

Tellurium Automated Web Testing Framework

Jian Fang

(John.Jian.Fang@gmail.com)

Vivek Mongolu

(vivekmongolu@gmail.com)

Who are these guys?

■ Jian Fang

- Senior Software engineer @ Jewelry Television
- Enterprise Applications focusing on the server side
- **Tellurium Project Lead**

■ Vivek Mongolu

- Software engineer @ Jewelry Television
- Enterprise Applications focusing on the server side and client side
- **Tellurium Project Member**

Outline

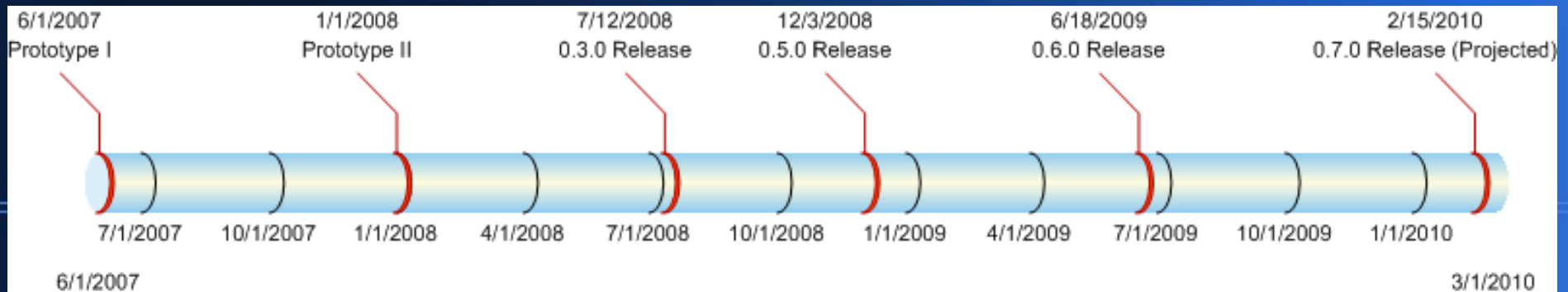
- **Overview**
- Problems in Selenium
- Tellurium in Action
- Tellurium's solution
- Tellurium concepts
- Architecture
- Tellurium UI Model Plugin(Trump)
- Projects
- 0.7.0 Feature Preview
- Resources

What is Tellurium?

- Tellurium is a UI module based web testing framework built on top of Selenium (currently building our own testing engine).
- Open source. <http://code.google.com/p/aost/>
- Tellurium is implemented in Groovy and Java
- Tellurium includes a set of Domain Specific Languages (DSLs) to define UI, actions, and tests
- UI is defined in Groovy class and tests can be written in
 - Java (JUnit 4 or TestNG)
 - Groovy
 - Pure DSL script

Tellurium History

- 2007-06-01, first prototype
- 2008-01-01, second prototype
- 2008-07-12, release 0.3.0
- 2008-08-13, release 0.4.0
- 2008-12-03, release 0.5.0
- 2009-06-18, release 0.6.0
- 2010-02-15, release 0.7.0 (projected)



Tellurium Main Features

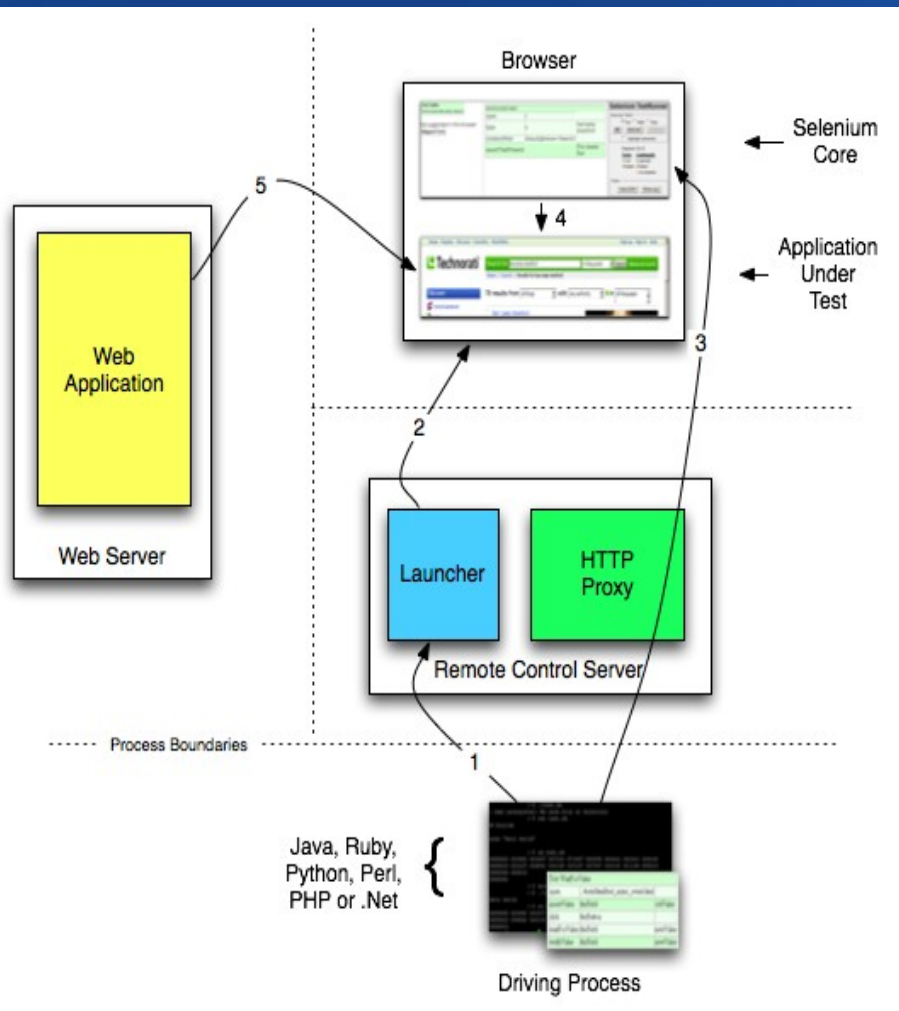
- **UI module** for structured test code and re-usability
- **Composite Locator** to use a set of attributes to describe a UI element
- **Group locating** to exploit information inside a collection of UI components
- **UI templates** for dynamic web content
- **JQuery selector** support to improve test speed in IE
- **Data driven testing** support
- **Selenium Grid** support
- **JUnit** and **TestNG** support
- **Maven** support

Outline

- Overview
- ***Problems in Selenium***
- Tellurium in Action
- Tellurium's solution
- Tellurium concepts
- Architecture
- Tellurium UI Model Plugin(Trump)
- Projects
- 0.7.0 Feature Preview
- Resources

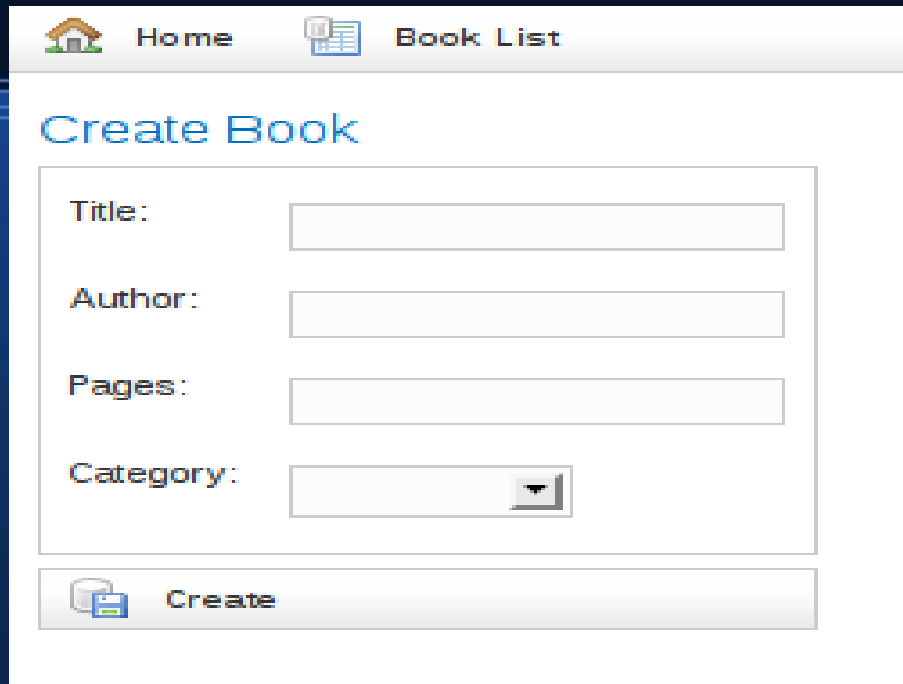
Selenium

- Selenium is a suite of tools to automate web app testing across many platforms.
 - Selenium Core: DHTML test execution framework
 - Selenium IDE: Firefox extension – record and playback
 - Selenium Remote Control
 - Server: launches/kills browser, HTTP Proxy
 - Client libraries
 - Selenium Grid
 - Run Selenium tests in parallel



1. The client/driver reaches out to the selenium-server.
2. Selenium-Server launches a browser with a URL that will load Selenium core web page.
3. Selenium-Core gets the first instruction from the client/driver (via the HTTP Proxy built into the Selenium RC Server).
4. Selenium-Core acts on that first instruction, typically opening a page of the AUT.
5. Web server is asked for that page, and it renders in the frame/window reserved for it.

What's wrong with Selenium



The screenshot shows a web browser window with two tabs: 'Home' and 'Book List'. The 'Book List' tab is active. The page title is 'Create Book'. The form contains four input fields: 'Title:', 'Author:', 'Pages:', and 'Category:'. The 'Category:' field is a dropdown menu. Below the form is a 'Create' button with a book icon.

- Simple BookStore App – Create Book
 - The above UI includes Create Book module with three text boxes, one select box and one button
 - Use Selenium IDE and it generates the following Selenium test case

```
package com.selenium.bookapp.test;

public class SeleniumCreateBookTestCase extends SeleneseTestCase {
    public void setUp() throws Exception {
        setUp("http://localhost:8080/BooksApp/", "*chrome");
    }

    public void testNew() throws Exception {
        selenium.open("book/create");
        selenium.type("title", "Tellurium");
        selenium.type("author", "Jian and Vivek");
        selenium.type("pages", "100");
        selenium.select("category", "label=Technical");
        selenium.click("//input[@value='Create']");
        selenium.waitForPageToLoad("30000");

        Assert.assertEquals(1,
            selenium.getXpathCount("/html/body/div[4]/div[1]"));
    }
}
```

Selenium RC Demo

What do you observe?

■ Verbose

- **selenium** everywhere
- Locators everywhere

■ Coupling

- UI locators, actions, and assertions

are coupled

■ Not expressive

- What does the UI look like?
- What UI element the locator is associated with?

■ Fragile

- Look at the **message** xpath

■ Record and Play

- Need to cover all testing scenarios
- How about data dependency?

■ Refactor

- Seems to be difficult.

■ Reusable

- Less likely

■ Dynamic factors on the web

- Dynamic data such as data grid and list of options and Javascript events

```
selenium.open("/book/create");
selenium.type("title", "Tellurium");
selenium.type("author", "Jian and
Vivek");
selenium.type("pages", "100");
selenium.select("category",
"label=Technical");

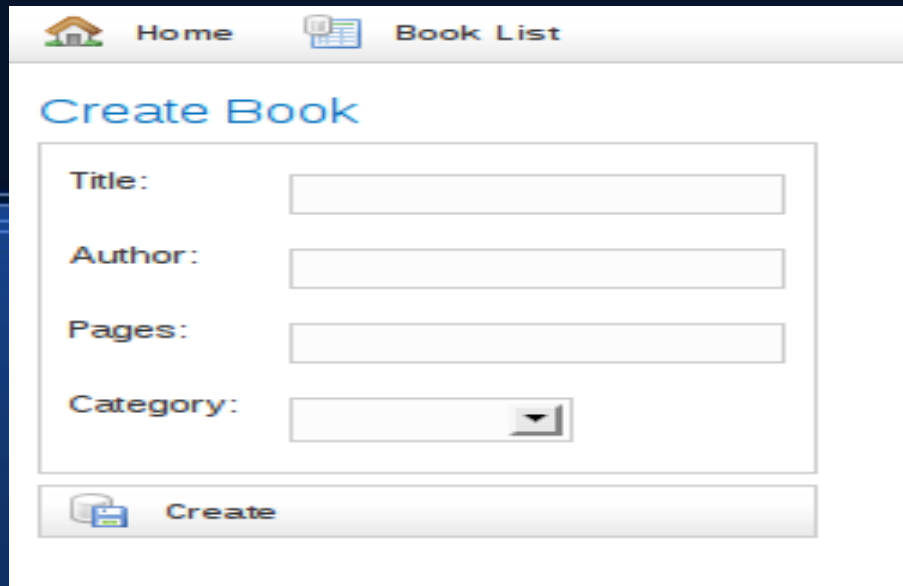
selenium.click("//input[@value='Create']"
);
selenium.waitForPageToLoad("30000");

//Assert the message count
Assert.assertEquals(1,
selenium.getXpathCount("/html/body/div[4]
/div[1]"));
```

Outline

- Overview
- Problems in Selenium
- ***Tellurium in Action***
- Tellurium's solution
- Tellurium concepts
- Architecture
- Tellurium UI Model Plugin(Trump)
- Projects
- 0.7.0 Feature Preview
- Resources

What Tellurium looks like?



The screenshot shows a web browser window with a navigation bar containing 'Home' and 'Book List' links. The main content area is titled 'Create Book' and contains a form with the following fields:

- Title:
- Author:
- Pages:
- Category:

At the bottom of the form is a 'Create' button with a document icon.

```
ui.Form(uid: "CreateForm", clocator: [tag: "form", method: "post",  
                                     action: "/BookApp/book/save"]){  
  Container(uid: "CreateTable", clocator: [tag: "table"]){  
    InputBox(uid: "TitleInput", clocator: [tag: "input", type: "text", name: "title",  
                                           id: "title"])  
    InputBox(uid: "Author", clocator: [tag: "input", type: "text", name: "author",  
                                       id: "author"])  
    InputBox(uid: "PagesInput", clocator: [tag: "input", type: "text", name: "pages",  
                                           id: "pages"])  
    Selector(uid: "CategoryInput", clocator: [tag: "select", id: "category", name:  
                                             "category"])  
  }  
  SubmitButton(uid: "CreateButton", clocator: [tag: "input", type: "submit",  
                                              value:"Create", class: "save"])  
}  
  
ui.Container(uid: "ShowBook", clocator: [:]){  
  Div(uid: "Message", clocator: [tag: "div", class: "message"])  
}
```

Methods defined for UI modules

```
public void createBook(String author, String pages, String title) {
    keyType "CreateForm.CreateTable.Author", author
    keyType "CreateForm.CreateTable.PagesInput", pages
    keyType "CreateForm.CreateTable.TitleInput", title
    selectByLabel "CreateForm.CreateTable.CategoryInput", category
    click "CreateForm.CreateButton"
    waitForPageToLoad 30000
}

public String getMessage(){
    getText "ShowBook.Message"
}
```

Tellurium Create Book Demo

Outline

- Overview
- Problems in Selenium
- Tellurium in Action
- ***Tellurium's solution***
- Tellurium concepts
- Architecture
- Tellurium UI Model Plugin(Trump)
- Projects
- 0.7.0 Feature Preview
- Resources

How challenges are addressed in Tellurium

- No longer “record and play” style, but UI module oriented
 - defining nested UI objects and methods as a UI module class, write your own test cases based on the UI module class.
- Decoupling
 - **UI and tests are separated.**
 - UI is defined in a Groovy UI module class and test cases in Java/Groovy/DSL script.
- Robust
 - Composite locator (UI element attributes) is used to define UI and the actual xpath will be generated at runtime, i.e., different attributes -> different runtime xpath. **UI module internal changes are addressed.**
 - Group locating is used to remove the dependency of the UI objects from external UI elements, i.e., **external UI changes will not affect the current UI module for *most* cases.**
- Easy to refactor
 - Refactor the individual UI module class

How challenges are addressed in Tellurium

- Reusable
 - UI module class could be re-used for the same application
 - Tellurium widgets can be re-used for different applications.
- Expressive
 - UI module, Groovy syntax and DSL for actions and tests.
- Dynamic factors on Web
 - UI templates for data grid and dynamic data lists
 - “respond” attribute in UI object for JavaScript events

How challenges are addressed in Tellurium

- Easy to use
 - Tellurium provides a configuration file for you to wire in your own new UI objects/widgets and customize the framework
 - Test can be written in Java/Groovy/DSL Script
 - Support JUnit 4 and TestNG
 - Reference projects are provided to guide users on how to use Tellurium
 - Supports ant and Maven build
- Data Driven Testing

Tellurium Validation Demo

Outline

- Overview
- Problems in Selenium
- Tellurium in Action
- Tellurium's solution
- ***Tellurium concepts***
- Architecture
- Tellurium UI Model Plugin(Trump)
- Projects
- 0.7.0 Feature Preview
- Resources

Why Tellurium is A NEW Approach?

- Tellurium comes with a set of new concepts that make Tellurium a different and new approach for Web Testing
 - UI Object
 - UI Module
 - UID
 - Composite Locator
 - Group Locating
 - UI Template
 - Widgets
 - DslContext

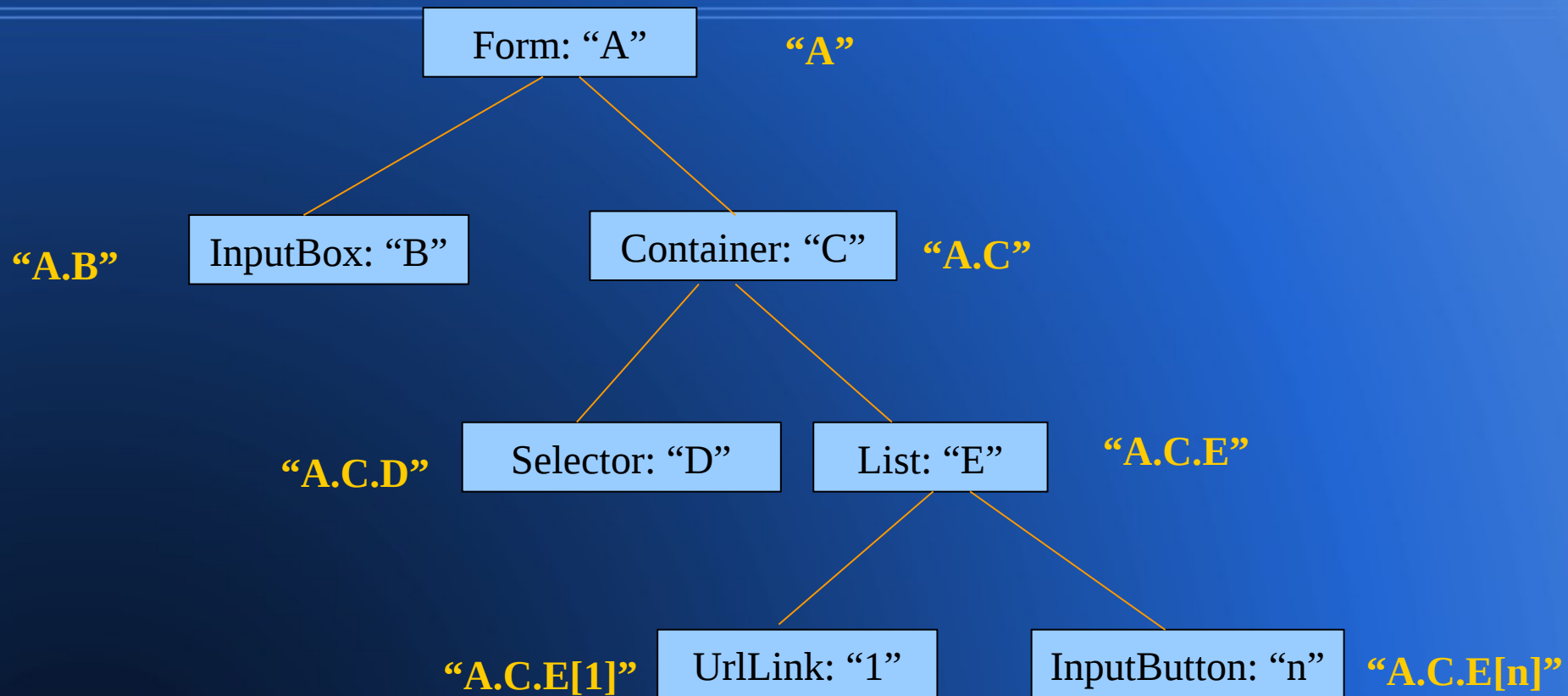
New Concepts: UI Object

- UI Object includes
 - *uid*: identifier.
 - *namespace*: XHTML/XForm
 - *locator*: the locator of the UI object, could be
 - A base locator (relative xpath)
 - A composite locator (object attributes)
 - *group*: flag for group locating.
 - *respond*: JavaScript events the UI object should respond to.
- Simple UI Objects: Button, CheckBox, Selector, ...
- Container: List, Table, Frame, ...

New Concepts: UI Module

- UI module is a collection of UI elements you group them together
- UI module represents a composite UI object in the format of nested basic UI elements.
- UI elements inside the composite UI object have relationship to each other, i.e., logically or to be on the same subtree on the DOM
- For each UI module, define set of methods to act on the UI.

New Concepts: UID



- Each UI object has a UID and is referenced by cascading UIDs
- Elements in List and Table are referenced by indexes
- UID makes Tellurium less verbose and more expressive

New Concepts: Composite Locator

- Composite Locator (*clocator*): A set of attributes used to define the object and the actual locator will be derived automatically at run time. For example,

```
SubmitButton(uid: "CreateButton", clocator: [tag: "input", type: "submit", value: "Create", class: "save"])
```

which is equal to the following runtime xpath:

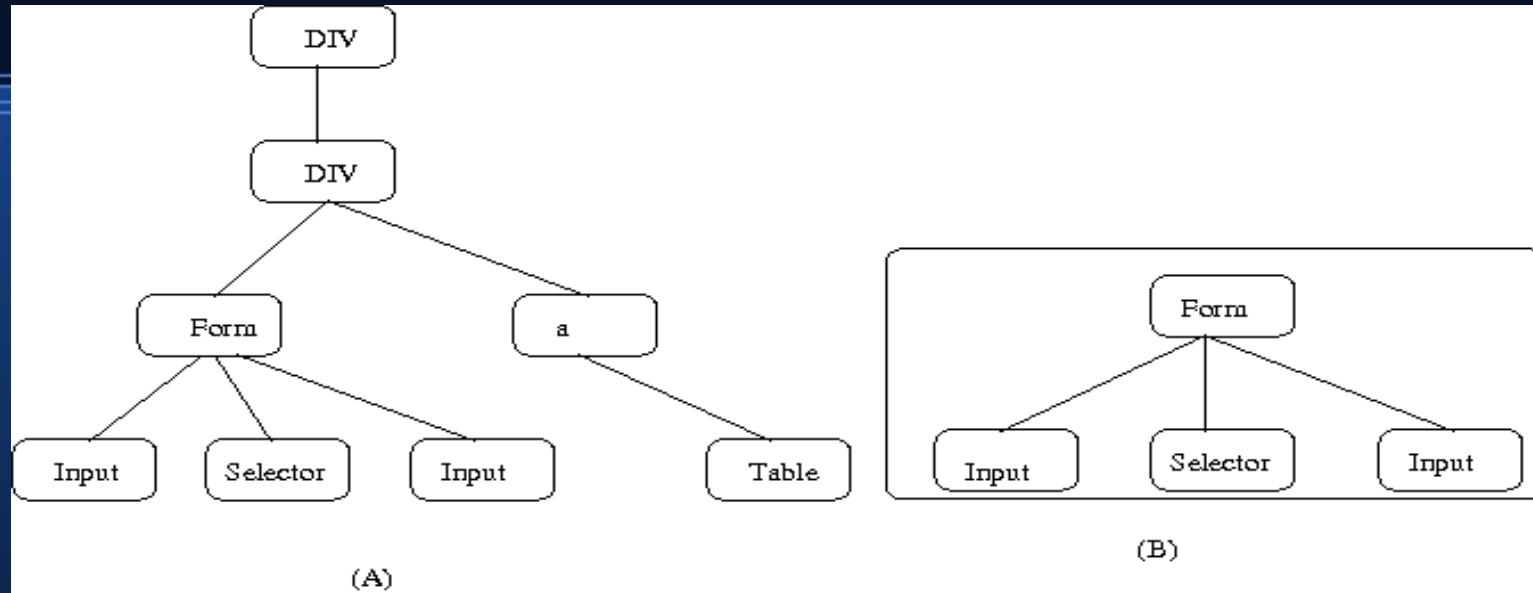
```
//descendant-or-self::form[@method="post" and @action="/BooksApp/book/save"]/descendant-or-self::input[@type="submit" and @value="Create" and @class="save"]
```

- The class definition is as follows,

```
class CompositeLocator {  
    String header  
    String tag  
    String text  
    String trailer  
    def position  
    Map<String, String> attributes = [:]  
}
```

- Each clocator will generate a run time relative locator and it makes Tellurium robust to changes inside UI module.

New Concepts: Group Locating



- Usually XPath only includes one path to the target node, for example, **`/div/div/form/input`**, that is to say, only information along this path is used.
- We can use information in the nested object such as the whole form in Fig (B), i.e., **“A Form whose children are input (attributes), selector (attributes), and input(attributes)”**
- The composite object has a reference point and once the reference point is determined, the other UI elements can be located by their relations to the reference point.

New Concepts: UI Template

- UI template can be used for
 - Unknown number of UI elements, but the types are known
 - Many and similar UI elements
- Tellurium List and Table objects use UI templates to define the UI elements they include inside.

```
ui.Table(uid: "table", clocator: [:]){
  InputBox(uid: "row: 1, column: 1", clocator: [:])
  Selector(uid: "row: *, column: 2", clocator: [:])
  UrlLink(uid: "row: 3, column: *", clocator: [:])
  TextBox(uid: "all", clocator: [:])
}
```

New Concepts: Widgets

- Tellurium provides the capability to group UI objects into a widget object and then you can use the widget directly just like using a tellurium UI object.
 - Reusable
 - Use name space to avoid name collision
 - Compiled as a jar file
 - Load from Tellurium configuration file
- Tellurium provides an onWidget method
 - onWidget(String uid, String method, Object[] args)

Example: JtvTabContainer DOJO widget

```
ui.Container(uid: "productTab", clocator: [tag: "div"], id:
"*jtv_widget_JtvTabContainer"){
    . . . . .
    DOJO_JtvTabContainer(uid: "TabContainer", clocator: [:], tabs["Tab 1", "Tab
2", "Tab 3"])
}
onWidget "ProductTab.TabContainer", method, parameters
```

New Concepts: DslContext

- The DSL context is the heart of the Tellurium and all UI modules must extend the DslContext class
- Defines DSL expression, for instance, “click id”
- DSL context includes
 - UI object definitions
 - event actions
 - data access
 - Test support
- DslContext is extended to support Data Driven Testing, which includes input file layout, test definition, and testing flow.

Tellurium BookList Demo

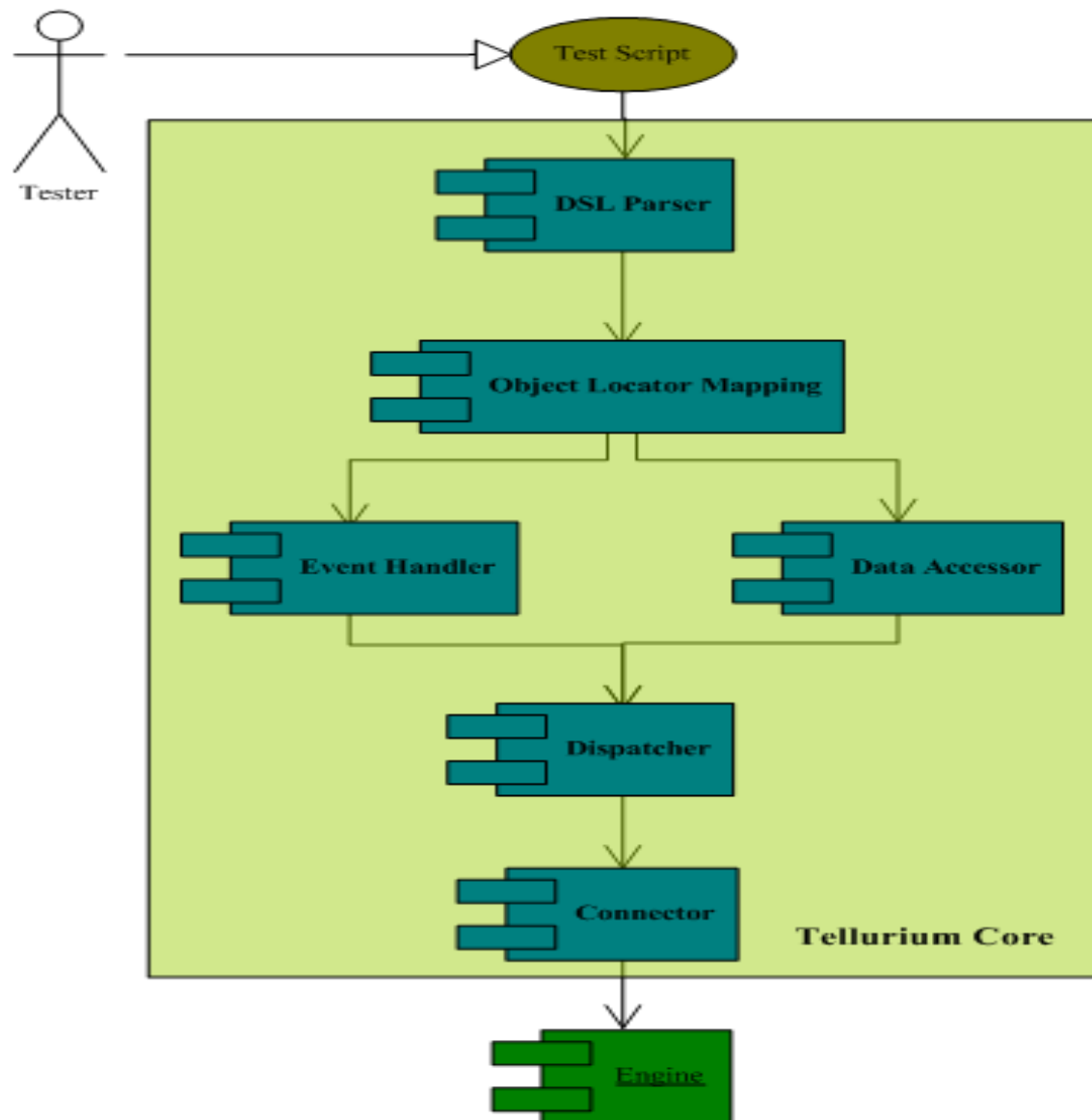
Do you need to know Groovy ?

- Groovy is a dynamic language for Java
 - Compatible with Java syntax
 - Object oriented and expressive
 - MetaClass for meta programming
 - Gstring: “label=\${target}”
 - Closures and Optional types
- Tellurium Core is implemented in Groovy and your **UI module class must be a Groovy class.**
- Test cases can be written in Java, Groovy, or pure DSL scripts.
- You do not really need to know Groovy when use Tellurium
 - UI definition is descriptive using DSL
 - Methods for UI module can be written in Java syntax
 - But using Groovy features will make your code more concise
- Groovy IDE support (Eclipse, Netbeans, IntelliJ)

Outline

- Overview
- Problems in Selenium
- Tellurium in Action
- Tellurium's solution
- Tellurium concepts
- Tellurium UI Model Plugin (Trump)
- ***Architecture***
- Projects
- 0.7.0 Feature Preview
- Resources

Tellurium Architecture



Tellurium Architecture

- Test scripts
 - UI Modules (UI definition)
 - Extend DslContext
 - Consist of nested UI objects
 - Simple Objects: Button, CheckBox, Selector, ...
 - Container: List, Table, Frame, ...
 - Widgets: Tellurium widget extension
 - Test Cases
 - Tellurium Groovy Test Case
 - Tellurium JUnit Test Case (JUnit 4 annotations)
 - Tellurium TestNG Test Case (TestNG annotations)
 - Pure DSL Script
- Object-Locator Mapping
 - XPath
 - JQuery selector

Tellurium Architecture (Cont'd)

- Event Handler

- Default events: mouseOver, focus, blur, ...
- UI Object's respond attribute

```
InputBox(uid: "searchbox", clocator: [title: "Google Search"], respond: ["click",  
"focus", "mouseOver", "mouseOut", "blur"])
```

- Data Accessor

- Data/Attributes
- Status/Availability

- Dispatcher

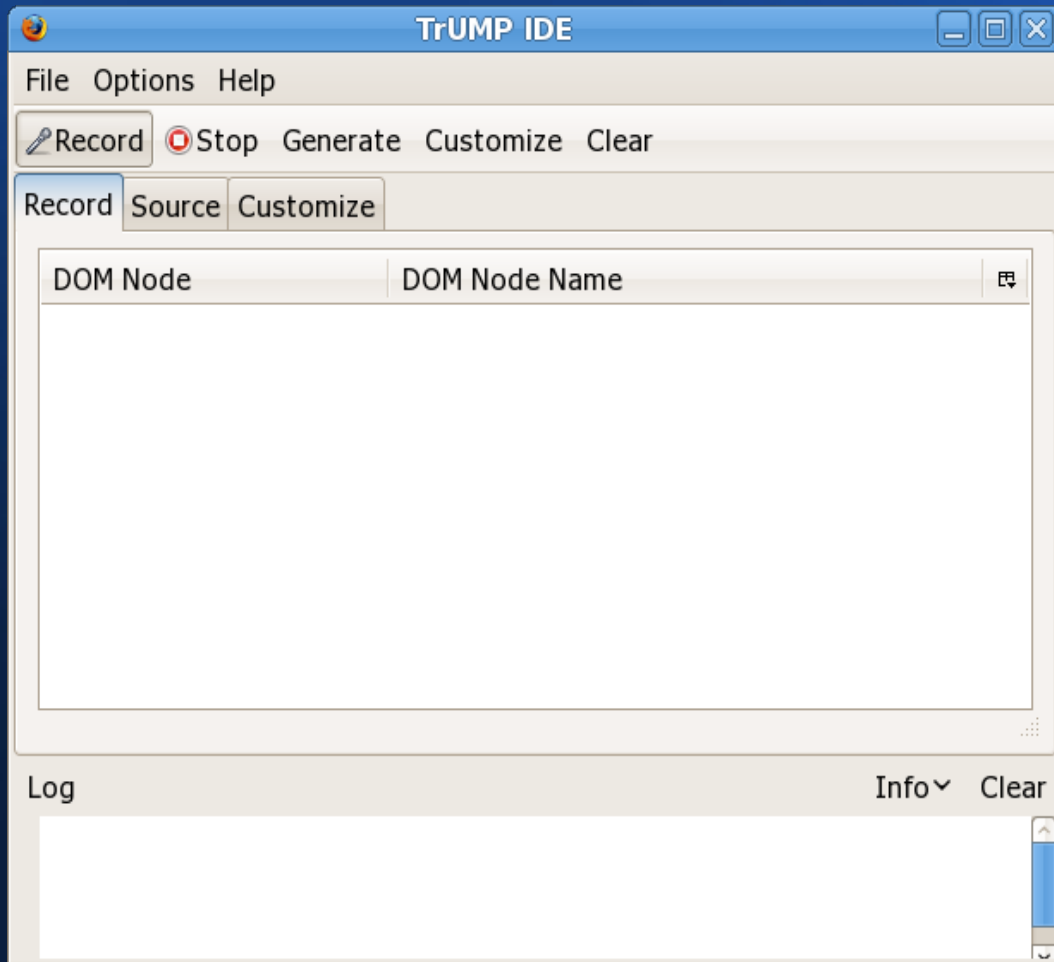
- Engine

- Modified Selenium Core
- JQuery Selector support
- Locator caching

Outline

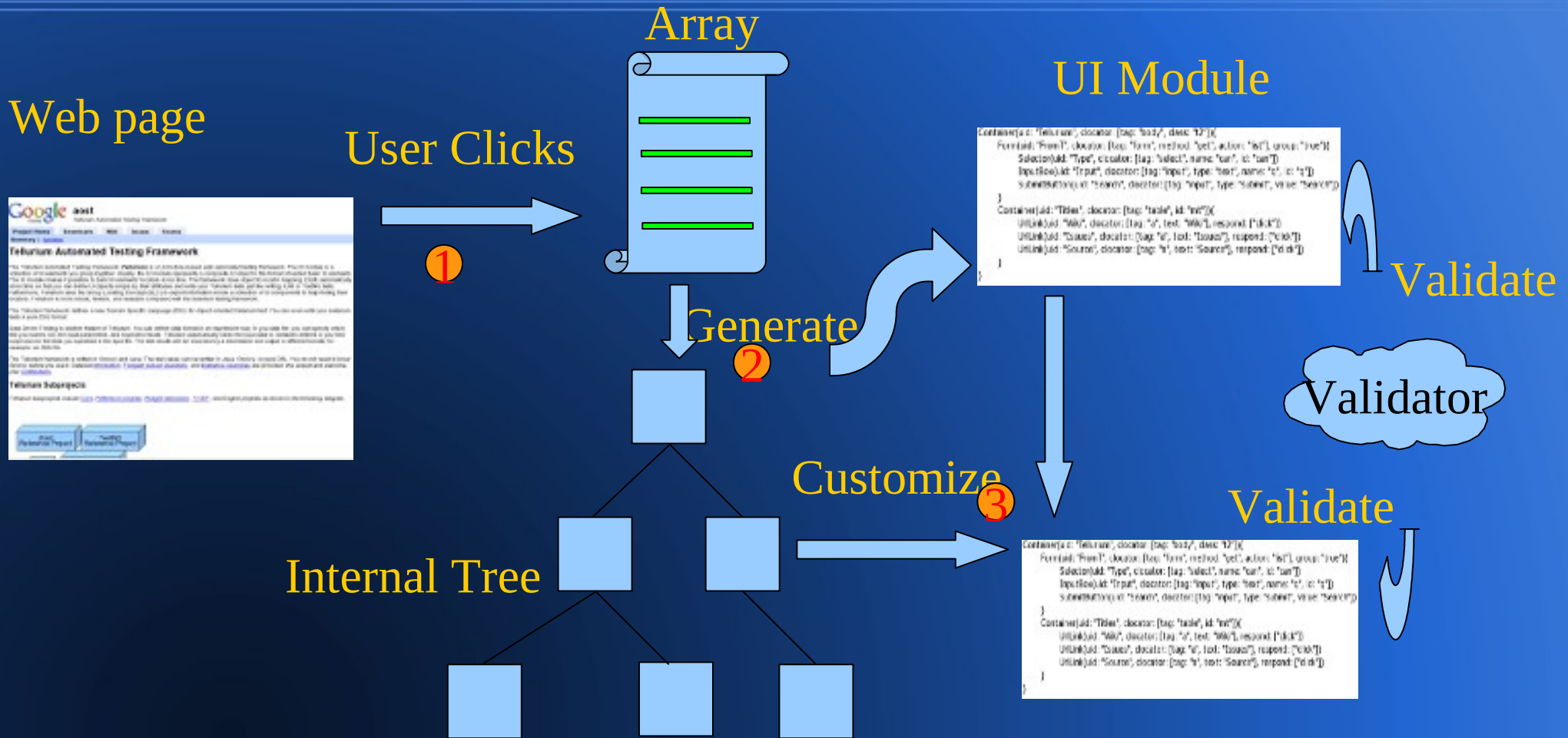
- Overview
- Problems in Selenium
- Tellurium in Action
- Tellurium's solution
- Tellurium concepts
- Architecture
- ***Tellurium UI Model Plugin (Trump)***
- Projects
- 0.7.0 Feature Preview
- Resources

Tellurium UI Model Plugin (TrUMP)

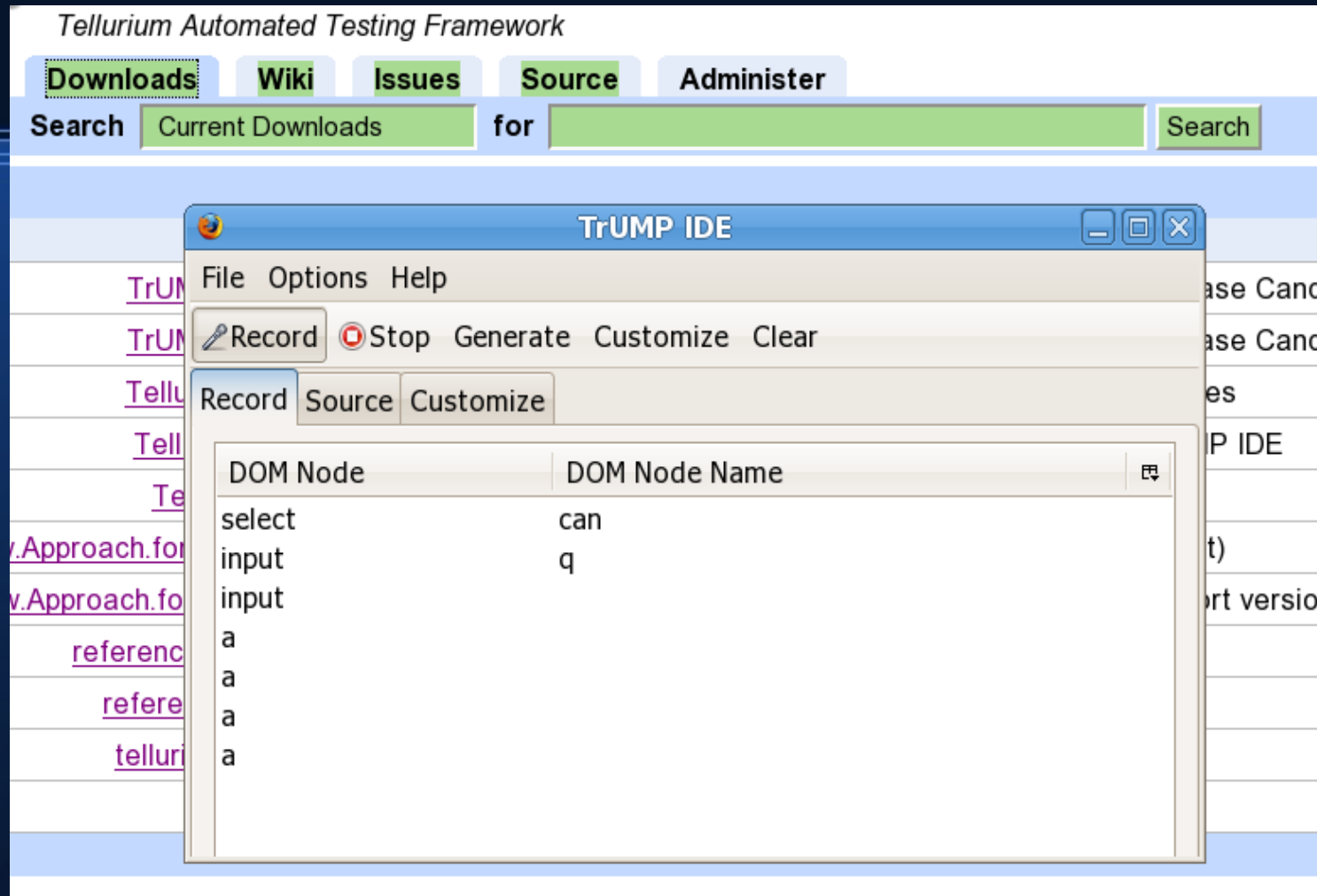


- TrUMP IDE is a Firefox plugin to automatically create UI modules for users
- Download TrUMP xpi file from Tellurium project site to install it
- Different xpi files for Firefox 3 and Firefox 2.
- They are the same except the way to show the view is different
- Logging and preference support

TrUMP: How it works

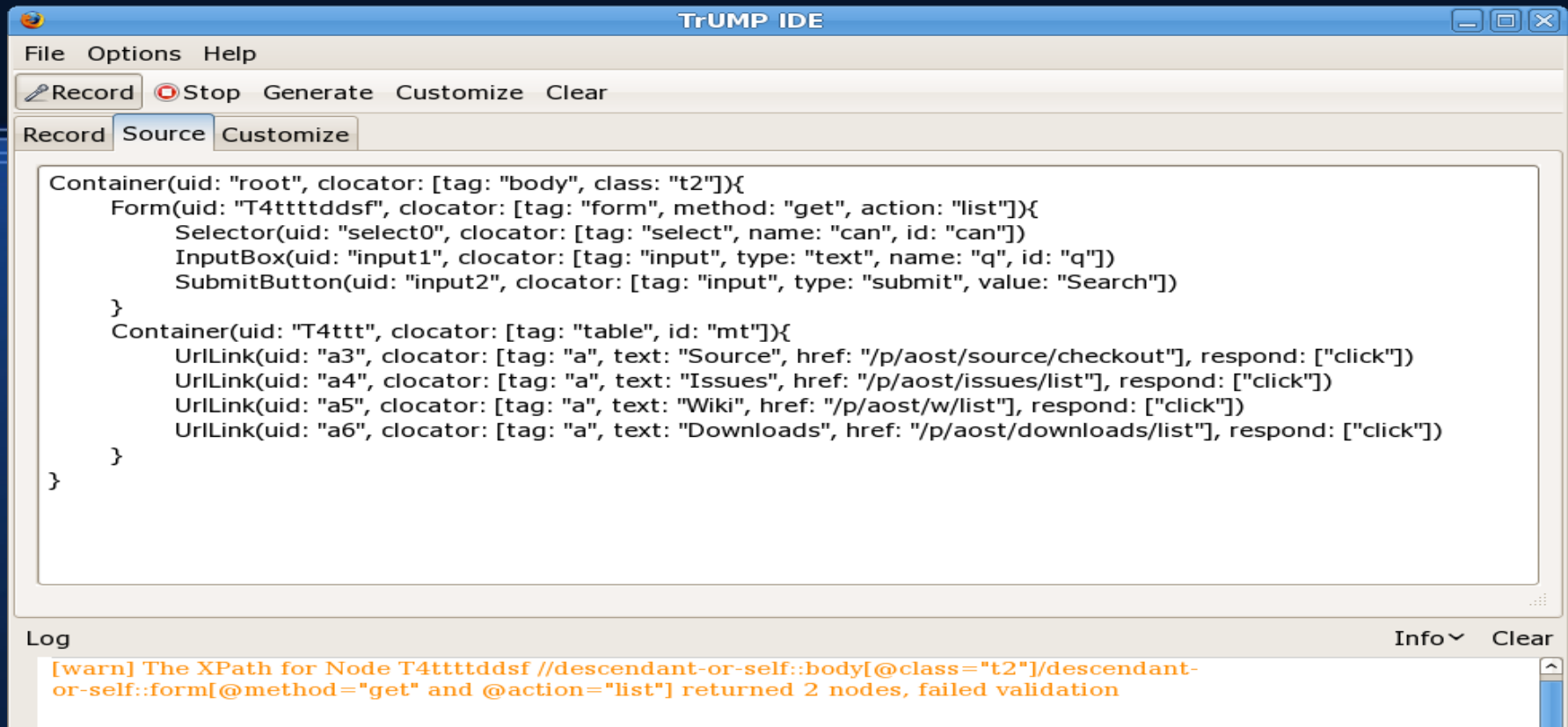


TrUMP: Recording



- Click on any element you want to choose
- Click one more time to unselect
- Click on the selected element and TrUMP will show you where it is on the web

TrUMP: Generate



The screenshot shows the TrUMP IDE window with the following components:

- Menu:** File, Options, Help
- Buttons:** Record, Stop, Generate, Customize, Clear
- Tabbed Interface:** Record, Source, Customize
- Code Editor:** Contains the following JSON-like code:

```
Container(uid: "root", clocator: [tag: "body", class: "t2"]){
  Form(uid: "T4ttttddsf", clocator: [tag: "form", method: "get", action: "list"]){
    Selector(uid: "select0", clocator: [tag: "select", name: "can", id: "can"])
    InputBox(uid: "input1", clocator: [tag: "input", type: "text", name: "q", id: "q"])
    SubmitButton(uid: "input2", clocator: [tag: "input", type: "submit", value: "Search"])
  }
  Container(uid: "T4ttt", clocator: [tag: "table", id: "mt"]){
    UrlLink(uid: "a3", clocator: [tag: "a", text: "Source", href: "/p/aost/source/checkout"], respond: ["click"])
    UrlLink(uid: "a4", clocator: [tag: "a", text: "Issues", href: "/p/aost/issues/list"], respond: ["click"])
    UrlLink(uid: "a5", clocator: [tag: "a", text: "Wiki", href: "/p/aost/w/list"], respond: ["click"])
    UrlLink(uid: "a6", clocator: [tag: "a", text: "Downloads", href: "/p/aost/downloads/list"], respond: ["click"])
  }
}
```
- Log:** Shows a warning message: `[warn] The XPath for Node T4ttttddsf //descendant-or-self::body[@class="t2"]/descendant-or-self::form[@method="get" and @action="list"] returned 2 nodes, failed validation`

- The “Generate” button will generate an internal tree first
- Then TrUMP generates the default UI module
 - Selected attributes from a White list
 - Event handlers are converted to the respond attribute
 - Group option is off by default

TrUMP: Customize

The screenshot shows the TrUMP IDE interface. The main window displays a tree view of web elements. The selected element is a `Container` with `uid: 'T4ttt'` and `doclocator: [tag: 'table', id: 'mt']`. Inside this container, there are several `UrlLink` elements. The selected `UrlLink` has `uid: 'a3'` and `doclocator: [tag: 'a', text: 'Issues', href: '/p/aost/issues/list', respond: ['click']]`. The configuration panel on the right shows the following details:

- UID: a3
- Type: UrlLink
- Group:
- Attributes:
 - href: /p/aost/issues/list
 - onclick: cancelBubble=true;
 - text: Issues
 - header: /th[4]/div[@class='tab inactive]

The Log window at the bottom shows the following messages:

```
[warn] The XPath for Node T4tttdds1 //descendant-or-self::body[@class="t2"]/descendant-or-self::form[@method="get" and @action="list"] returned 2 nodes, failed validation
[warn] The XPath for Node T4tttdds1 //descendant-or-self::body[@class="t2"]/descendant-or-self::form[@method="get" and @action="list"] returned 2 nodes, failed validation
```

- Group option is available to check
- Optional attributes are available to select
 - Position, header, and may others

TrUMP: How to Customize

The screenshot shows the TrUMP IDE interface. The main window is titled "TrUMP IDE" and has a menu bar with "File", "Options", and "Help". Below the menu bar are buttons for "Record", "Stop", "Generate", "Customize", and "Clear". The "Customize" tab is active, showing a tree view of UI elements. The tree view contains a "Container" with UID "Tellurium" and locator "[tag: 'body', class: 't2']". Under this container are several elements: a "Form" with UID "Form" and locator "[tag: 'form', method: 'get', action: 'list', group: 'true']", a "Selector" with UID "DownloadType" and locator "[tag: 'select', name: 'can', id: 'can']", an "InputBox" with UID "SearchBox" and locator "[tag: 'input', type: 'text', name: 'q', id: 'q']", and a "SubmitButton" with UID "Search" and locator "[tag: 'input', type: 'submit', value: 'Search']". Below the container are four "UrlLink" elements with UIDs "Source", "Issues", "Wiki", and "Downloads", each with a locator "[tag: 'a', text: '...', respond: ['click']]". To the right of the tree view is a configuration panel with "Show" and "Save" buttons. The panel has fields for "UID" (Tellurium), "Type" (Container), and "Group" (unchecked). Below these is a table of "Attributes":

Name	Value
<input checked="" type="checkbox"/> class	t2
<input type="checkbox"/> header	/html

At the bottom of the window is a "Log" panel with "Info" and "Clear" buttons. It contains two warning messages:

```
[warn] The XPath for Node T4ttttddsf //descendant-or-self::body[@class="t2"]/descendant-or-self::form[@method="get" and @action="list"] returned 2 nodes, failed validation  
[warn] The XPath for Node T4ttttddsf //descendant-or-self::body[descendant::form[@method="get" and @action="list"] and
```

- You can change UIDs and UI types
- Select or remove attributes
- Turn on group option
- TrUMP automatically validates the UI module. Avoid attributes that may change such as position and header.
- Red **X** mark.

TrUMP: Export UI Module

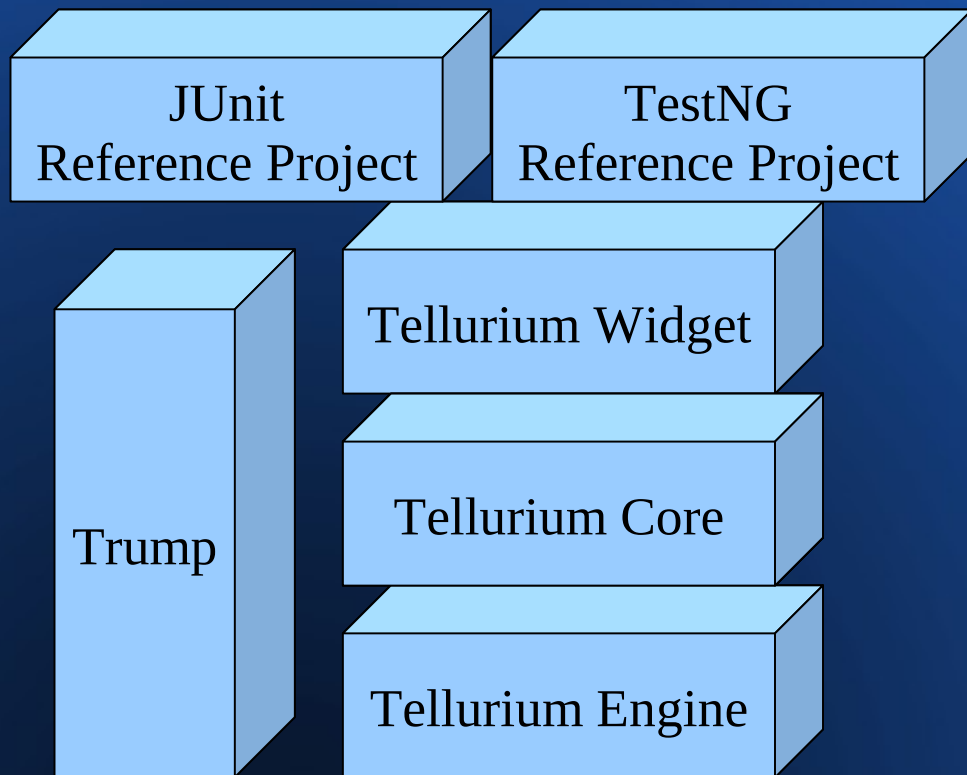
```
package tellurium.ui
import org.tellurium.dsl.DslContext
/**
 * This UI module file is automatically generated by
 * TrUMP 0.1.0.
 */
class NewUiModule extends DslContext{
    public void defineUi() {
        ui.Container(uid: "Tellurium",
        clocator: [tag: "body", class: "t2"]){
            Form(uid: "Form", clocator:
            [tag: "form", method: "get", action: "list", group:
            "true"]){
                Selector(uid:
                "DownloadType", clocator: [tag: "select", name:
                "can", id: "can"])
                InputBox(uid:
                "SearchBox", clocator: [tag: "input", type: "text",
                name: "q", id: "q"])
                SubmitButton(uid:
                "Search", clocator: [tag: "input", type: "submit",
                value: "Search"])
            }
            Container(uid: "Title",
            clocator: [tag: "table", id: "mt"]){
                UrlLink(uid:
                "Issues", clocator: [tag: "a", text: "Issues"],
                respond: ["click"])
                UrlLink(uid: "Wiki",
                clocator: [tag: "a", text: "Wiki"], respond:
                ["click"])
                UrlLink(uid:
                "Downloads", clocator: [tag: "a", text: "Downloads"],
                respond: ["click"])
            }
        }
    }
    //Add your methods here
}
```

- Export UI module to a groovy file with everything setting up for you
- The plugin only generates the UI modules, not the test itself, which is different from Selenium IDE. You need to define methods for the UI modules.
- Create JUnit or TestNG Java test cases based on the UI module

Outline

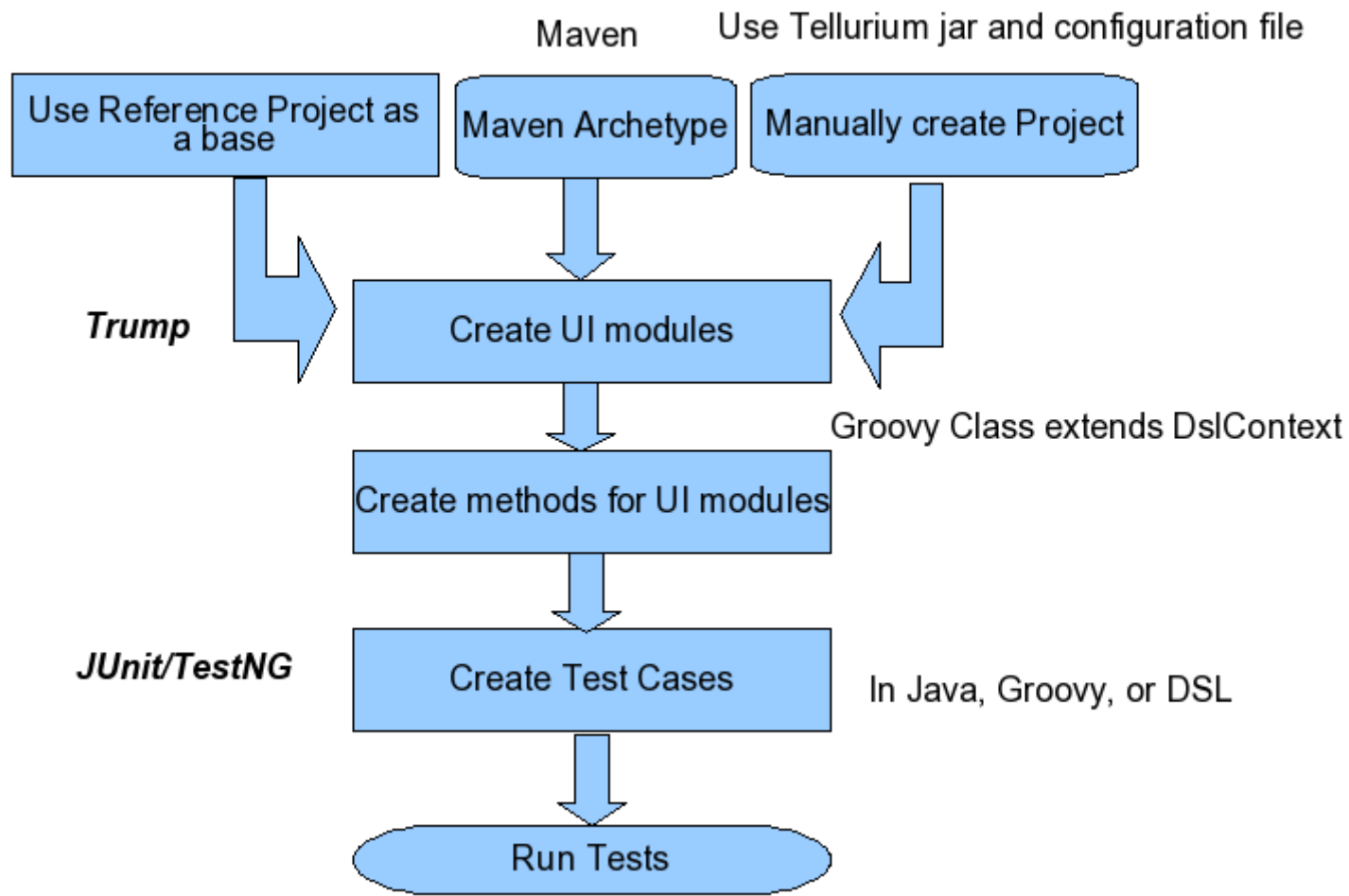
- Overview
- Problems in Selenium
- Tellurium in Action
- Tellurium's solution
- Tellurium concepts
- Architecture
- Tellurium UI Model Plugin (Trump)
- ***Projects***
- 0.7.0 Feature Preview
- Resources

Tellurium Projects



- Tellurium Engine: Similar to Selenium Core and will be Tellurium test driving engine. More efficient and supports nested UI object.
- Tellurium Core: UI objects, locator mapping, and test support.
- Tellurium widget: DOJO/ExtJS widget extension
- Tellurium Reference Projects: JUnit 4 and TestNG projects to demonstrate how to create real world Tellurium test cases using Tellurium project web site as a reference.
- Trump: Firefox Plugin to automatically generate UI modules

How to use Tellurium?

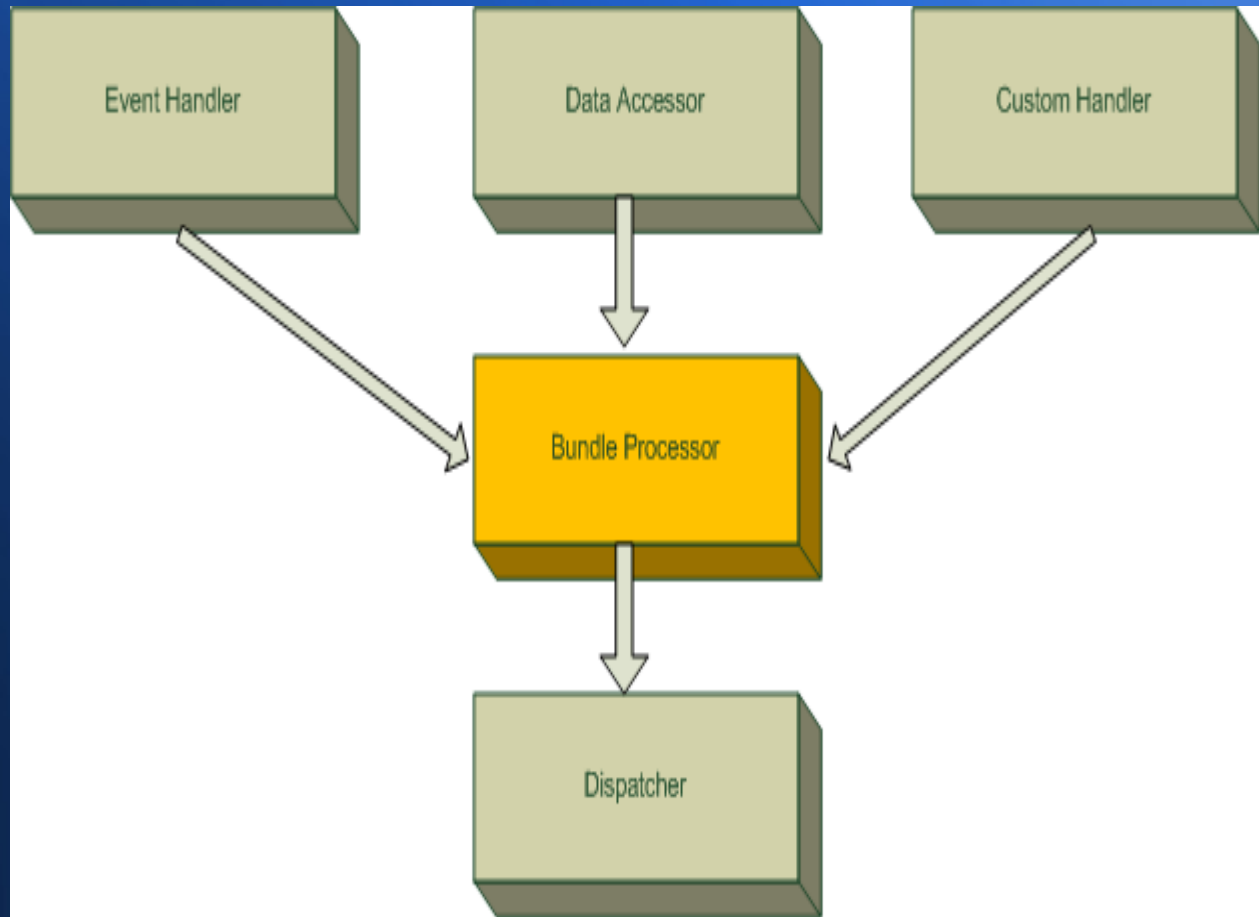


Outline

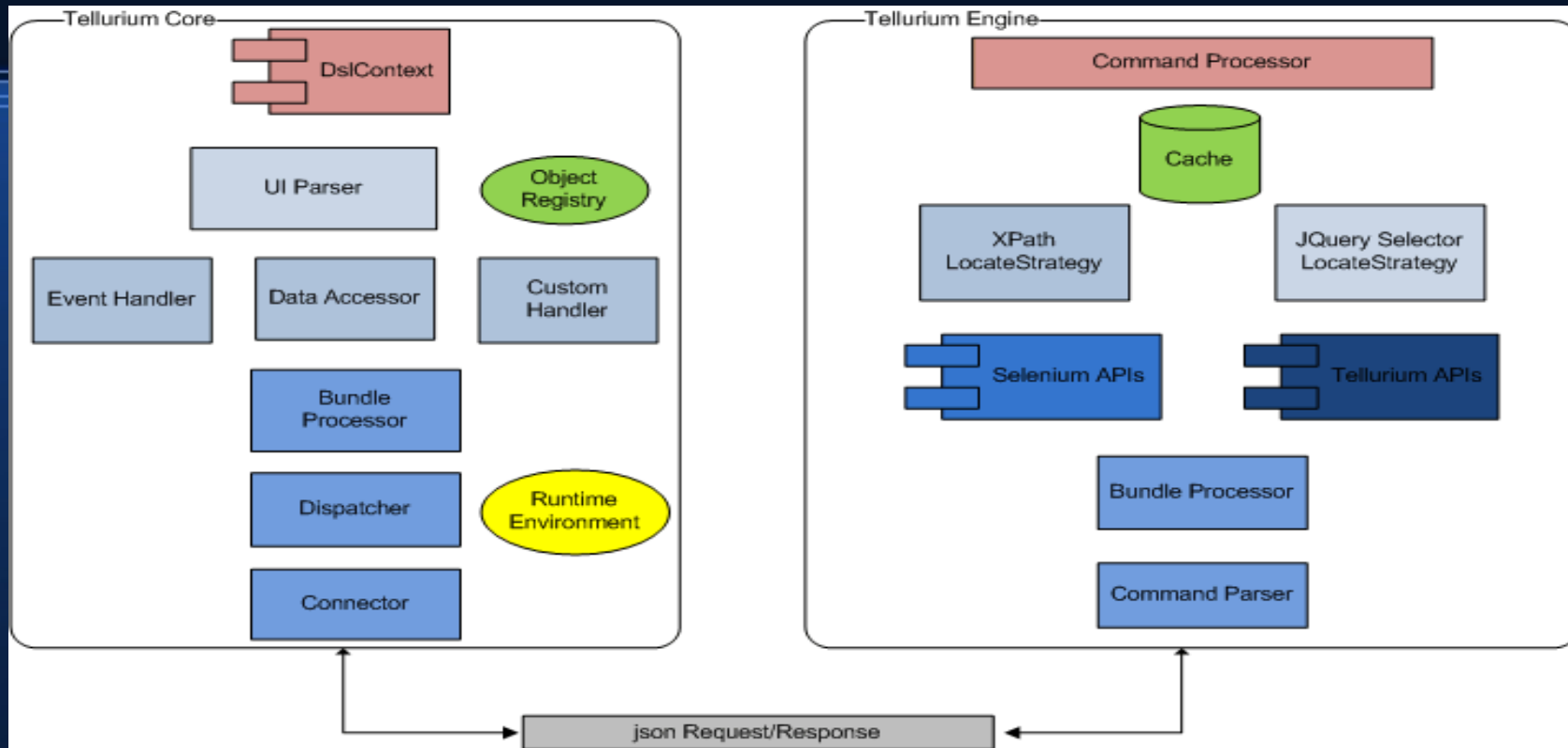
- Overview
- Problems in Selenium
- Tellurium in Action
- Tellurium's solution
- Tellurium concepts
- Architecture
- Tellurium UI Model Plugin (Trump)
- Projects
- ***0.7.0 Feature Preview***
- Resources

0.7.0 Feature Preview

- Macro Command
- I18N support
- Runtime diagnosis
 - *diagnose(uid)*
- More flexible UI templates (*UID Description Language*)
- Build-in trace on execution time
 - *useTrace(boolean)*



0.7.0 Feature Preview (Cont'd)



- UI module caching
- New API based on jQuery selector
- Group locating based on partial information
- Integration with Finesse

Outline

- Overview
- Problems in Selenium
- Tellurium in Action
- Tellurium's solution
- Tellurium concepts
- Architecture
- Tellurium UI Model Plugin (Trump)
- Projects
- 0.7.0 Feature Preview
- ***Resources***

Resources

- **Tellurium Project website:** <http://code.google.com/p/aost/>
- **Tellurium User Group:** <http://groups.google.com/group/tellurium-users>
- **Trump:** <https://addons.mozilla.org/en-US/firefox/addon/11035>
- **Tellurium on Twitter:** <http://twitter.com/TelluriumTest>
- **Tellurium IRC channel at freenode.net: #tellurium**

Q & A