Experience with Selenium 2 WebDriver

'Linking to Data Generation and Existing Test Frameworks'

Austin Chamberlin
Gopal Addada
Nil Weerasinghe

Agenda

- Introduction to Selenium 2 - WebDriver
- RC vs WebDriver
- Our experience
- Interesting WebDriver APIs/Classes
- Observations and caveats
- Integration with Selenium GRID 2
- To the Cloud
- Integration with TestNG
- Test Coverage of Spacebook
- QA

Introduction to Selenium 2 WebDriver

- Browser automation.
    - Most popular free Automation in the land of QTP, TestComplete and copious other paid automation tools.

- Marrying of WebDriver and Selenium 1 Projects.

- Native Browser interaction/better user action simulation.

Why we do WebDriver the way we do...
Architecture overview
http://www.aosabook.org/en/selenium.html

Latest WebDriver Talk
http://www.youtube.com/watch?v=OsNkcUq0veI

RC vs Web Driver

- WD
  - Cleaner and better API for OO automation development
  - Simpler API
  - HTML Unit Support
  - IE 9 Support
  - IOS/Android
  - Better/Native Browser Interaction
  - Better popup and dialogs handling
  - Window focus issues not a problem
  - No RC, Cleaner multiple Window Interaction

- type
- typeKeys
- typeKeysNative
- keydown
- keypress
- keyup
- keydownNative
- keypressNative
- keyupNative
- attachFile

Here's the equivalent in the WebDriver API:

- sendKeys

- RC
  - Stable - Maintenance Mode
  - Better browser support
  - Easier bug fixing/maintenance due to injection
  - Better language support
    - Java/Ruby/Python/C#
    - Perl/PHP/NodeJS

Our Experience

Selenium

GRID

Interesting WebDriver APIs/Classes

WebDriver

IE

```
driver = new InternetExplorerDriver();
```

FFX

```
File file =new File(selenium2PropertyReader.get("ffx.profile"));
FirefoxProfile profile = new FirefoxProfile(file);
```

HTMLUNIT

```
driver = new HtmlUnitDriver(com.gargoylesoftware.htmlunit.BrowserVersion.FIREFOX_3_6);
((HtmlUnitDriver)driver).setJavascriptEnabled(true);
```

CHROME

```
System.setProperty("webdriver.chrome.driver", selenium2PropertyReader.get("chrome.location"));
```

GRID

```
capability = DesiredCapabilities.firefox();
capability.setBrowserName(DesiredCapabilities.firefox().getBrowserName());
capability.setCapability(FirefoxDriver.PROFILE, profile);

capability.setPlatform(org.openqa.selenium.Platform.ANY);
driver = new RemoteWebDriver(new URL(selenium2PropertyReader.get("selenium.grid.url")), capability);
```

WebElement

```
WebElement search = driver.findElement(By.name("q"));
```

- className(String className) : By - By
- cssSelector(String selector) : By - By
- id(String id) : By - By
- linkText(String linkText) : By - By
- name(String name) : By - By
- partialLinkText(String linkText) : By - By
- tagName(String name) : By - By
- xpath(String xpathExpression) : By - By
- ByXPath - org.openqa.selenium.By
- ByClassName - org.openqa.selenium.By
- ByCssSelector - org.openqa.selenium.By
- ById - org.openqa.selenium.By
- ByLinkText - org.openqa.selenium.By
- ByName - org.openqa.selenium.By
- ByPartialLinkText - org.openqa.selenium.By
- ByTagName - org.openqa.selenium.By
- class : Class<org.openqa.selenium.By>

WebDriverWait

```
WebDriverWait wait = new WebDriverWait(driver, Integer.parseInt(selenium2PropertyReader.get("maxWaitTime")), Integer.parseInt(selenium2PropertyReader.get("sleepTime")));
wait.until(presenceOfElementLocated(By.xpath(xpathToWaitFor)))

private static Function<WebDriver,WebElement> presenceOfElementLocated(final By locator) {
    return new Function<WebDriver,WebElement>() {
        @Override
        public WebElement apply(WebDriver driver) {
            if((driver.findElement(locator)).isEnabled() && (driver.findElement(locator)).isDisplayed()) {
                return driver.findElement(locator);
            } else {
                return null;
            }
        }
    };
}
```

JavascriptExecutor

```
WebDriver wd = drivers.get(name);
JavascriptExecutor js = (JavascriptExecutor) wd;
js.executeScript(script);
```

Observations and Caveats

Tips and Tricks

- Multi-threaded execution for efficiency.
    - Wow! On top of GRID and CLOUD...why?
        - Proper Leveraging of GRID and CLOUD parallelism
        - Faster test scripting feedback/debuging automation code.

- Widget library concept.
    - Re-use.
    - Should loosely couple it to driver.

- Model Actions vs Functions.
    - Larger Systems better test readability : Functions.
    - More control on UI interaction : Actions

Observations and Caveats

Tips and Tricks

- IE Slowness on XPATH Processing.
  - JS_XPATH Injection.

```
WDWrapper.injectExecuteJSintoDriver( drID, (new ReadFile(selenium2PropertyReader.get("js.xpath.location"))).getContents()  );

WDWrapper.injectExecuteJSintoDriver( drID, "(document.evaluate(\"" + (frameXPATH==null?"":frameXPATH) + (XPATH==null?"":XPATH)  +
"\", document, null, XPathResult.ORDERED_NODE_ITERATOR_TYPE, null)).iterateNext().click();");
```

  - tagsoup Library for HTML cleanup for XPATH Processing.
    + org.apache.xpath.XPathAPI
    + org.ccil.cowan.tagsoup.Parser

  - All browsers saw improvement.

Observations and Caveats

Caveats

- IE frame handle loss.
  - Need switch to window and frame all the time before action execution.
    + drivers.get(name).switchTo().defaultContent();
    + drivers.get(name).switchTo().frame(drivers.get(name).findElement(By.xpath(frameXPATH[0])));

- IE Window Focus Issues.

- IE Scrolling Issues.

- Constant reloading of DOM done by IE and StaleElementReferenceException

```
        click();
    } catch (org.openqa.selenium.StaleElementReferenceException err) {
        //DANGEROUS, INFINITE LOOP POSSIBLE ***
        logger.error("**** STALE RETRY...");
        e=(WDWrapper.findElement(drID, frameXPATH, XPATH)).e;
        clickAfterEnableAndDisplay();
    }
```

Observations and Caveats

Caveats

- HTMLUnit for LoadTest on a single machine doesn't scale unless on the cloud with dedicated CPU thread.

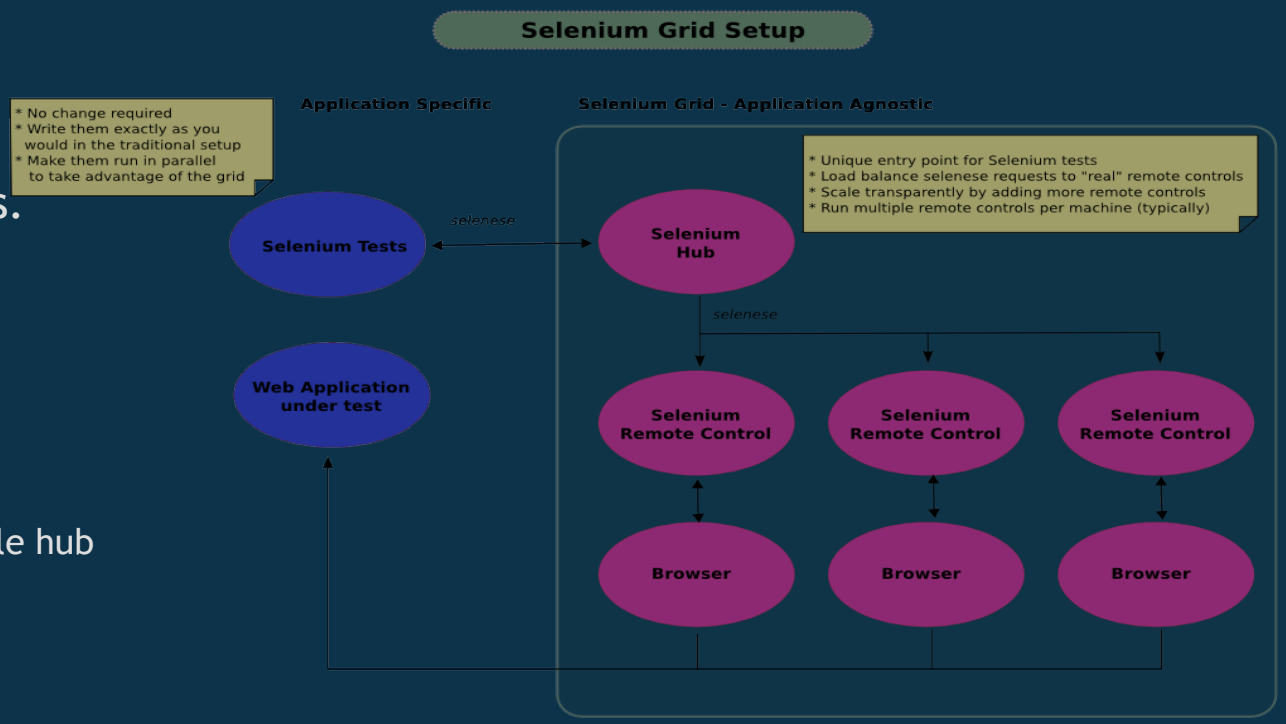
- No NTLM v2 Support for HTMLUNIT.
  - class ExtendedHtmlUnitDriver extends HtmlUnitDriver
  - jcifsEngine

```
WebClient webClient = super.getWebClient();
webClient.setWebConnection(new HttpWebConnection(webClient) {

    @Override
    protected synchronized AbstractHttpClient getHttpClient() {
        AbstractHttpClient httpClient = super.getHttpClient();
        httpClient.getAuthSchemes().register(AuthPolicy.NTLM, new AuthSchemeFactory() {

            @Override
            public AuthScheme newInstance(final HttpParams httpParams) {
                return new NTLMScheme(new JCIFSEngineOLD());
            }
        });
        httpClient.getCredentialsProvider().setCredentials(new AuthScope(null, -1),
                new NTCredentials(username, password, mac, domain));

        return httpClient;
    }
});
// TODO Auto-generated method stub
return super.getWebClient();
```

Web Driver Integration with Selenium GRID

Why?

1. Multiple OS/Browser Combinations.
2. Throw Hardware for efficiency.
3. Load Tests



GRID

java -jar selenium-server-standalone-2.4.0.jar -role hub

http://weerasin-03:4444/grid/console

java -jar selenium-server-standalone-2.4.0.jar
-role webdriver
-hub http://weerasin-03:4444/grid/register
-Dwebdriver.chrome.driver=C:/workspace/EAAutomation/profiles/chrome/chromedriver.exe
-browser browserName=iexplore,platform=WINDOWS -port 5556

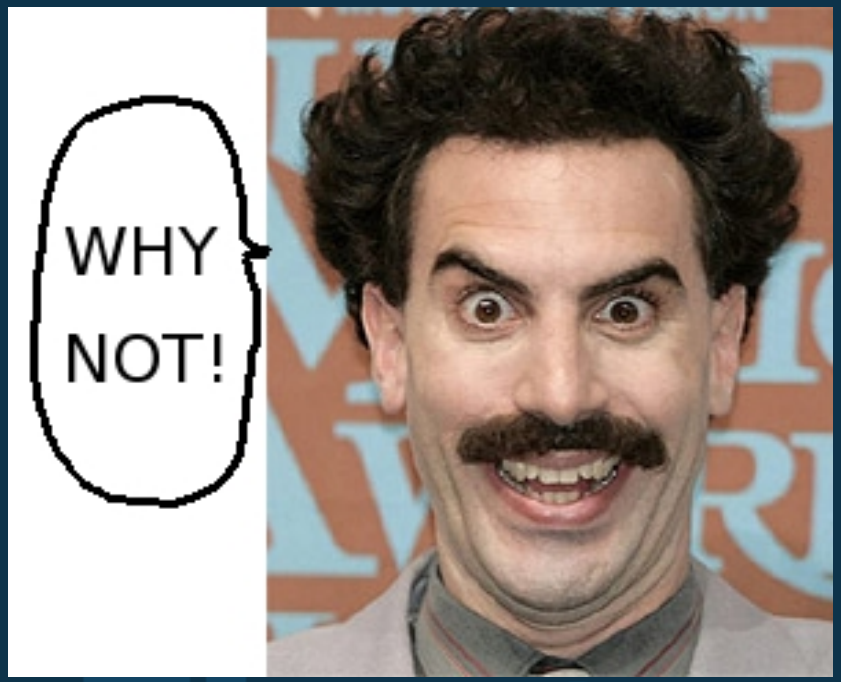
GRID

```
capability = DesiredCapabilities.firefox();
capability.setBrowserName(DesiredCapabilities.firefox().getBrowserName());
capability.setCapability(FirefoxDriver.PROFILE, profile);

capability.setPlatform(org.openqa.selenium.Platform.ANY);
driver = new RemoteWebDriver(new URL(selenium2PropertyReader.get("selenium.grid.url")), capability);
```

Off to the Cloud...

Why?

1. Even more hardware.
2. IAAS for Cost Cutting...



Web Driver Integration with TestNG

- Better support for maintaining test suites & test Execution
  - Grouping of tests
    - A test can fall into multiple (smoke, regression and happy path) groups
    - Dependency between the tests
  - **Parallel execution of tests :)**
    - **can run independent tests in parallel on multiple browsers**
  - Efficient annotation support
    - @BeforeSuite & @AfterSuite, @BeforeTest & @AfterTest, @BeforeGroup & @AfterGroup
  - Exception testing and recover from test failure
- Can easily Build UI based test execution reports on the top of selenium tests (*See screen shots in next slides*)

Web Driver Integration with TestNG

Example:

```
public class AU {
@Test(threadPoolSize = 1, invocationCount = 1,  timeOut = 200000)
public void test_1() throws Throwable {
flow_1("1", "CHROME");
AssertJUnit.assertTrue(true);
}

private void flow_1(String driverId, String Browser) throws Throwable {
        // WRITE A FUNCTIONAL TEST HERE
  }
```

OR edit TestNG.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Suite" parallel="methods" thread-count="5">
  <test name="Test" preserve-order="false">
    <classes>
      <class name="org.finra.spacebook.AU"/>
    </classes>
  </test>
</suite>
```

Web Driver Integration with TestNG

Debug

Create, manage, and run configurations

Project not specified

Configurations:

- Eclipse Application
- Java Applet
- Java Application
- JUnit
- JUnit Plug-in Test
- Remote Java Application
- SWT Application
- TestNG
  - New_configuration

Name: New_configuration

Test | Arguments | Classpath | JRE | Environment

Project

Browse...

Run...

Class Browse...

Groups Browse...

Suite Browse...

Runtime

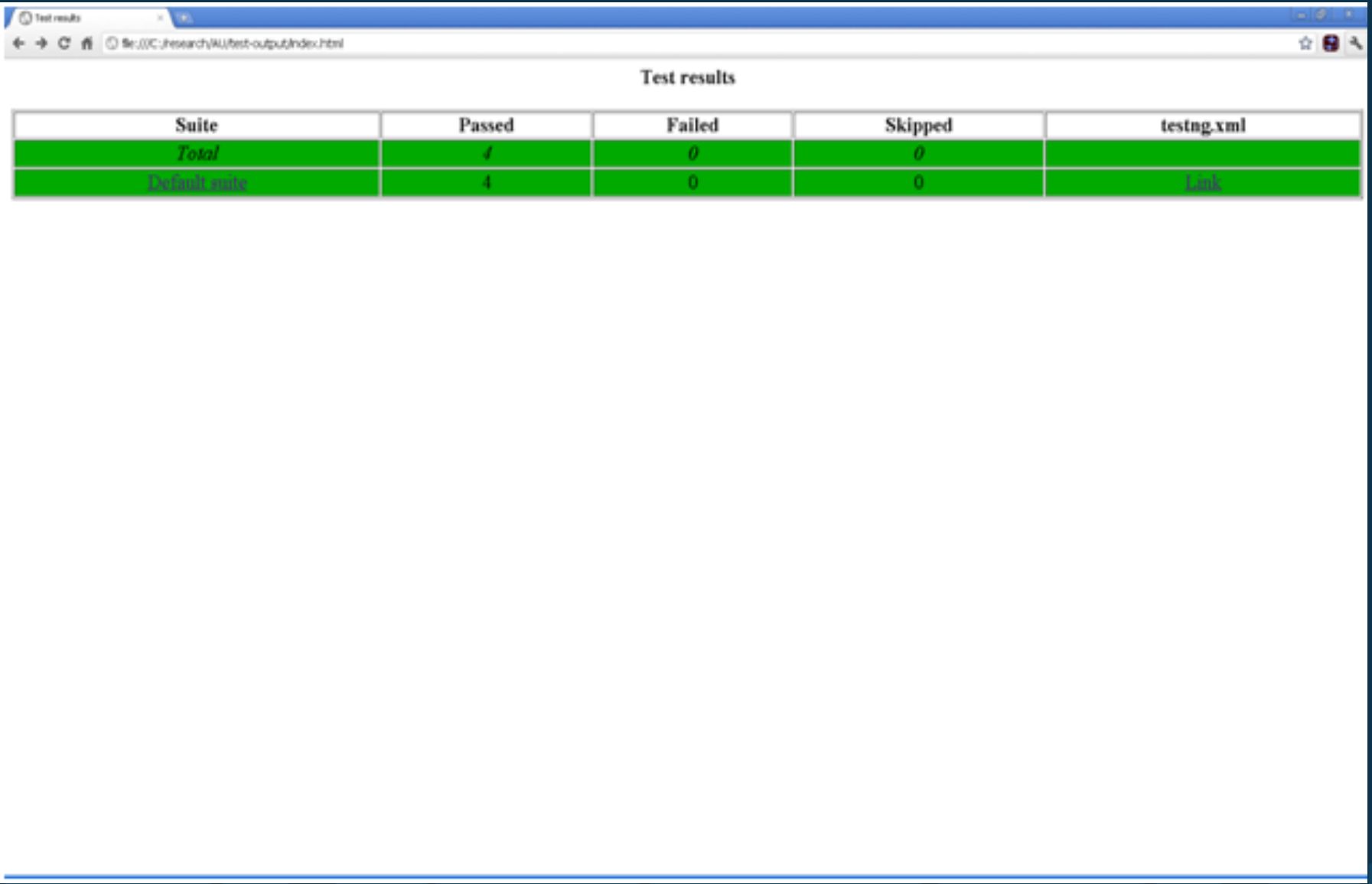
Compliance level (the test sources are needed for JDK 1.4) 1.4

Log level (0-10) 2

New Delete Apply Revert

Debug Close

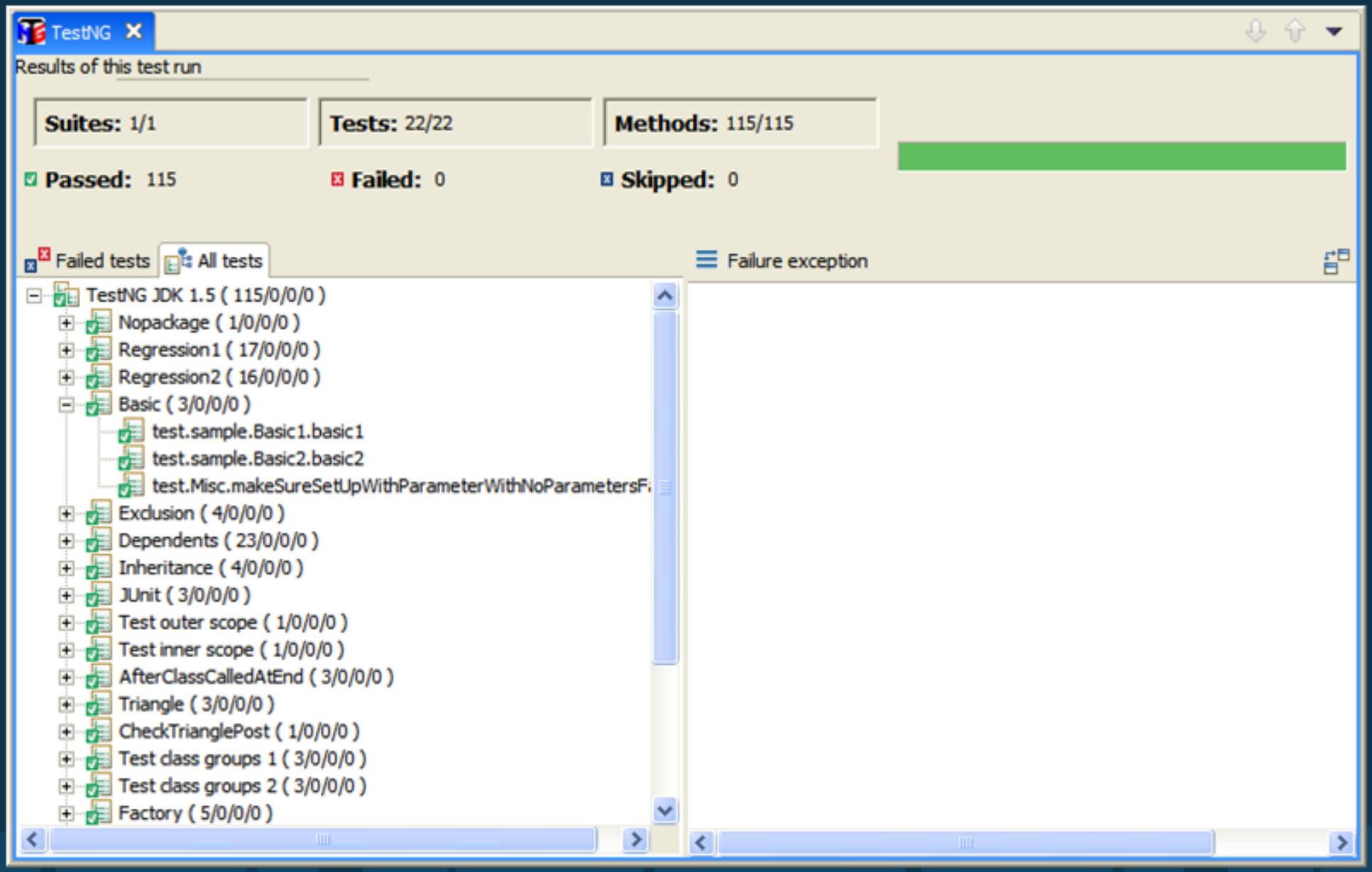
Web Driver Integration with TestNG



Test results

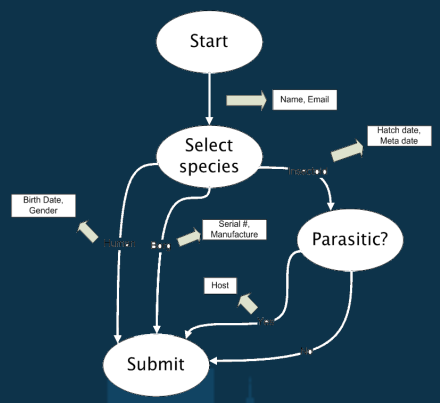
Suite	Passed	Failed	Skipped	testng.xml
Total	*4*	*0*	*0*	
Default suite	4	0	0	Link

Web Driver Integration with TestNG



Test Coverage of Demo App

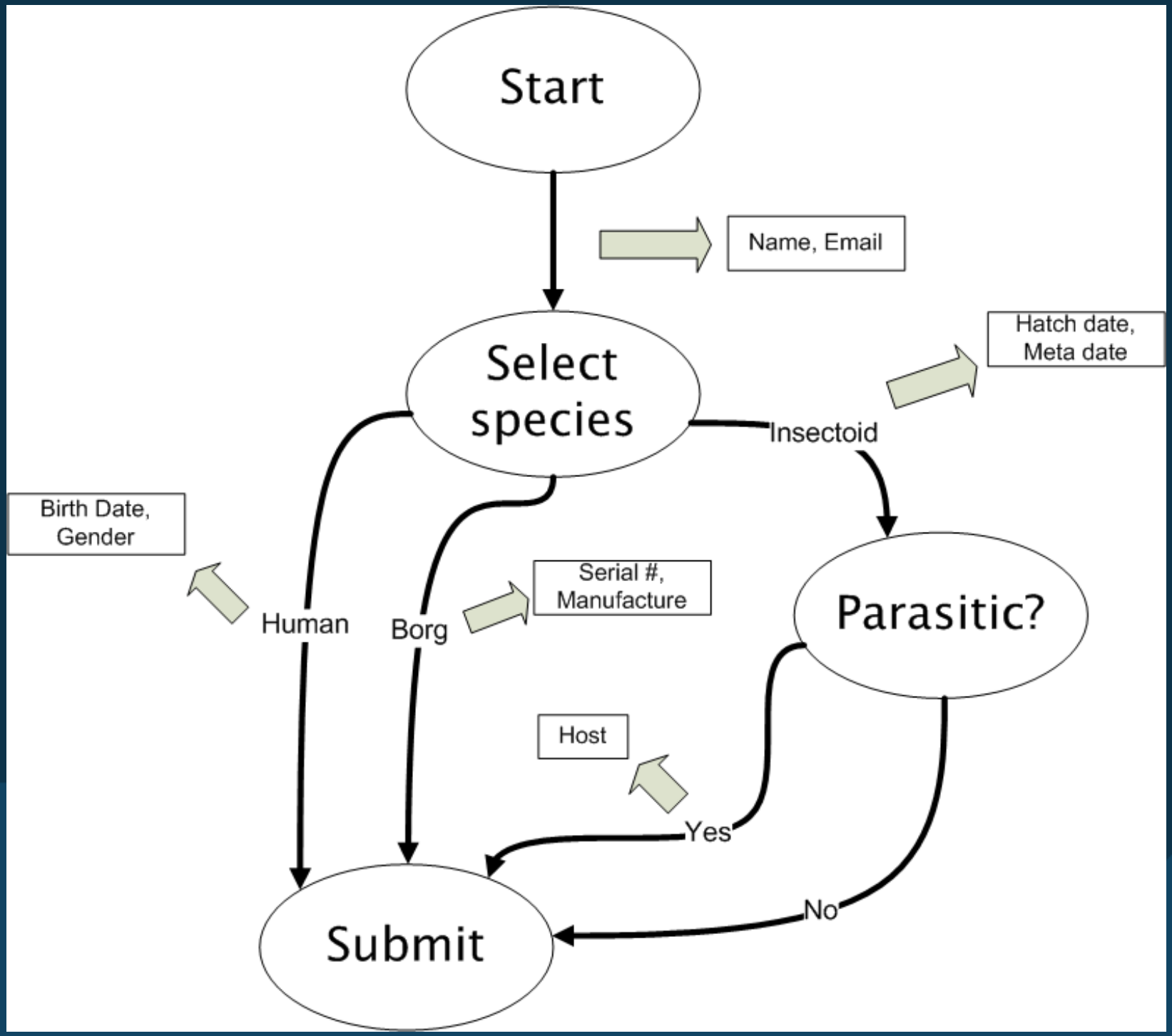
- Dynamic user registration form
- Problem: Test all possible UI flows
- Built Datagenerator application to produce test cases



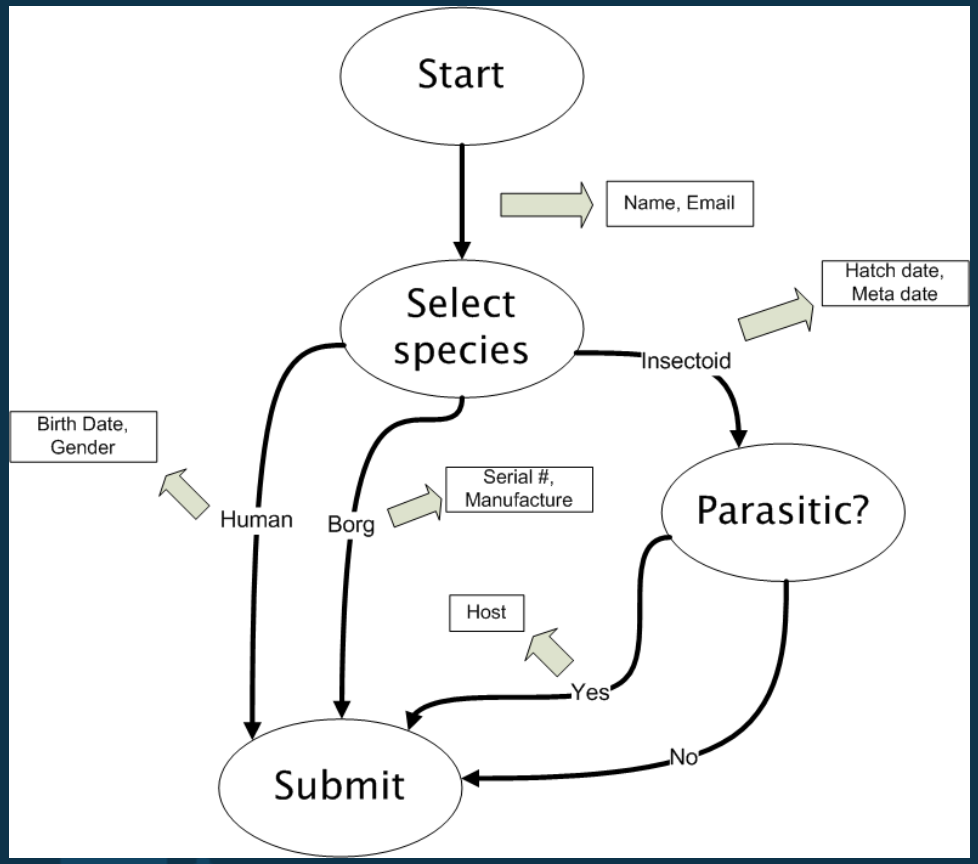
Graph model

Datagenerator

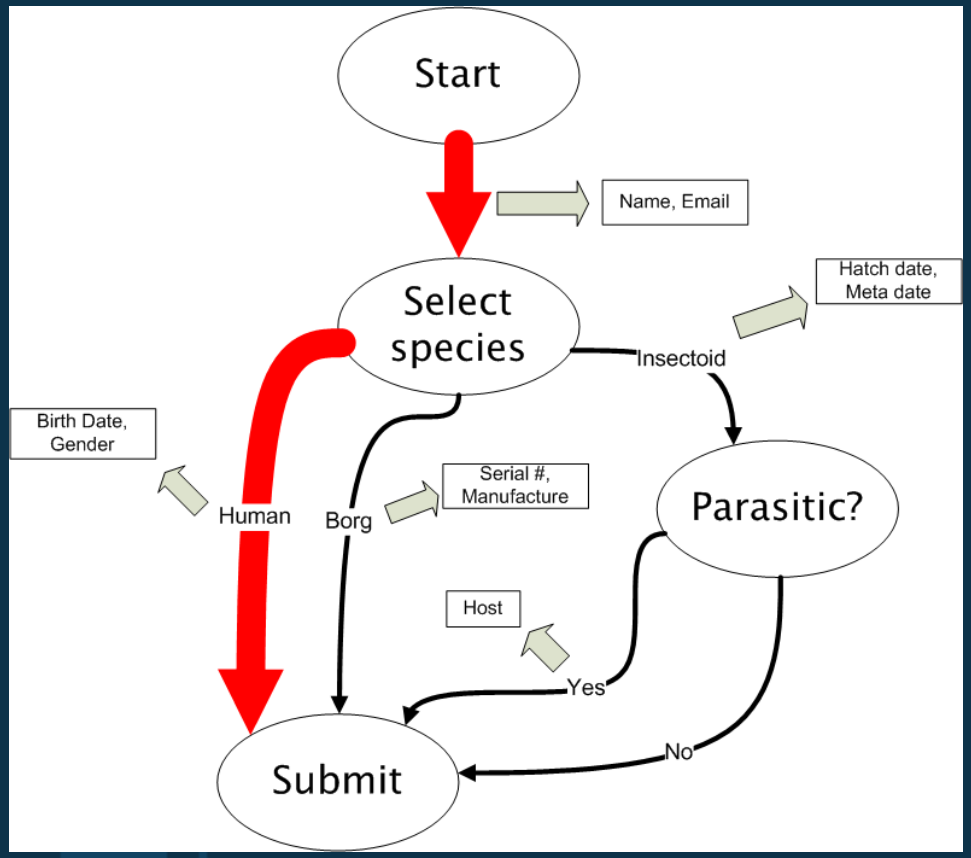
Test Cases


Start
Name, Email
Select species
Hatch date, Meta date
Insectoid
Birth Date, Gender
Human
Borg
Serial #, Manufacture
Parasitic?
Host
Yes
No
Submit

Test Coverage of Demo App

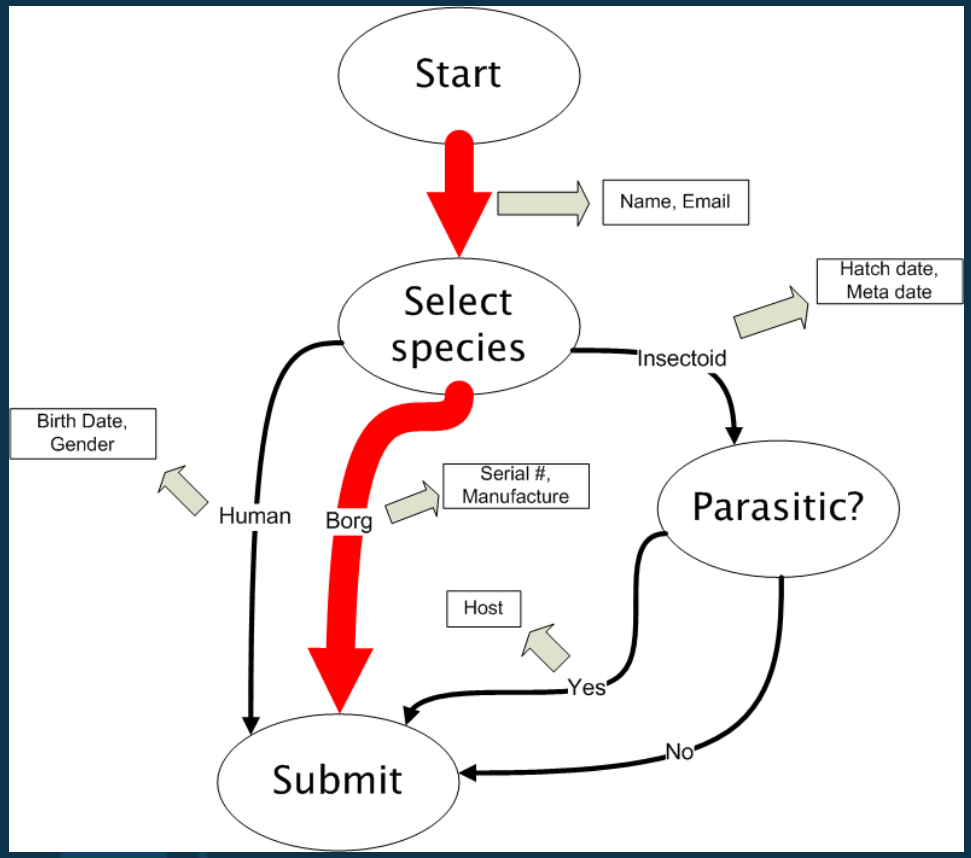


Test Coverage of Demo App



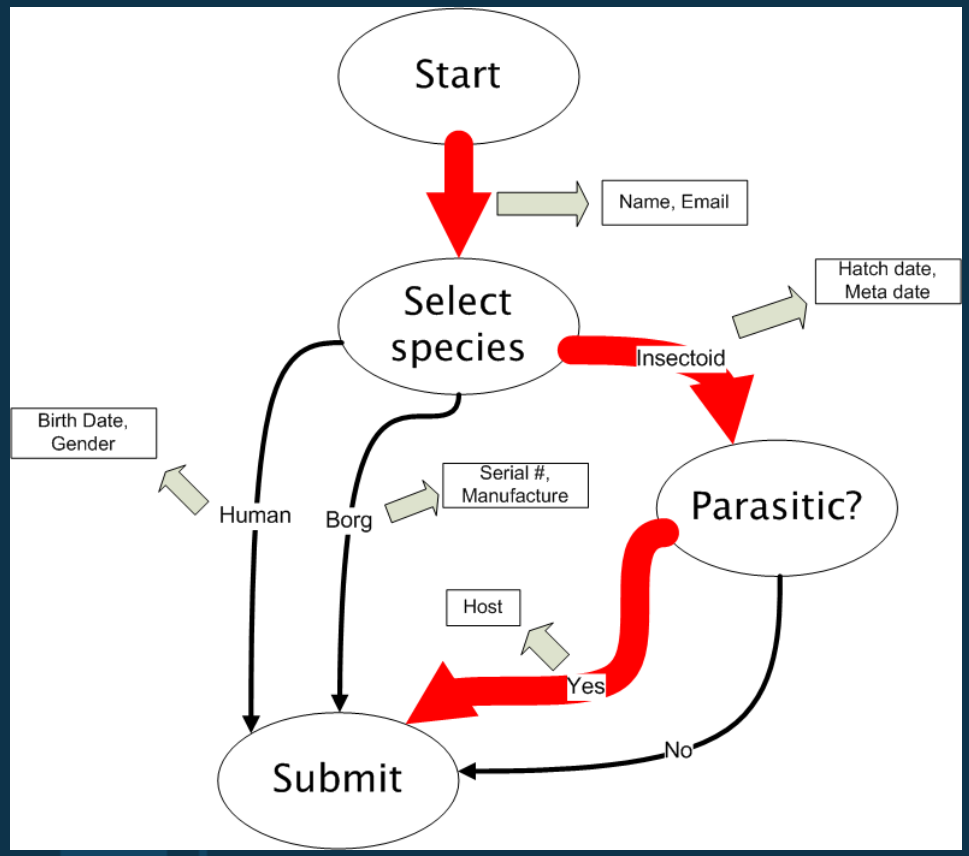
Apache Velocity

Test Coverage of Demo App



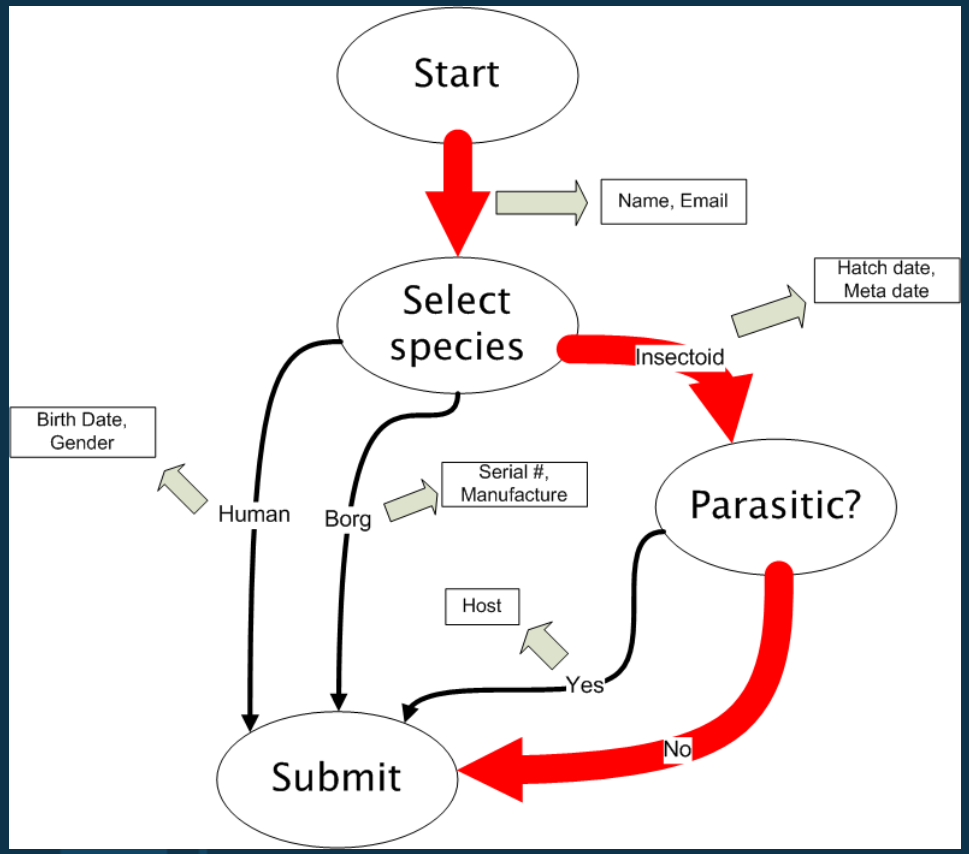
Apache Velocity

Test Coverage of Demo App



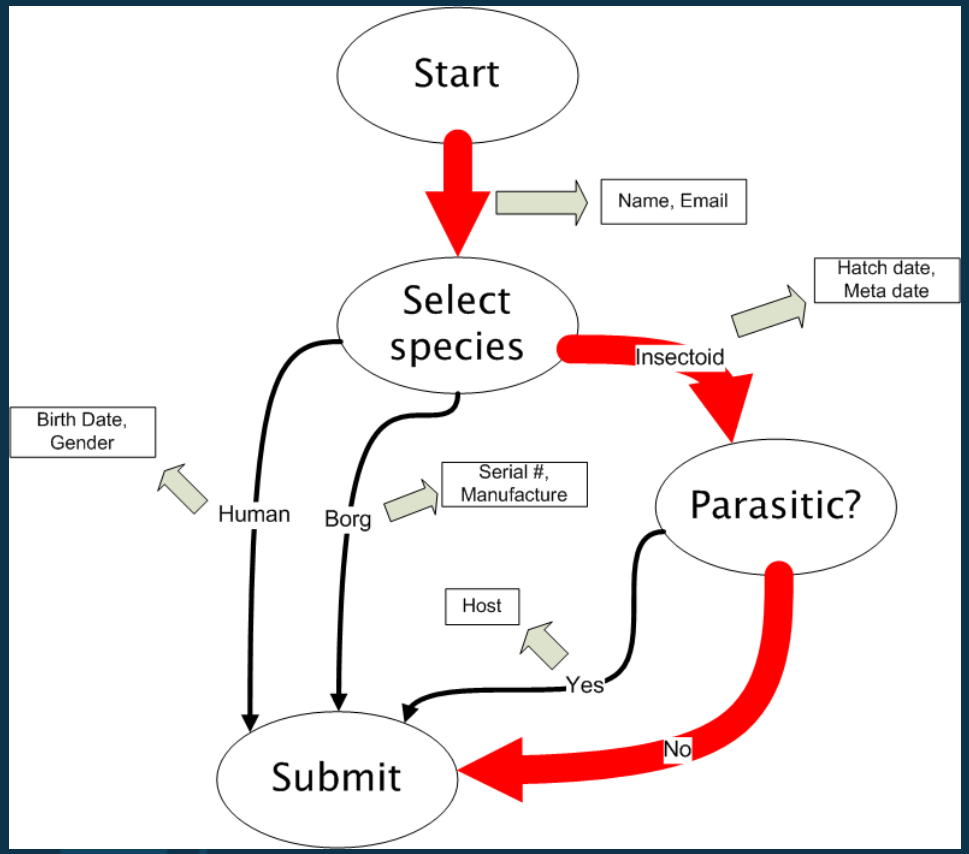
Apache Velocity

Test Coverage of Demo App



Apache Velocity

Test Coverage of Demo App



Apache Velocity

Templatized Java Code

Test Coverage of Demo App

Datagenerator

- Requirements modeled as graph
- Algorithm guarantees all paths are tested

- Other features!
  - Variations of data within a flow
  - Negative scenarios
  - Any text file can be templatized
    - html
    - xml
    - sql
  - Test Data, Test Scripts, Expectations generated in one shot!