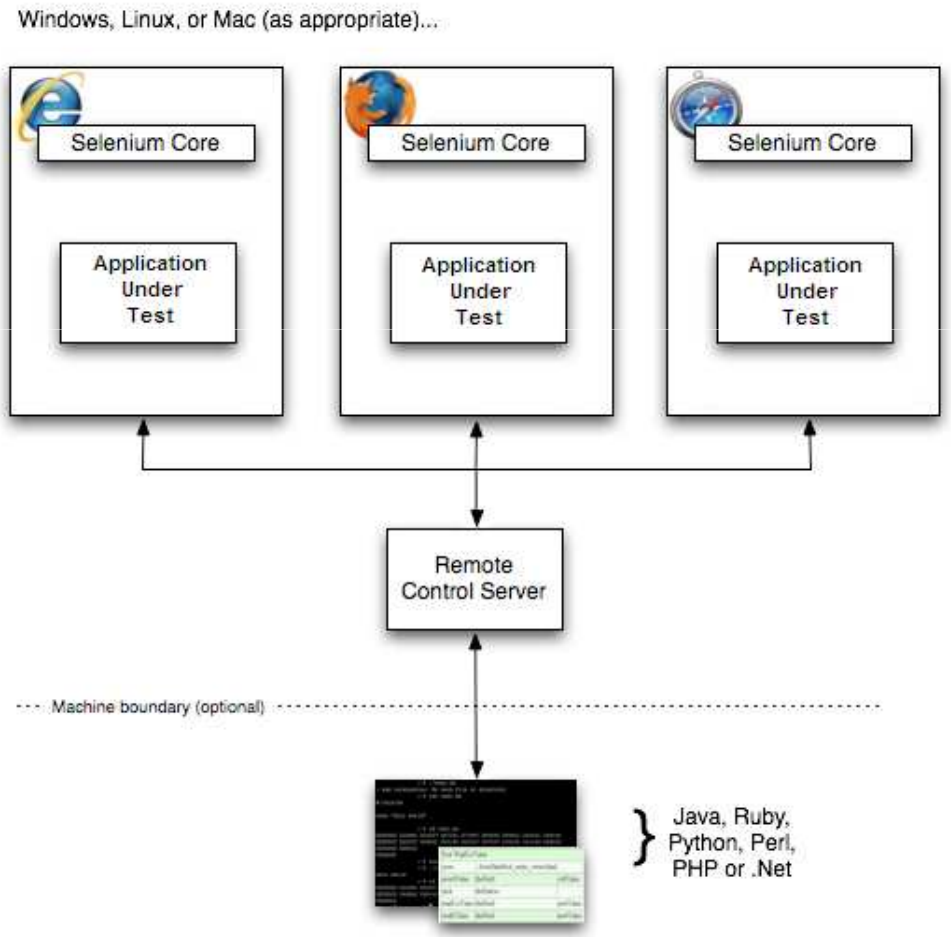
WebDriver/Selenium RC

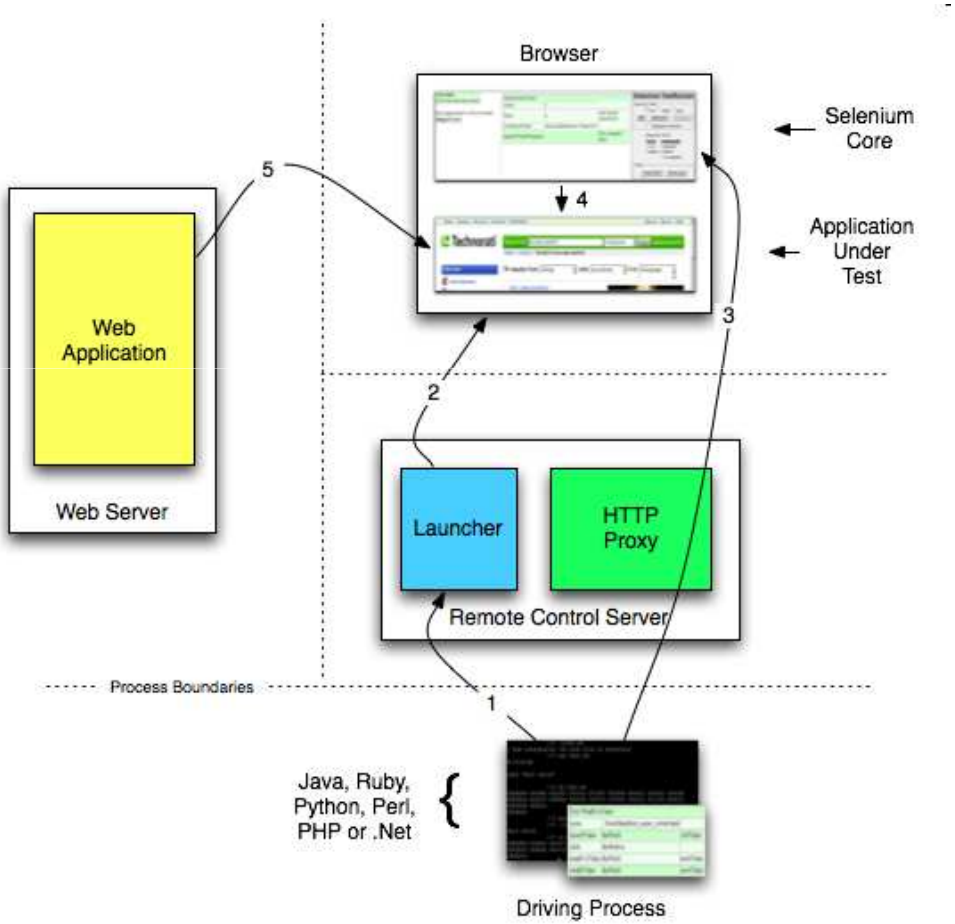
Use full power of Selenium

Selenium RC: Architecture



- Embeds Selenium Core and inject it to browser
- Uses HTTP protocol to communicate with client test code
- Contains code to start different browsers
- Client driver may be written in any language

Selenium RC: Workflow



1. Client/driver connects to selenium server
2. Selenium server launches browser with Selenium Core
3. Client/driver sends command to Selenium Core via HTTP proxy in the Selenium server
4. Selenium Core loads the page inside
5. Web application is asked for page to render

Selenium RC: Browsers

- *firefox – Firefox
- *iexplore – Internet Explorer
- *googlechrome – Google Chrome
- *iehta – Internet Explorer as an HTML Application (HTA) *[deprecated]*
- *chrome – Firefox using a chrome URL *[deprecated]*
- *custom – Custom browser settings
- *pifirefox – Firefox with "proxy injection mode"
- *piiexplore – Internet Explorer with "proxy injection mode“
- *opera – Opera
- *netscape – Netscape
- *konqueror – Konqueror
- *safari – Safari
- *seamonkey – Seamonkey
- *omniweb – Omniweb
- *camino – Camino

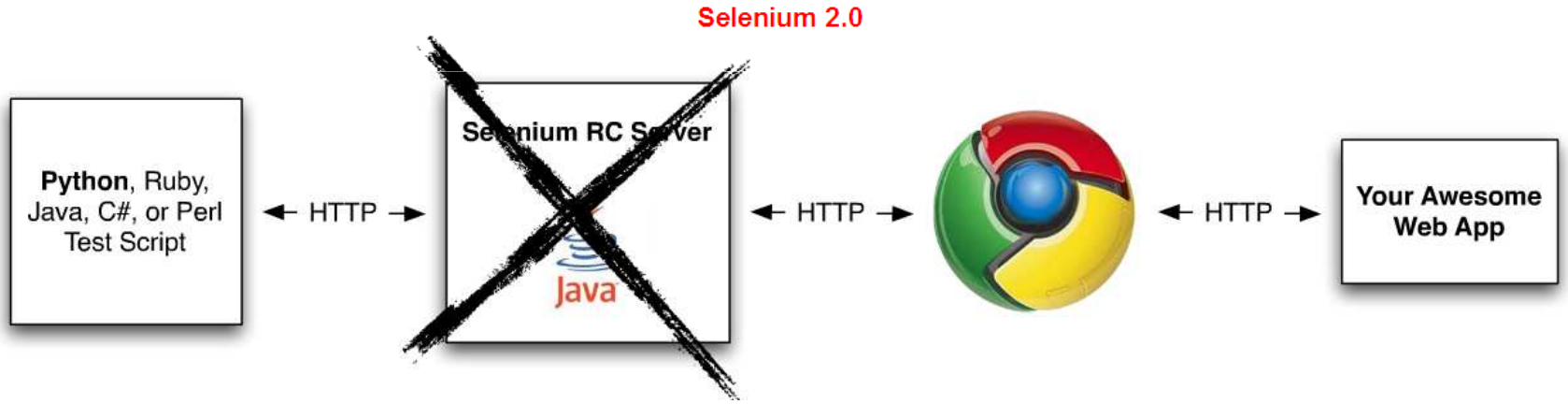
Selenium RC: Common Issues

- Same origin policy (testing some sites together)
- Https support
- Password managers
- Browser dialogs (JavaScript dialogs like alerts and confirmation are fixed)
- Selenium IDE usage

WebDriver: Concepts

- Well-defined API for web testing
- Improved consistency between browsers
- Better emulation of user interactions
- Resolving issues existing in Selenium
- Support by browser vendors
- Additional features
  - Page navigation
  - AJAX-based UI elements
  - Drag-and-drop
  - Windows and frames handling
  - Navigation and cookies

WebDriver: Architecture



WebDriver VS Selenium RC

- Java is not more required
- Old architecture with remote browsers is still possible
- Simplified usage on local machine
- Removed old limitations caused by JavaScript usage (same origin policy, limited access to browser controls, etc.)
- Drivers are developed by different teams
- Still a lot of bugs found
- Frequent releases with fixes and stable roadmap

WebDriver: HtmlUnitDriver

- Fastest implementation of WebDriver
- A pure Java solution, platform independent
- Supports JavaScript (Rhino engine)
- Emulates other browser's JavaScript behavior

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

WebDriver: FirefoxDriver

- Works via native browser manager
- Fast enough
- Based on popular browser

```
DesiredCapabilities capabilities = DesiredCapabilities.firefox();
capabilities.setJavascriptEnabled(true);
WebDriver driver = new FirefoxDriver(capabilities);
```

WebDriver: FirefoxDriver

- Configurable
  - webdriver.firefox.bin
  - webdriver.firefox.profile
  - many other properties

```
File firebug = new File("firebug-1.9.2.xpi");
File netExport = new File("netExport-0.8.xpi");

profile.addExtension(firebug);
profile.addExtension(netExport);
```

WebDriver: InternetExplorerDriver

- Works via native browser manager
- Require standalone IEDriverServer
- Slower than other popular browsers
- Works only on Windows ;)

```
DesiredCapabilities capabilities = DesiredCapabilities.internetExplorer();
capabilities.setJavascriptEnabled(true);
WebDriver driver = new InternetExplorerDriver(capabilities);
```

WebDriver: InternetExplorerDriver

- Requires IE manager downloading (http://selenium2.ru/articles/52-how-to-start-internet-explorer.html)
- Configuration requires some magic steps
  - Set Protected Mode for each zone to same value
  - Set browser zoom level to 100% to enable native mouse events

WebDriver: RemoteWebDriver

- Separates where test and browser location
- Allow to run tests on unsupported browsers
- Introduces extra latency to tests
- Requires an external servlet container to be running
  - *java -jar selenium-server-standalone-{VERSION}.jar*

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setJavascriptEnabled(true);
URI serverPort = URI.create("http://localhost:4444");
WebDriver driver = new RemoteWebDriver(serverPort.toURL(), capabilities);
```

WebDriver: Other Drivers

- OperaDriver
- AndroidDriver
- IPhoneDriver
- Any browser via RemoteWebDriver
- ChromeDriver
- BlackberryDriver
- WebKitDriver
- SafariDriver

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setBrowserName("safari");
URL seleniumRC = URI.create("http:localhost:4444/").toURL();
URL testedURL = URI.create("http://www.google.com/").toURL();
CommandExecutor executor = new SeleneseCommandExecutor(seleniumRC, testedURL, capabilities);
WebDriver driver = new RemoteWebDriver(executor, capabilities);
```

WebDriver: Basics

- Navigate to page
  - *driver.get("http://www.google.com");*
  - *driver.navigate().to("http://www.google.com");*
- Find elements on page
  - *driver.findElement(By.name("passwd"));*
- Interact with elements
  - *driver.findElement(By.id("submit")).click();*
  - *driver.findElement(By.id("passwd-id")).sendKeys("some text");*

WebDriver: Advanced Features

- Navigate to window
  - *driver.switchTo().window("windowName");*
- Navigate to frame
  - *driver.switchTo().frame("frameName");*
- Navigate via history
  - *driver.navigate().forward();*
  - *driver.navigate().back();*
- Drag and Drop

```
WebElement element = driver.findElement(By.name("source"));
WebElement target = driver.findElement(By.name("target"));
(new Actions(driver)).dragAndDrop(element, target).perform();
```

WebDriver: Complex Actions

- Manual Drag and Drop

```
Action dragAndDrop = new Actions(driver)
        .clickAndHold(element)
        .moveToElement(target)
        .release(element)
        .build();
dragAndDrop.perform();
```

- Complex mouse/keyboard interactions

```
Action complexClick = new Actions(driver).keyDown(Keys.CONTROL)
        .click(element)
        .doubleClick(element)
        .keyUp(Keys.CONTROL).build();
complexClick.perform();
```

WebDriver: JavaScript Code

- Use it only if really no other way found!

```
FirefoxDriver driver = new FirefoxDriver();
driver.executeScript("your code here", 5);
driver.executeAsyncScript("your async code", 3, "callback");
```

- Use return to get value from JavaScript code
- Alerts and prompts can be handled

```
Alert alert = driver.switchTo().alert();
alert.getText();
alert.accept();
```

WebDriver: Warnings

- WebElement has all possible methods independent on context and element type
- JavaScript is disabled in HtmlUnitDriver by default
- Disable native events on Firefox profile if it make difference for you
- Only Firefox accepts untrusted SSL certificates
- XPath is emulated in some drivers, so not always case insensitive

Web Driver: Code Sample

```
@Test
public void userNameSuggestions() {
    WebDriver driver = new FirefoxDriver();

    driver.get(APP_URL + "/bank");

    driver.findElement(By.linkText("Manage Accounts")).click();
    driver.findElement(By.id("user")).sendKeys("j");

    new WebDriverWait(driver, 30).until(new Predicate<WebDriver>() {
        public boolean apply(WebDriver webDriver) {
            return webDriver.findElement(By.cssSelector(".ac_results")).isDisplayed();
        }
    });
    driver.findElement(By.id("user")).sendKeys(Keys.ENTER);

    Assert.assertEquals("John", driver.findElement(By.id("user")).getAttribute("value"));
}
```

WebDriver: Bot Style Tests

```
public class ActionBot {
    private final WebDriver driver;

    public ActionBot(WebDriver driver) {
        this.driver = driver;
    }

    public void click(By locator) {
        driver.findElement(locator).click();
    }

    public void submit(By locator) {
        driver.findElement(locator).submit();
    }

    public void type(By locator, String text) {
        WebElement element = driver.findElement(locator);
        element.clear();
        element.sendKeys(text + "\n");
    }
}
```

WebDriver: Page Object Pattern

- LoadableComponent make page selfmanaged
- PageFactory helps to init page components

```
public class AccountsManagementPage extends LoadableComponent<AccountsManagementPage> {
    private final WebDriver driver;

    //will be located automatically
    @CacheLookup
    private WebElement user;

    @FindBy(name = "amount")
    @CacheLookup
    private WebElement amount;

    public AccountsManagementPage(WebDriver driver) {
        this.driver = driver;
        PageFactory.initElements(driver, this);
    }

    @Override
    protected void load() {
        driver.get("http://localhost:8080/bank/manageAccounts");
    }

    @Override
    protected void isLoaded() throws Error {
        String url = driver.getCurrentUrl();
        assertTrue(url.endsWith("/manageAccounts"));
    }
}
```

WebDriver: “Ideal” Page Object

```
public class AccountsPage extends AbstractPage {
    //wrapped automatically by name or id
    private Form accountForm;

    @FindBy(css = "table")
    private Table accounts;

    public AccountsPage(WebDriver driver) {
        super(driver);
        PageFactory.initElements(driver, AccountsPage.class);
    }

    public AbstractPage open() {
        open("/manageAccounts");
        return this;
    }

    public AbstractPage registerAccount(String name, double amount) {
        accountForm.set("user", name)
                .set("amount", "" + amount)
                .submit();
        return new AccountsPage(driver);
    }

    public int getRegisteredUsersCount() {
        return accounts.getRowCount();
    }
}
```

WebDriver: Driver Configuration

- Configuring custom user agent

```
FirefoxProfile profile = new FirefoxProfile();
profile.setPreference("general.useragent.override", "GoogleBot");
WebDriver driver = new FirefoxDriver(profile);
```

- Configuring custom Firefox profile

```
ProfilesIni allProfiles = new ProfilesIni();
FirefoxProfile profile = allProfiles.getProfile("WebDriver");
WebDriver driver = new FirefoxDriver(profile);
```

```
String profilePath = "C:\\Users\\lumii.KIEV\\AppData\\Roaming\\" +
        "Mozilla\\Firefox\\Profiles\\a24yah8x.default";
FirefoxProfile profile = new FirefoxProfile(new File(profilePath));
WebDriver driver = new FirefoxDriver(profile);
```

WebDriver: Monitoring

- Taking screenshots

```
File screenshot = driver.getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(screenshot, new File("c:\\tmp\\screenshot.png"));
```

- Journaling and logging
  - By xUnit or TestNG framework
  - By EventFiringWebDriver wrapper

```
new EventFiringWebDriver(driver).register(new AbstractWebDriverEventListener() {
    @Override
    public void afterClickOn(WebElement element, WebDriver driver) {
        LOG.log(Level.INFO, "Click on element " + element.getTagName());
    }
});
```

WebDriver: Selenium Emulation

- Allows WebDriver and Selenium tests to live together
- Helps with tests migration
- Doesn’t require Selenium RC to be run
- Doesn’t emulate every method, some are also slower

```
WebDriver driver = new FirefoxDriver();
Selenium selenium = new WebDriverBackedSelenium(driver, "http://www.google.com");
selenium.open("http://www.google.com");
selenium.type("name=q", "cheese");
selenium.click("name=btnG");
WebDriver driverInstance = ((WebDriverBackedSelenium) selenium).getWrappedDriver();
driver.close();
```

WebDriver: Migration Notes

- *type*, *keyDown*, *keyPress*, *keyUp* was replaced with *sendKeys*
- *waitForPageToLoad* doesn't really wait for page to be loaded
  - Use *WebDriverWait* class or specify default wait timeout
  - *driver.manage().timeouts().implicitlyWait(**30**, TimeUnit.SECONDS);*
- CSS and XPath works differently because they are emulated for some browsers
- JavaScript code is not more used for WebDriver core, so some objects are not available

WebDriver: TestNG Benefits

- Support parameters and data providers
- Powerful execution model
- Test groups
- Dependent test methods
- Parallel tests execution and timeouts
- Rerunning failed tests
- Dependency injection
- Invoked method listeners
- Test factories

WebDriver: Demo

- Test creation
- Test refactoring
- Running tests with JUnit and TestNG
- Running tests in different browsers
- Tune browser configuration

WebDriver: Best Practices

- Create base tests with all driver details
- Use helpers or tests hierarchy
- Utilize Page Object and other patterns implemented by WebDriver
- Select one language for writing tests
- Use Selenium IDE for tests creation
- Reuse code
- Give meaningful names to combined operations

Questions & Answers