TESTING TOOLS

Objectives

- At the end of this lecture, you should be able to
  - Describe some common software tools
  - Describe how different parts of a multi-tier application are tested
  - Describe how databases can be tested/monitored

Web Page testing

- API will send
  - Requests to load up web pages
  - Clicks of buttons (submit) and links
  - Setting of test fields in web forms
- API will parse response to determine
  - Title of page
  - Error status
  - Validity of fields on page

JWebUnit

- Provides web site testing API
- Features
  - Programmatic testing control
  - Location, click, page parsing
  - Javascript execution
- Limitations
  - Only some core javascript is executed and understood
  - Slower than browser, Javascript, v.slow

JWebunit Example

```
HtmlUnitTestingEngineImpl webunit=new HtmlUnitTestingEngineImpl();
        webunit.setScriptingEnabled(false);
        try {
        webunit.beginAt(new
URL("http://seb.fruitserver.com/mobile/lobby/Default.aspx"),new
TestContext());
        } catch (Exception exc) {
                exc.printStackTrace();
                System.exit(1);
        }
        webunit.clickLinkWithExactText("Sign In",0);
        System.out.println("Page title is
"+webunit.getCurrentPageTitle());
        webunit.setTextField("tbEmail", "testsc01");
        webunit.setTextField("tbPIN", "1608");
        webunit.submit();
        System.out.println("Page title is
"+webunit.getCurrentPageTitle());
```

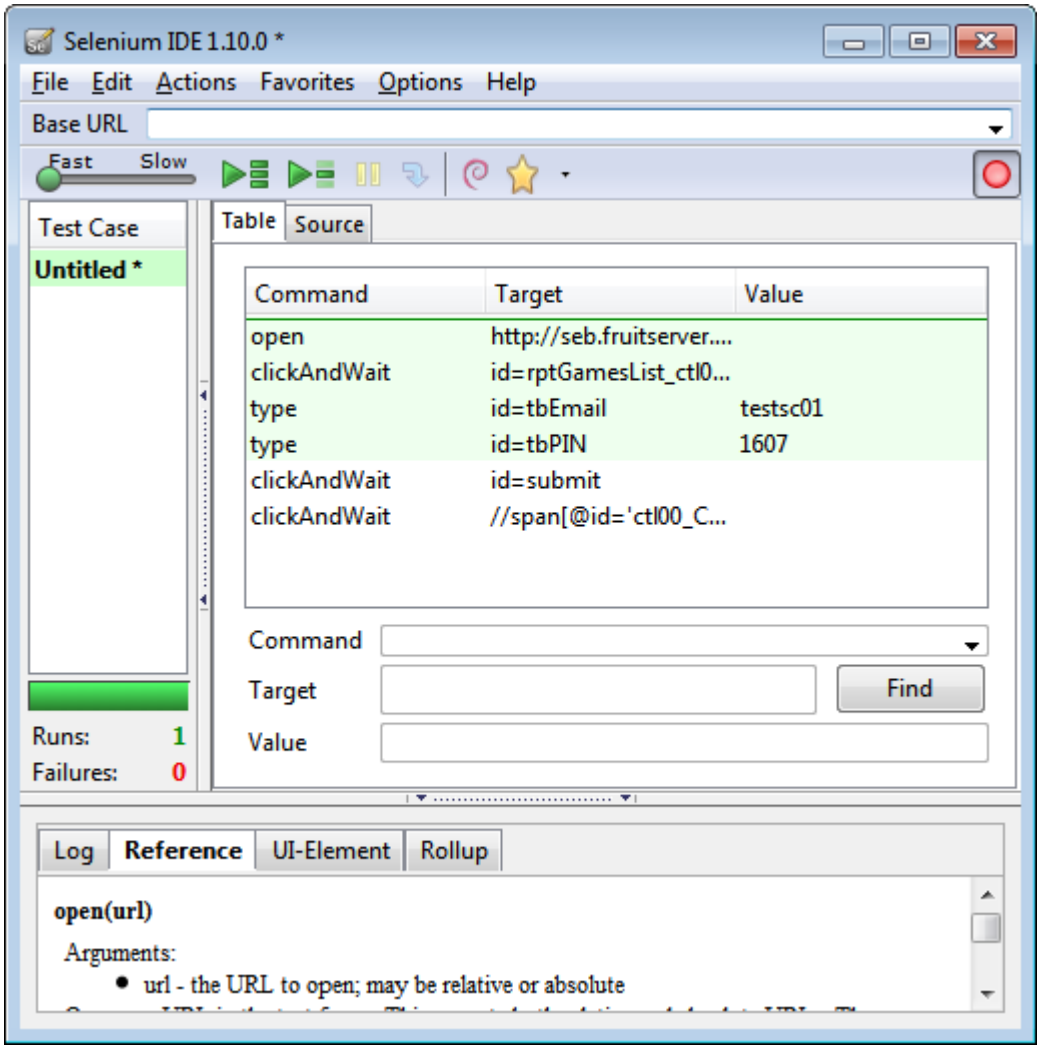
Remote controlled browser testing

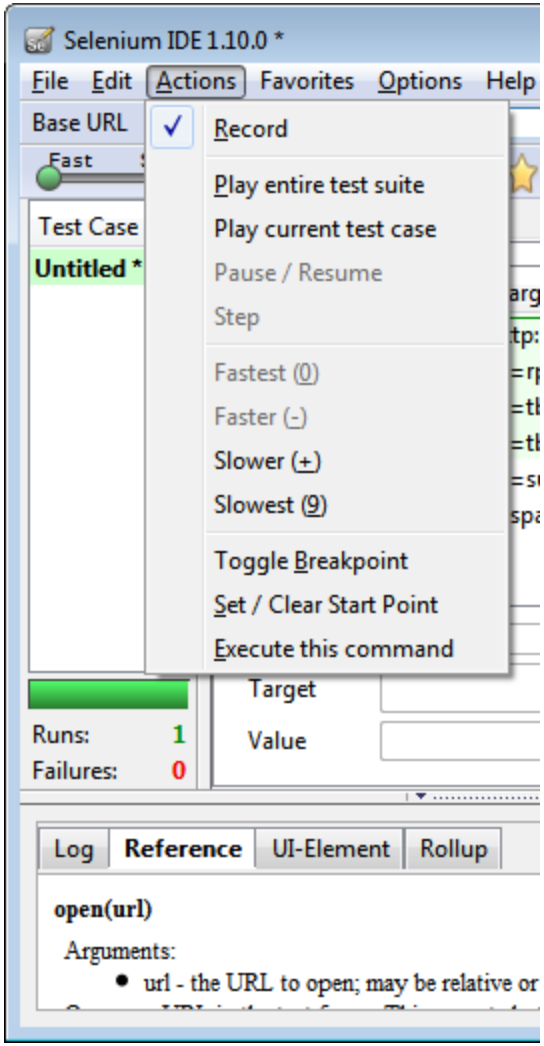
- Loads up the browser
- Requests browser to open page
- Populates forms on page with data
- Goes to next page
- Example
  - Selenium
    - IDE plug for Firefox (record, playback, web actions)
    - Remote control API (Firefox, Chrome, iOS)

Why remote control

- Software doesn’t run in simulated but real environment
- Javascript/css support is different in different browsers
- Performance changes across browsers
- Javascript interactions such as Ajax will work properly

Selenium IDE



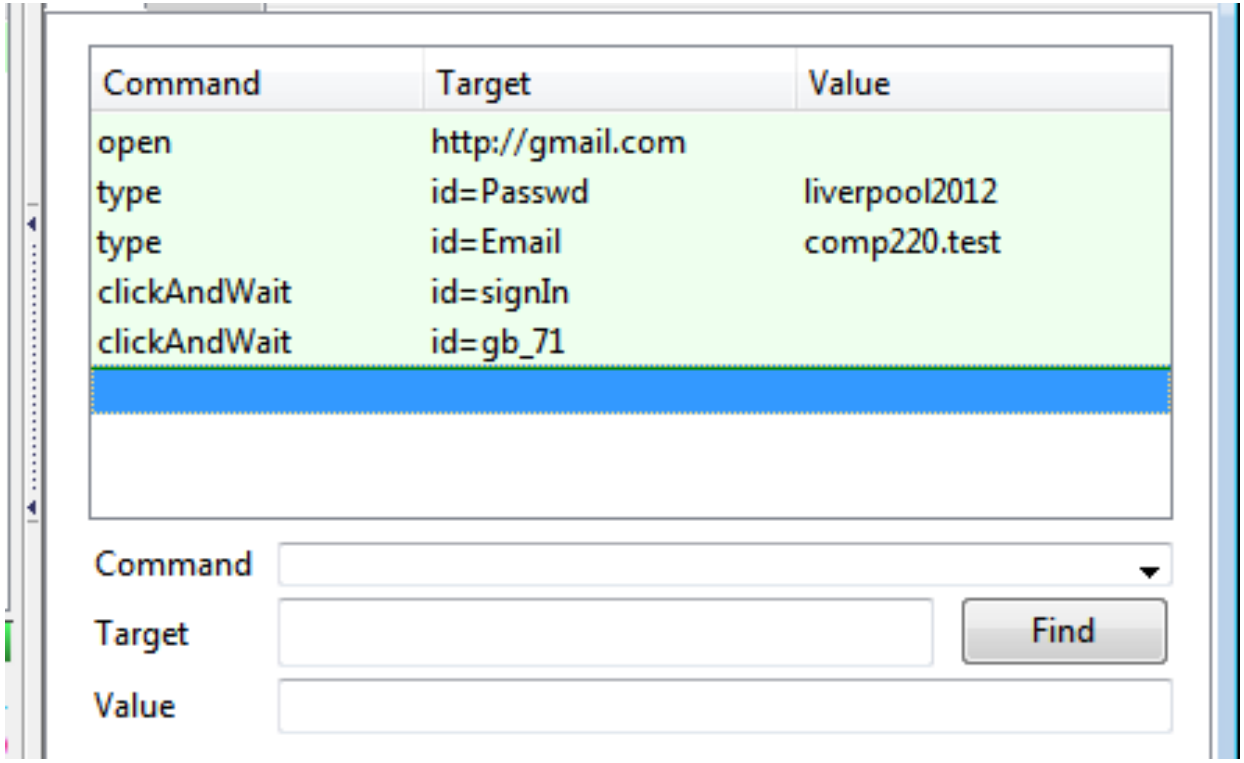


Selenium IDE commands

- All commands have name, target, value
- open
  - Opens a page
  - Arguments
    - target = URL
    - target=http://gmail.com
- type
  - Types text into text box
  - target (identity of element)
    - id=
    - name=
  - value
    - Value to type in

Example open gmail account

- Username = comp220.test
- Password = liverpool2012

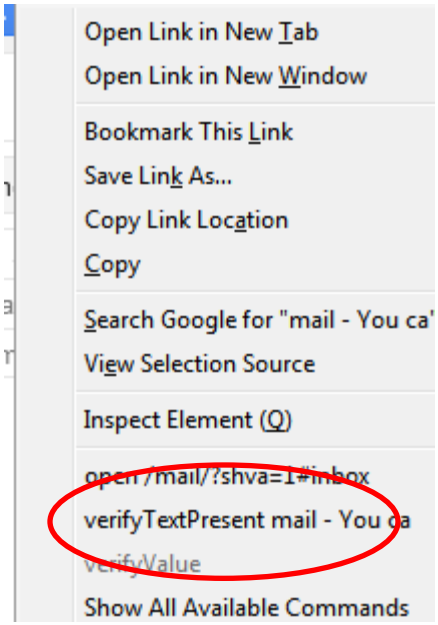


clickAndWait

- Clicks an element on the web page and then waits for it to load
- Target defined as
  - id
  - name

Assertions

- Test if values are what is expected
- Adding an assertion using Selenium IDE
  - Right click on element
  - Look for
    - VerifyTextPresent



Test case format

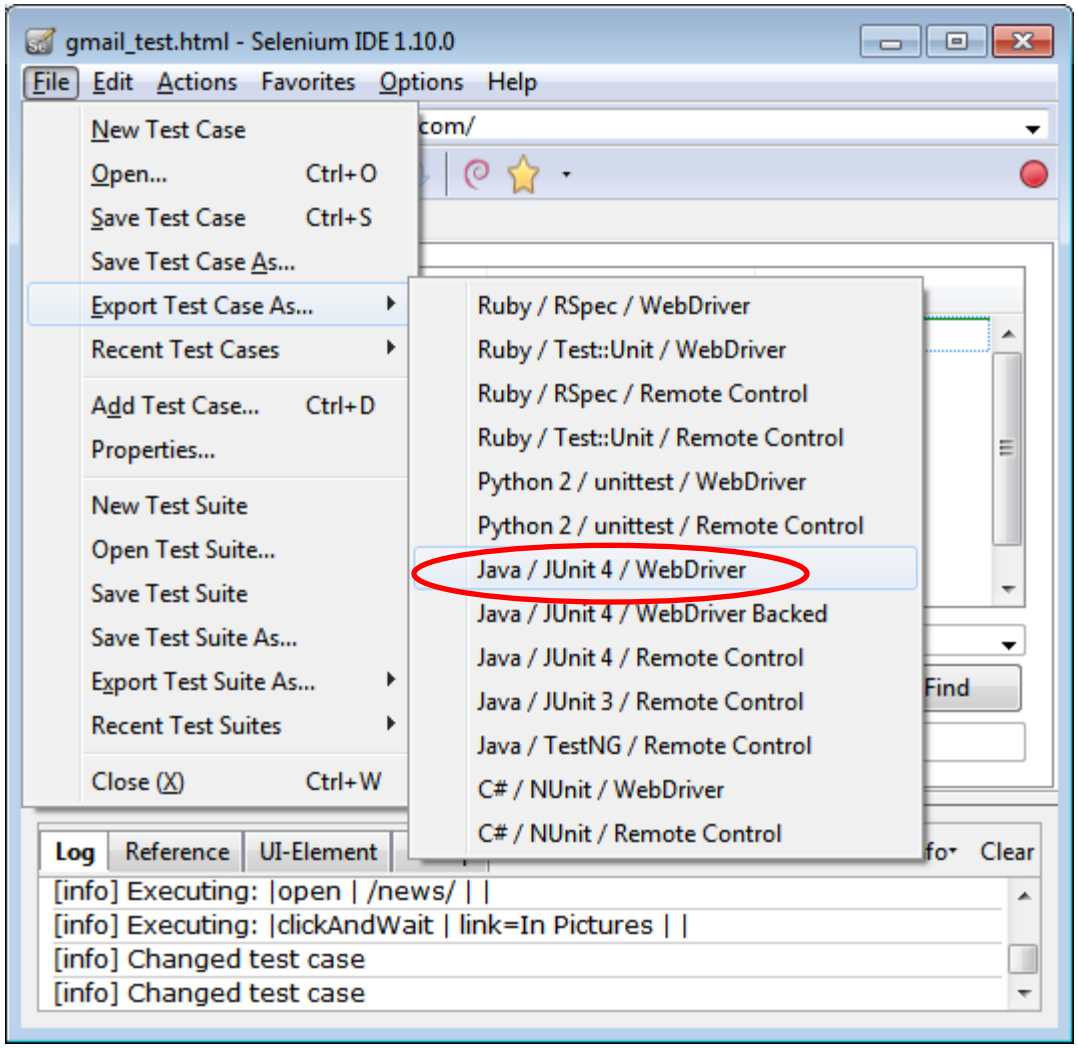
```
<tr>
        <td>open</td>
        <td>http://gmail.com</td>
        <td></td>
</tr>
<tr>
        <td>type</td>
        <td>id=Passwd</td>
        <td>liverpool2012</td>
</tr>
<tr>
        <td>type</td>
        <td>id=Email</td>
        <td>comp220.test</td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>id=signIn</td>
        <td></td>
</tr>
```

Selenium web driver

- Connects your program to browser to allow remote control
- Example code…

```
driver = new ChromeDriver();
driver.get("http://gmail.com");
driver.findElement(By.id("Passwd")).clear();
driver.findElement(By.id("Passwd")).sendKeys("liverpool 2012");
driver.findElement(By.id("Email")).clear();
driver.findElement(By.id("Email")).sendKeys("comp220.test");
driver.findElement(By.id("signIn")).click();
```

Exporting test cases to program



Test case exported

```
@Before
        public void setUp() throws Exception {
       System.setProperty("webdriver.chrome.driver",
"C:\\comp285\\chromedriver.exe");
          driver = new ChromeDriver();
          baseUrl = "https://accounts.google.com/";
          driver.manage().timeouts().implicitlyWait(30,
TimeUnit.SECONDS);
        }

        @Test
        public void testGmailTestWebdriver() throws Exception {
          driver.get("http://gmail.com");
          driver.findElement(By.id("Passwd")).clear();
          driver.findElement(By.id("Passwd")).sendKeys("liverpool2012");
          driver.findElement(By.id("Email")).clear();
          driver.findElement(By.id("Email")).sendKeys("comp220.test");
          driver.findElement(By.id("signIn")).click();
```

Test case exported

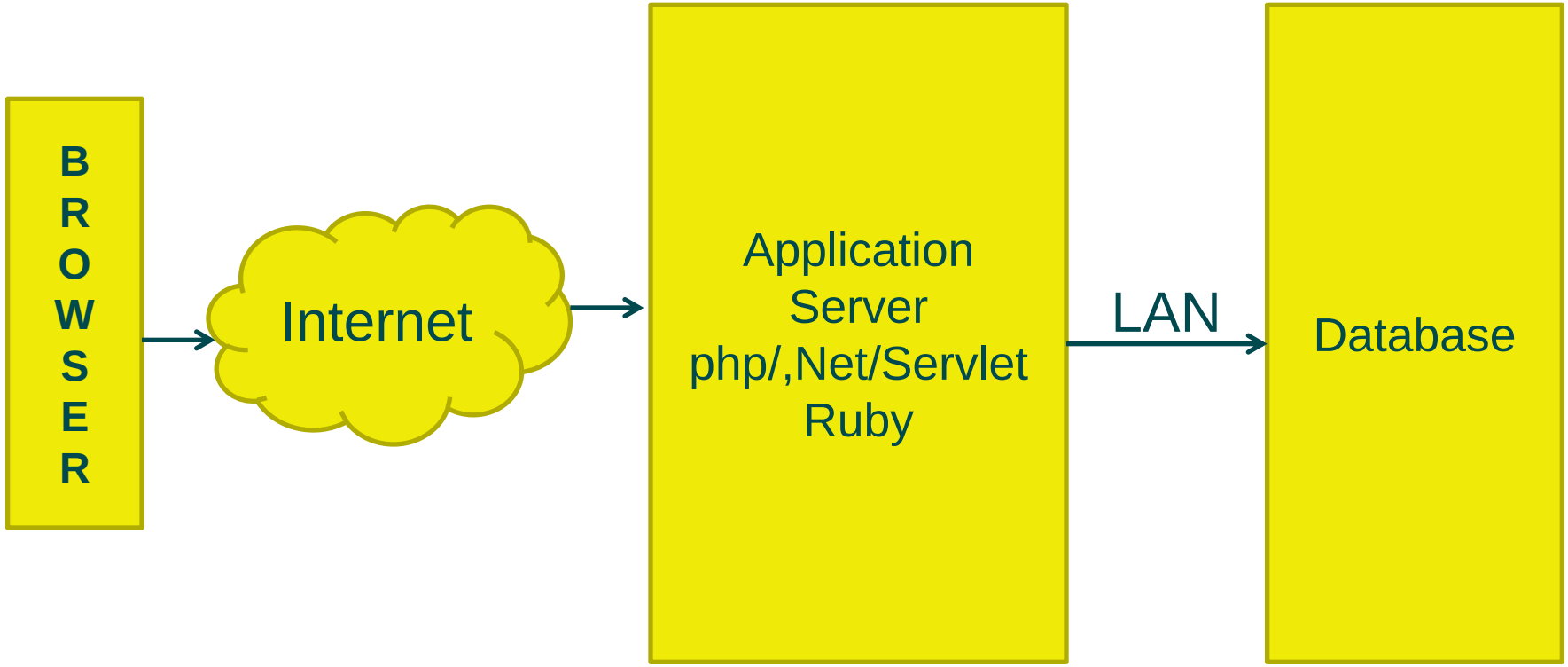
```
        try {

assertTrue(driver.findElement(By.cssSelector("BODY")).getText().matches(
"^[\\s\\S]*comp220\\.test@gmail\\.com[\\s\\S]*$"));
        } catch (Error e) {
          verificationErrors.append(e.toString());
        }
assertTrue(driver.findElement(By.cssSelector("BODY")).getText().matches(
"^[\\s\\S]*comp220\\.test@gmail\\.com[\\s\\S]*$"));
        driver.findElement(By.id("gbgs4dn")).click();
        driver.findElement(By.id("gb_71")).click();
      }
```

Test case exported

```
@After
public void tearDown() throws Exception {
  driver.quit();
  String verificationErrorString = verificationErrors.toString();
  if (!"".equals(verificationErrorString)) {
    fail(verificationErrorString);
  }
}
```

Performance testing

- Typical architecture



Performance testing

- Assuming a 3-tier architecture, problems with performance could be
  - In browser (e.g. javascript performance issue)
  - In transmission network (e.g. size of download, images, jscript, HTML)
    - Particularly true with mobile network
  - Within LAN
  - At Application server
  - At Database server

Testing Performance problems

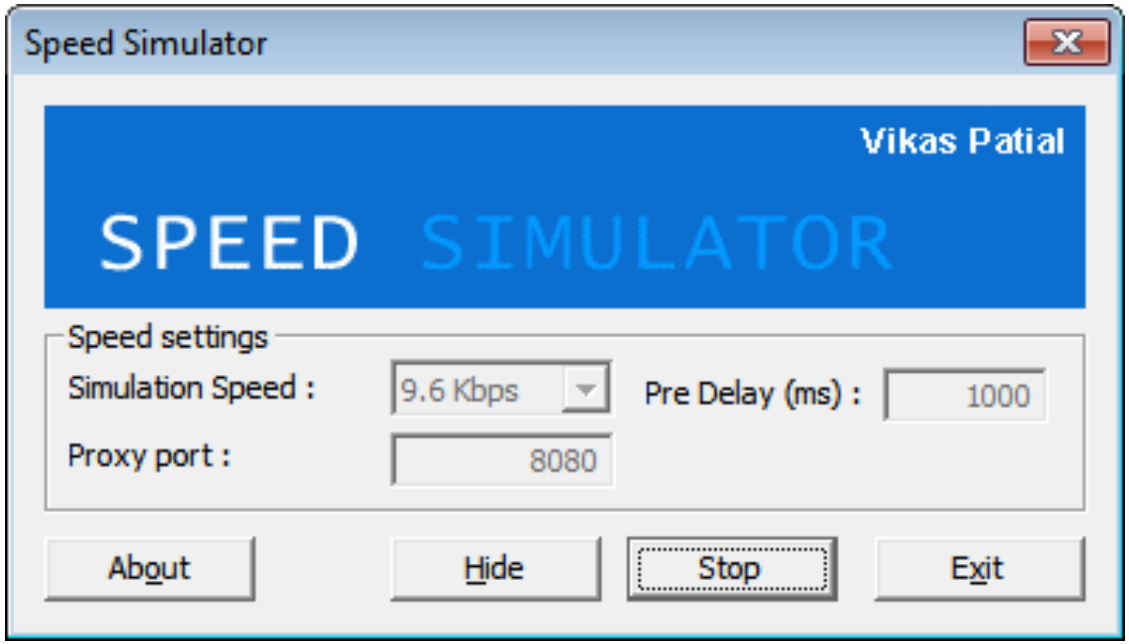
- Browser side
  - Function of content and browser
- So test needs to be done
  - On browser
  - But automatically
- Needs support for
  - Remote control of browser
  - See last topic

Internet speed simulation

- Slow internet connections cause
  - Connections to timeout
  - Requests to fail
  - User experience may suffer
- Issues
  - Software may send multiple requests
  - Loading screens need testing
  - Code can measure link speed, then optimize
  - With mobile applications slow network speed is common

Internet Speed Simulation

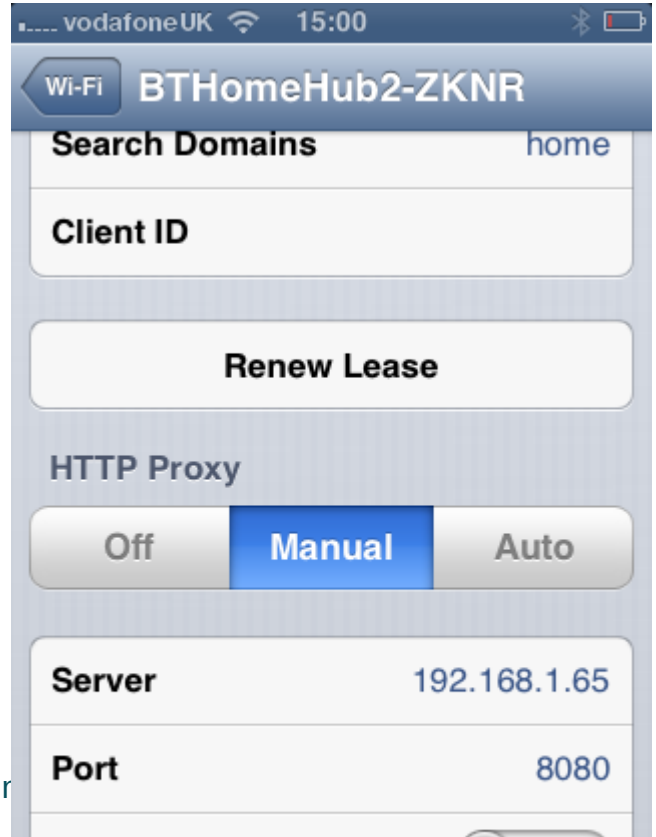
http://www.ngcoders.com/downloads/internet-speed-simulator-proxy-throttler/



Browser connects to speed simulator via proxy interface

Configuring the proxy interface

- Firefox
  - Tools/Options/Advanced/Network/Settings
- iPhone
  - Settings/WiFi/Choose a Network



Android proxy configuration

- Applications/Settings/Wi-Fi
- Tap and hold Wi-Fi connection, then Modify Network
- Check Show Advanced options
- Enter proxy via manual settings

Server side performance testing

- Application server
  - Performance related to request load and service time
  - In general traffic measured in Erlangs
  - Erlang = service time/request interval
  - So
    - A load of 200 requests per second = request interval = 5 ms
    - If each request takes on average 3 ms to service
    - Load = 3/5 = 0.6 Erlangs

Application server/Database loading

- If the transaction only involves the middle tier
  - Performance generally only current load
- Database transaction time depends on
  - Database load
  - State of database
    - Size of tables
    - Complexity of transaction (table joins etc.)
    - Table locking or transactional operations

Example SQL

- Balance query
  - Select sum(amount) from account_transaction where userid=1234 and account=25
- Login
  - select * from users where login="fred"
- Joins (orphan transaction search)
  - select TransactionID from user as a right outer join account_transaction as b on a.CustomerID = b.CustomerID where a.CustomerID is null;

Bench marking queries

- Important to benchmark queries with given sizes of table
  - Looking up user/password in
    - 10,000 users
    - 250,000 users
  - Find transaction balance for table with
    - 100,000 or 250,000,000 transactions
- At what point will service suffer
- Need to insert alerts when tables grow to critical size

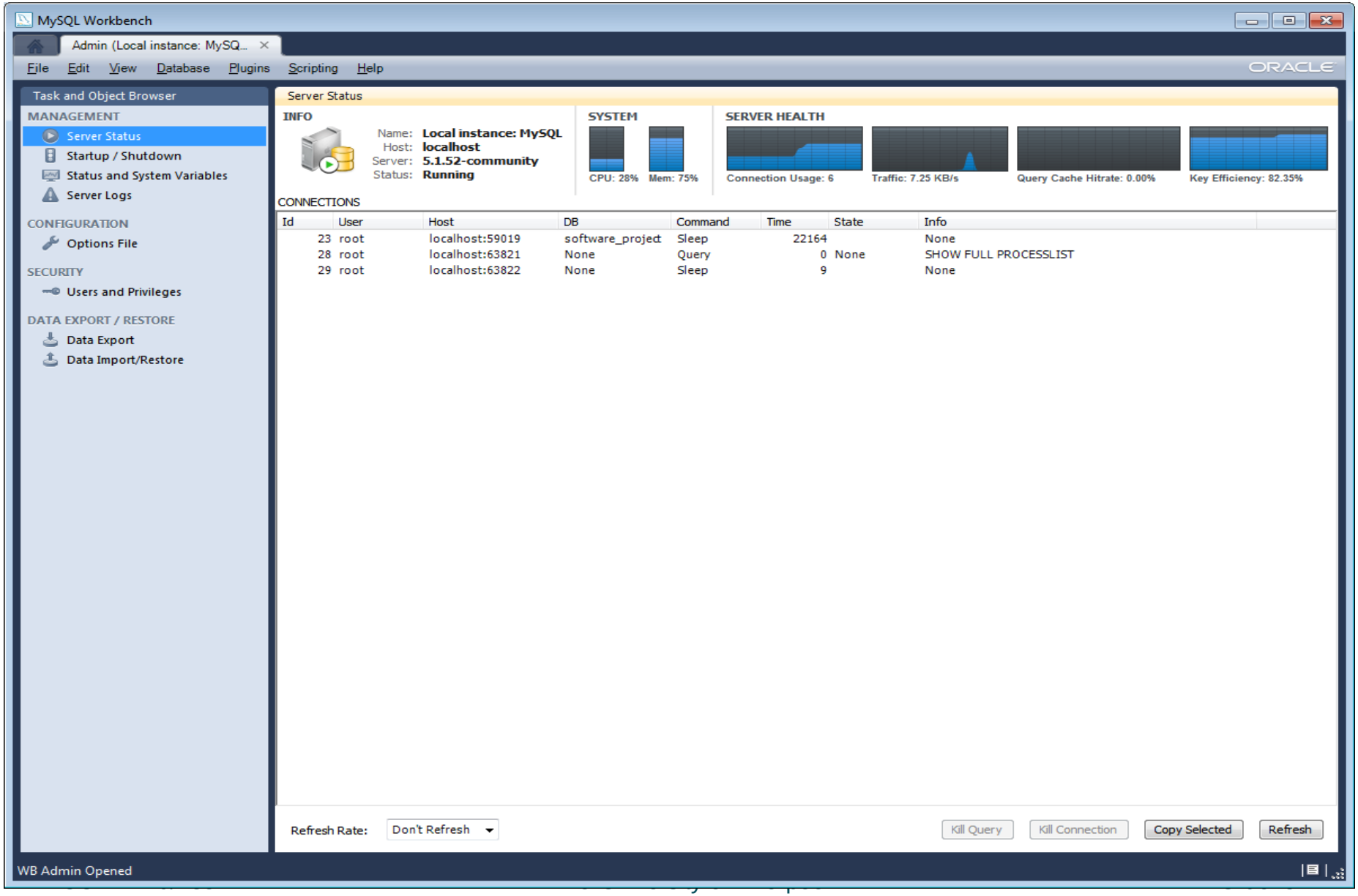
mysqlslap

- Used to stress test your database
- Can run this
  - On it own to get benchmark
  - Concurrently with application
- Parameters
  - --concurrency=500
  - --iterations=20000
  - --query
  - --user, --password

Database load testing

- Have test patterns which
  - Do complex operations such as creating many new accounts
  - Fill up tables with multiple transactions
  - Do many different transactions at the same time
    - Creating accounts
    - Logging in
    - Performing transactions

Database monitoring



Database monitoring and alerting

- Monitoring
  - Examine in real time state of database
  - Heavy loading could be due to
    - Normal increase in customer traffic
    - DOS attack
    - Problems with table scaling
- Alerts
  - Certain load conditions
  - Slow queries
  - Too many SQL errors

MySQL slow query log

- Can be configured to store queries which take over a certain time threshold
  - e.g. >10 seconds
- Slow queries will
  - Cause that user to suffer poor service
  - Push other queries back up the queue
  - If combined with table locking, cause other queries to lock out

Summary

- Testing involves
  - Strategy of what to test
- Need to test
  - All parts of multi-tier application
  - Under different conditions
  - Database performance changes as it grows
- Need for monitoring
  - Keep testing/monitoring live site