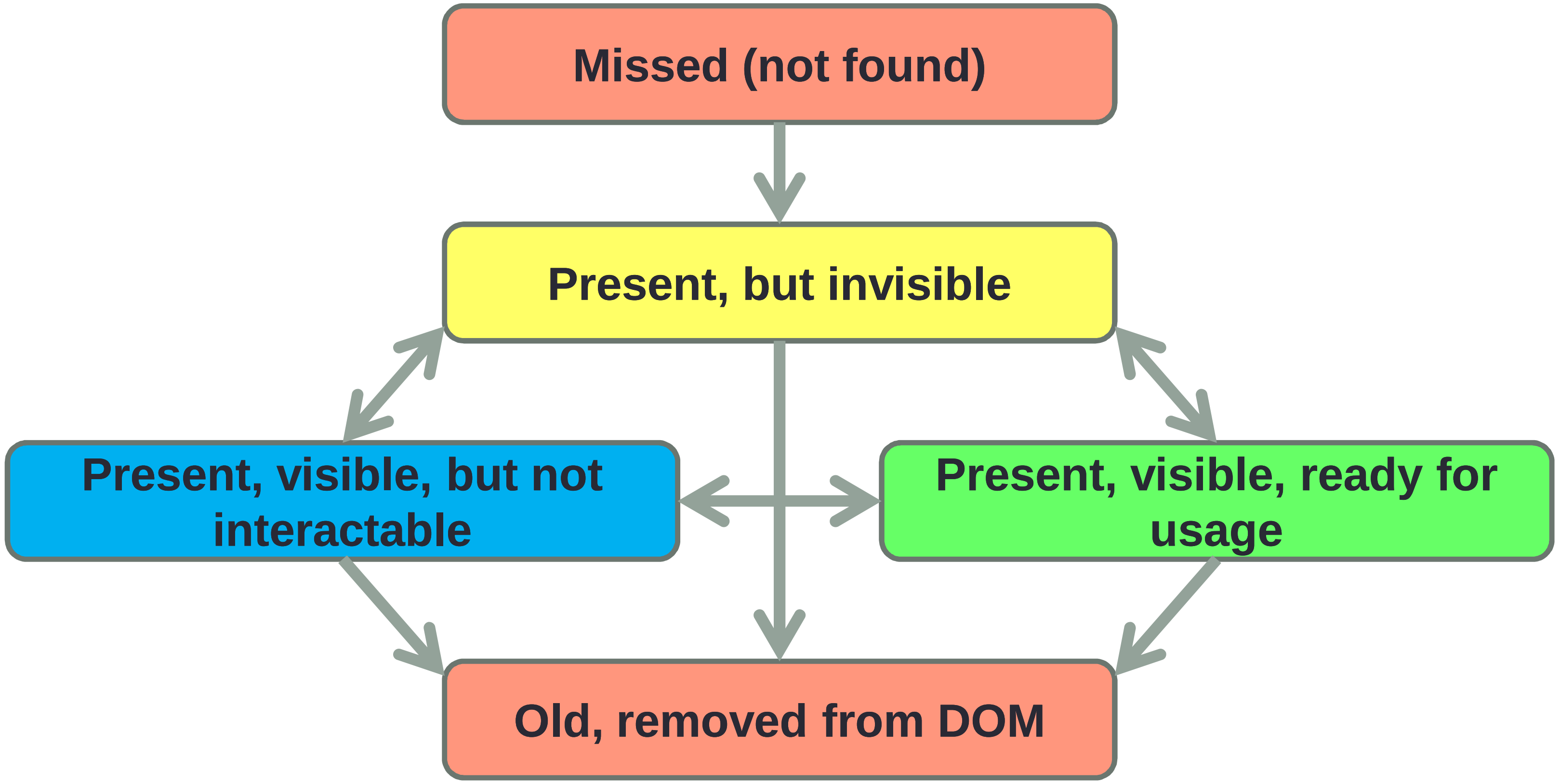
WebDriver works with DOM



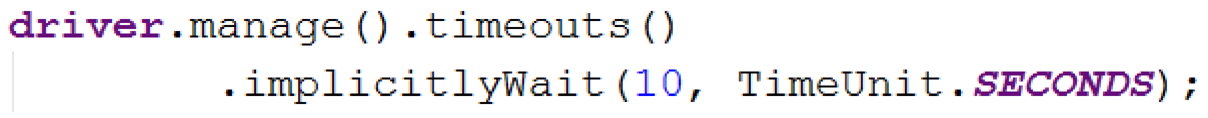
```
⊟ <body>
  ⊟ <div id="container">
    ⊞ <div id="header">
    ⊟ <div id="notebook">
      ⊟ <div id="tabs">
        ⊟ <ul id="tabRow" class="ui-sortable">
          ⊞ <li id="Tab-1" class="tabItem ui-d
          ⊞ <li id="Tab-5" class="tabItem ui-d
          ⊞ <li id="Tab-4" class="tabItem ui-d
          ⊞ <li id="Tab-9" class="tabItem ui-d
          ⊞ <li id="Tab-10" class="tabItem ui-
          </ul>
        ⊞ <div id="icons_TabRow">
        </div>
        <div class="clear"></div>
      ⊟ <div id="tabContent">
        ⊟ <div id="contentHolder">
          ⊟ <div id="tabData">
            ⊟ <ul id="Column-1" class="colum
              ⊟ <li id="List-14" class="so
                ⊟ <ul id="List-14" class=
                  ⊞ <div id="List-14" c
                  ⊟ <div id="List-14" c
                    ⊟ <li id="Item-79"
                        <div class="
                        <div class="
                        <div class="
                      </li>
                    </div>
                  </ul>
                </li>
              ⊞ <li id="List-3" class="sort
              ⊞ <li id="List-16" class="so
              </ul>
            ⊞ <ul id="Column-2" class="colum
              <ul id="Column-3" class="colum
```

- Everything is dynamic
- Complex unreliable structure
- JavaScript libraries
- Third-party widgets
- Rules are broken sometimes

DOM element workflow


Missed (not found)
Present, but invisible
Present, visible, but not interactable
Present, visible, ready for usage
Old, removed from DOM

WebDriver is clever enough



```
driver.manage().timeouts()
        .implicitlyWait(10, TimeUnit.SECONDS);
```

- Waiting on browser side
  - Can’t stop earlier
  - *findElements* wait for at least one element
  - May become hidden cause of long tests
- Doesn’t work for element presence check
- Works for all *findXXX* methods transparently

Manual waiting is available

```java
private void waitForSuggestions() {
    new WebDriverWait(driver, 30).until(new Predicate<WebDriver>() {
        public boolean apply(WebDriver webDriver) {
            By selector = By.cssSelector(".ac_results");
            return webDriver.findElement(selector).isDisplayed();
        }
    });
}
```

Lots of ready to use wait conditions

```
WebDriverWait wait = new WebDriverWait(driver, 30);
wait.until(presenceOfElementLocated(By.id("foo")));
wait.until(titleIs("title"));
wait.until(titleContains("part"));
wait.until(visibilityOfElementLocated(By.id("foo")));
wait.until(invisibilityOfElementLocated(By.id("foo")));
wait.until(stalenessOf(driver.findElement(By.id("foo"))));
wait.until(textToBePresentInElement(By.id("user"), "text"));
wait.until(frameToBeAvailableAndSwitchToIt("main"));
wait.until(elementToBeClickable(By.id("foo")));
```

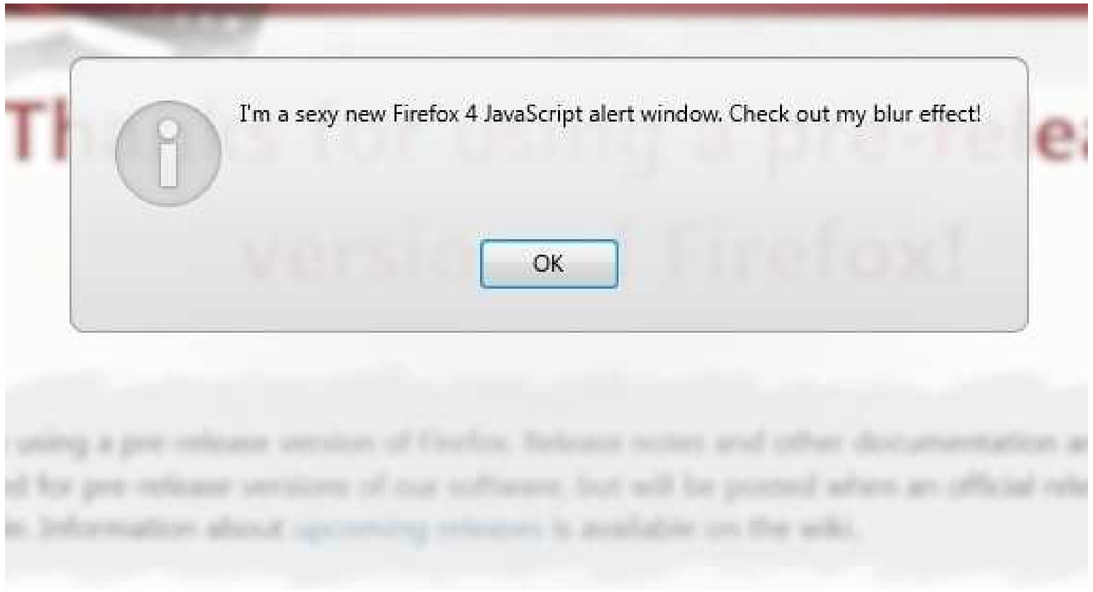
FluentWait for tuned configuration

```java
// Waiting 30 seconds for an element to be present on the page,
// checking for its presence once every 5 seconds.
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
        .withTimeout(30, TimeUnit.SECONDS)
        .pollingEvery(5, TimeUnit.SECONDS)
        .ignoring(NoSuchElementException.class);

WebElement foo = wait.until(new Function<WebDriver, WebElement>() {
    public WebElement apply(WebDriver driver) {
        return driver.findElement(By.id("foo"));
    }
});
```

Hope you don’t use Alerts…

- *UnhandledAlertException* may break each test
- No more *‘hung forever’* mode on alerts
- *driver.switchTo().alert()* waits 2 seconds in Firefox
- *alertIsPresent* method in *ExpectedCondition*


I'm a sexy new Firefox 4 JavaScript alert window. Check out my blur effect!

Windows are not so simple

- Use *driver.getWindowHandles()* to store all windows before action
- Wait until list of windows is changed
- Use *driver.switchTo().window(handle)* to switch
- But not so quick ☺ : http://code.google.com/p/selenium/issues/detail?id=2764
- And don’t forget to return back when window is closed

You need to fire ‘AJAX’ event

- Text typing
- Slow typing
- Key press/up/down
- Key combinations
- Mouse move/over/up
- Left or right click
- Drag and drop
- Double click
- Focus

WebDriver has some syntax sugar

```
Select country = new Select(driver.findElement(By.id("country")));
country.selectByValue("US");
country.selectByIndex(3);
country.selectByVisibleText("United States");
if (country.isMultiple()) {
    List<WebElement> options = country.getAllSelectedOptions();
    options.get(4).click();
}
```

Mouse operations

- You can click on any VISIBLE element
- OS events are emulated
- DOM events are processed as always
- Click is performed in the center of element area
- Auto scrolling is performed for click, but not reliable ☺

Keyboard operations

- You can type on any VISIBLE element
- OS events are emulated
- DOM events are processed as always
- Every key is typed separately, so keyDown/keyUp/keyPress are fired for each key

```
FirefoxProfile p = new FirefoxProfile();
p.setEnableNativeEvents(false);
```

Want to speedup?

- Ctrl-A/Ctrl-C/Ctrl-V work well for large text
- JavaScript code to change element value directly
- For file inputs will work quickly by default

Actions for experts

- moveToElement
- contextClick
- clickAndHold
- release
- dragAndDrop
- moveToElement
- moveByOffset
- dragAndDropBy
- keyUp
- keyDown

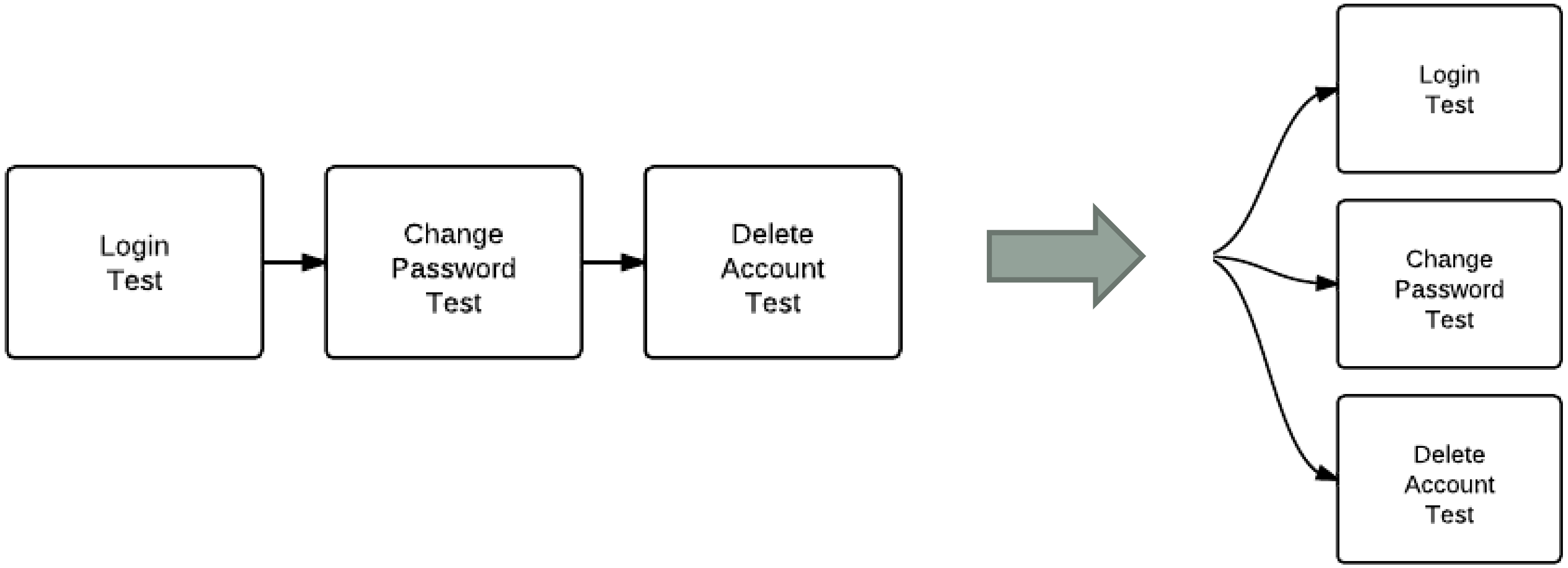
AJAX testing principles

- Workflow:
  - Try to understand what action is needed to start
  - Fire it as directly as possible
  - Try to understand what changed from end user perspective
  - Wait for these changes
  - Assert expected state
- Advices
  - Don’t use pauses and sleeping
  - Take a look at AjaxElementLocator for PageObject pattern
  - Use DOM viewers to understand page structure

Performance tips

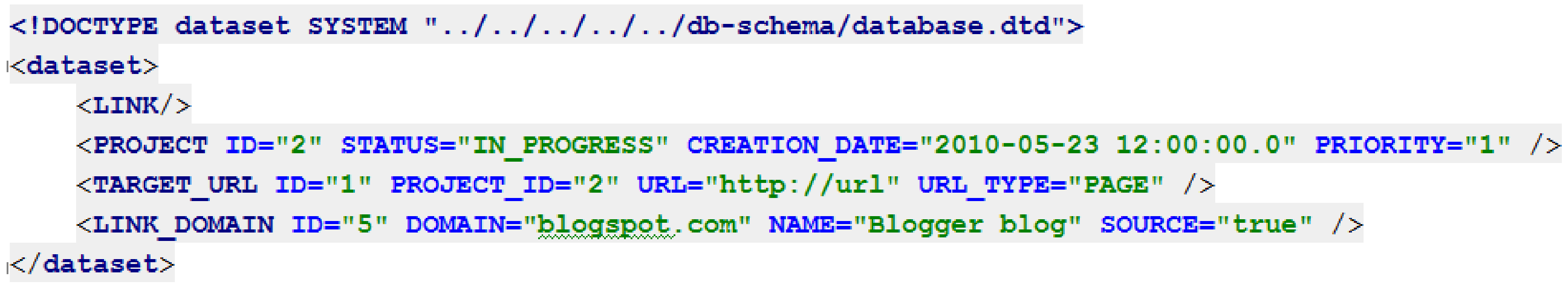
Tip #1: Data independent tests

- Most dependencies are data related
- Dependent tests = no parallel execution
- Data should be test specific with no reuse
- Use small focused datasets



Test data generation techniques

- Use Registry with counter to generate unique data
- Fill database with large amount of data and use reservation
- Use database sharding on application side
- Shard data by unique key (user name, email, etc.) and insert data sets with DbUnit



```
<!DOCTYPE dataset SYSTEM "../../../../../db-schema/database.dtd">
<dataset>
    <LINK/>
    <PROJECT ID="2" STATUS="IN_PROGRESS" CREATION_DATE="2010-05-23 12:00:00.0" PRIORITY="1" />
    <TARGET_URL ID="1" PROJECT_ID="2" URL="http://url" URL_TYPE="PAGE" />
    <LINK_DOMAIN ID="5" DOMAIN="blogspot.com" NAME="Blogger blog" SOURCE="true" />
</dataset>
```

Tip #2: Atomic focused tests

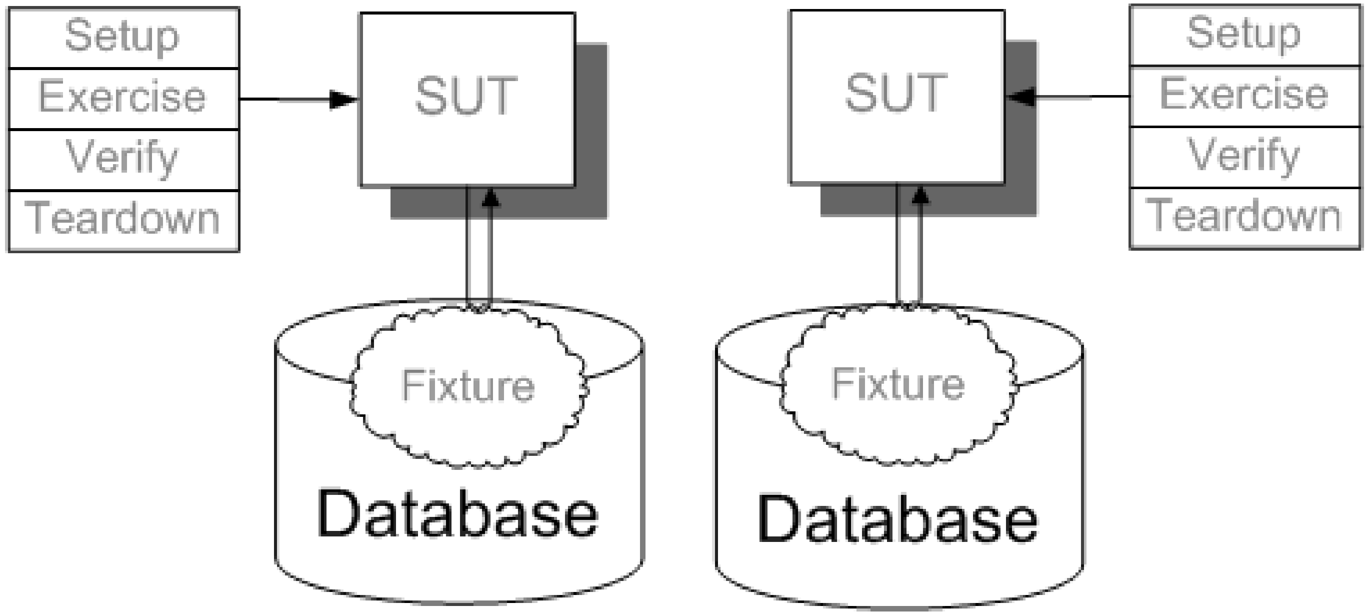
- Small test has clear goal and easy to understand
- Easy to divide in suites
- Flexible running in multiple stages
- Report failures are easier to understand
- Run quickly so higher level of parallelization

Tip #3: Test only functionality

- Don’t pay attention on design and content
- Simplify everything except functionality under test
- Run complex tests in reliable browsers

Tip #4: Generate application state

- Insert data directly in data storage
- Don’t use complex UI to generate state
- Use small isolated datasets and simple tools



Tip #5: Test widgets in isolation

- Create unit tests for JavaScript code
- Load widget on empty page and test it well
- Try to use reliable widgets library

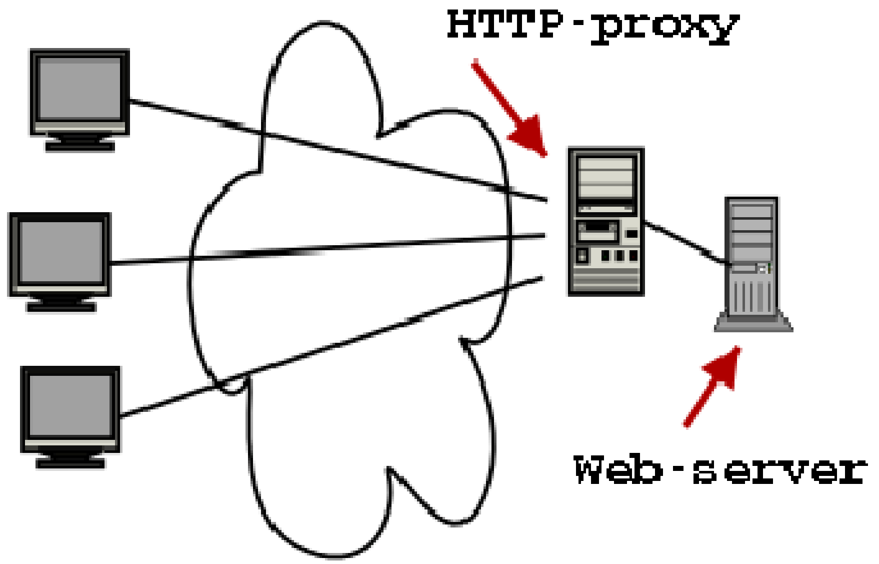
LATEST TALK
Which Movies Made You Cry This Year?
What Should American Horror Story Do for Its Second Season?
SEE ALL CONVERSATIONS
LinkedIn
YouTube
Stumble
RSS
More subscription options »
Subscribe to the Newsletter:
Subscribe
Join Over 321000 Readers
Business Insider Mobile
GET IT NOW
Popular Articles
Is Your Tribe Holding You Down?
Are Social Media "Experts" Worthless?
REAL-TIME CRUNCHBASE
All Activity
MOST COMMENTED

Tip #6: Isolate all third-parties

- Use quick fake email server
- Mock all external services
- Switch all third-party components in quick predictable mode
- Run everything locally

Use HTTP proxy for texts

- Blacklist external resources (Facebook, Twitter, Ads, etc.)
- Cache images and other nonfunctional resources
- Collect HTTP traffic for analysis (404, redirects, loading time, etc.)
- Speedup page loading

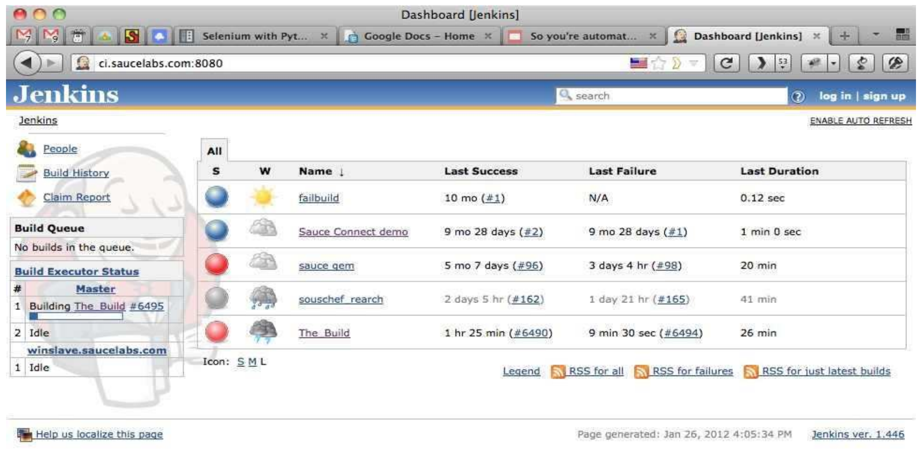


Tip #7: Use smart waiting

- Every delay is multiplied by number of tests
- Don’t use speed and pauses at all
- Use implicit waits carefully
- Always think about the worst scenario and set good timeouts

Tip #8: Monitor your tests

- Use CI server to gather time metrics
- Check trends to select slow tests
- Spend some time to improve them regularly
- Set max allowed tests execution time and try to archive it



Build Queue

No builds in the queue.

Build Executor Status

#	Master
1	Building The_Build #6495
2	Idle
	winslave.saucelabs.com
1	Idle

All

S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	failbuild	10 mo (#1)	N/A	0.12 sec
●	☁	Sauce Connect demo	9 mo 28 days (#2)	9 mo 28 days (#1)	1 min 0 sec
●	☁	sauce gem	5 mo 7 days (#96)	3 days 4 hr (#98)	20 min
●	☁	souschef_rearch	2 days 5 hr (#162)	1 day 21 hr (#165)	41 min
●	☁	The_Build	1 hr 25 min (#6490)	9 min 30 sec (#6494)	26 min

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

Help us localize this page

Page generated: Jan 26, 2012 4:05:34 PM Jenkins ver. 1.446

Questions & Answers