codecentric
agile software factory

GEB

Groovy Browser Automation

AGENDA

- What is Geb?
- Quickstart
- Configuration & Testing
- Page Object Pattern
- Asynchronous Assertions

WHAT IS GEB?

Geb is …

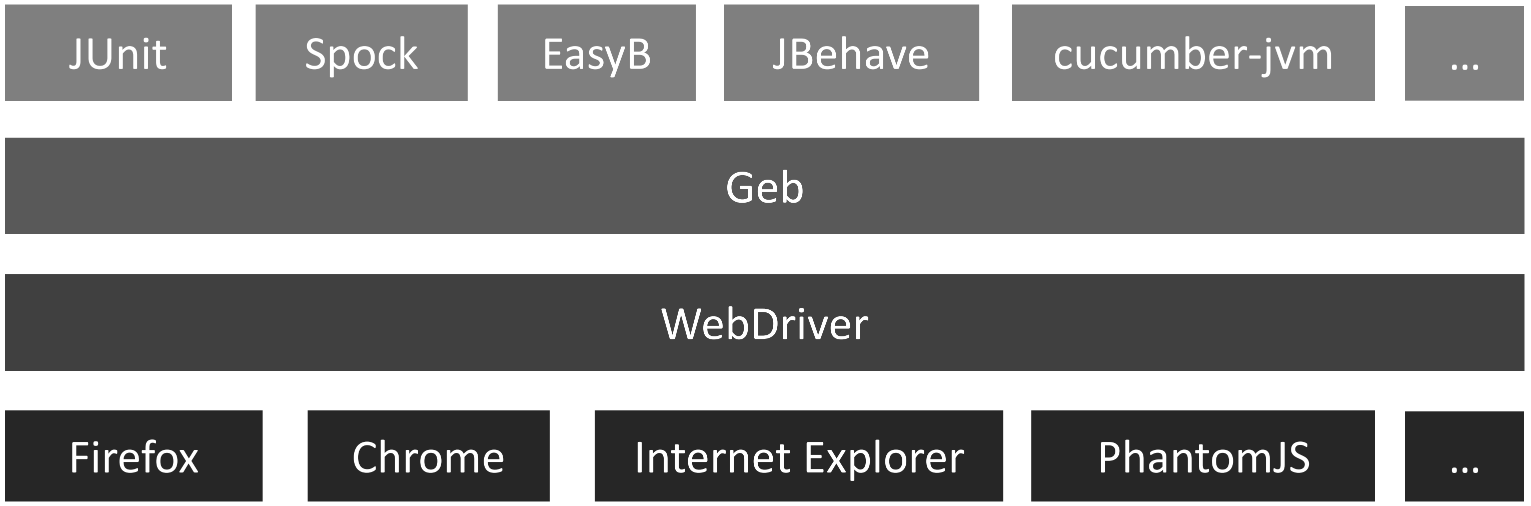
- … pronounced „Jeb“
- … the Egyptian god of the Earth
- … a browser automation solution
- … written in Groovy
- … a library on top of WebDriver (aka Selenium 2)

WHAT IS GEB?

Geb offers …

- … a jQuery-like selection & traversal API
- … support for the Page Object Pattern
- … good integration in testing frameworks
- … easy configuration
- … asynchronous element selection

AUTOMATED (WEB) ACCEPTANCE TESTING



QUICKSTART

CODING DEMO

CONFIGURATION & TESTING - MAVEN

Step 1: Maven configuration

- groovy
  - gmaven or groovy-eclipse-compiler
- spock
  - or any other test framework
- geb-spock
  - or geb-core, geb-junit, ...
- selenium-firefox-driver
  - or selenium-chrome-driver, ...
- maven-surefire-plugin

```
<dependencies>
  <dependency>
    <groupId>org.codehaus.groovy</groupId>
    <artifactId>groovy-all</artifactId>
    <version>2.1.1</version>
  </dependency>
  <dependency>
    <groupId>org.spockframework</groupId>
    <artifactId>spock-core</artifactId>
    <version>0.7-groovy-2.0</version>
  </dependency>
  <dependency>
    <groupId>org.gebish</groupId>
    <artifactId>geb-spock</artifactId>
    <version>0.9.0-RC-1</version>
  </dependency>
  <dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-firefox-driver</artifactId>
    <version>2.29.1</version>
  </dependency>
</dependencies>
```

CONFIGURATION & TESTING - SPOCK

Step 2: A simple test case

- Base class GebReportingSpec
- Geb takes care of
  - WebDriver instantiation and shutdown
  - WebDriver caching
  - Cookie clearing between tests
- Automatic Reporting
  - Html and screenshot (configurable)
  - Directory configurable
  - Always or for failed tests only
- All urls relative to base url
- Configured via system properties

```
class SimpleSpec extends GebReportingSpec {
    def "create a new movie"() {
        given: "a user at add movie page"
        // go directly to base url
        // equivalent to: browser.go(baseUrl)
        go()

        // click on the new movie button
        $("a", text: "Add movie").click()

        when: "the user creates a new movie"
        // fill the input form
        def form = $("#pageContent form")
        form.title = "Django Unchained"
        form.startDate = "2012-01-17"
        form.description = "The latest Tarantino movie?"

        // submit the form
        form.find("button[type=submit]").click()

        then: "he can find it in the movie list"
        // assert that the new movie is in the movie-list
        assert $("#pageContent tr").any {
            it.find("td", 0).text() == "Django Unchained"
        }
    }
}
```

Run tests with:

```
mvn -Dgeb.build.baseUrl=http://localhost:8080/movies -Dgeb.build.reportsDir=path/to/geb/reports test
```

CONFIGURATION & TESTING - GEB

Step 3: Geb configuration

- Configuration via system properties
  - geb.driver
  - geb.build.baseUrl
  - geb.build.reportsDir
- Configuration in GebConfig.groovy
  - environment sensitive (system property „geb.env“)
  - possibility to use custom WebDriver implementations
  - configure presets for waiting
  - cacheDriverPerThread
  - reportOnTestFailureOnly

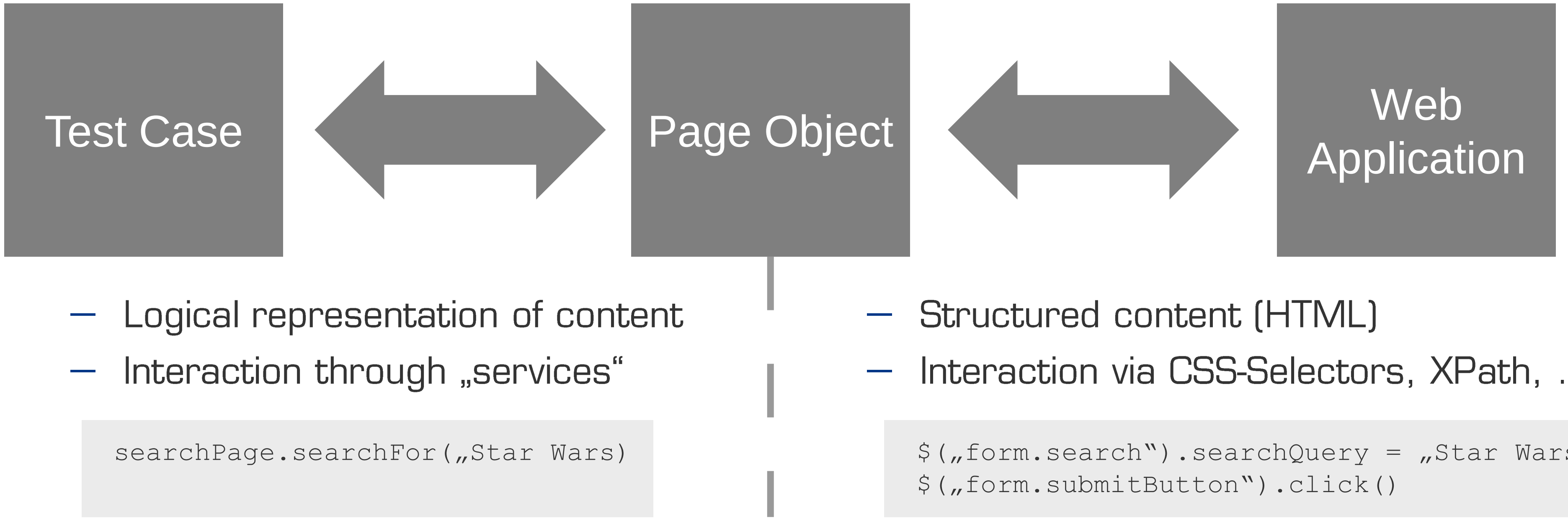
```
def caps = new DesiredCapabilities([
    "javascriptEnabled": true,
    "takeScreenshot": true])

// default driver if no geb.env system property is defined
driver = { new FirefoxDriver(caps) }

reportOnTestFailurOnly = true

environments {

  chrome { driver = { new ChromeDriver(caps) } }

  remote {
    driver = {
      // read driver url from system property
      def driverUrl = System.getProperty("geb.driverUrl")
      new RemoteWebDriver(new URL(driverUrl), caps)
    }
  }

  htmlunit { driver = { new HtmlUnitDriver() } }

}
```

CONFIGURATION & TESTING - DEMO

CODING DEMO

PAGE OBJECT PATTERN

- A page object is a logical representation of a single screen of the web application.
- Interaction of tests with the content should only happen via page objects.



- Logical representation of content
- Interaction through „services“

```
searchPage.searchFor(„Star Wars)
```

- Structured content (HTML)
- Interaction via CSS-Selectors, XPath, …

```
$(„form.search“).searchQuery = „Star Wars“
$(„form.submitButton“).click()
```

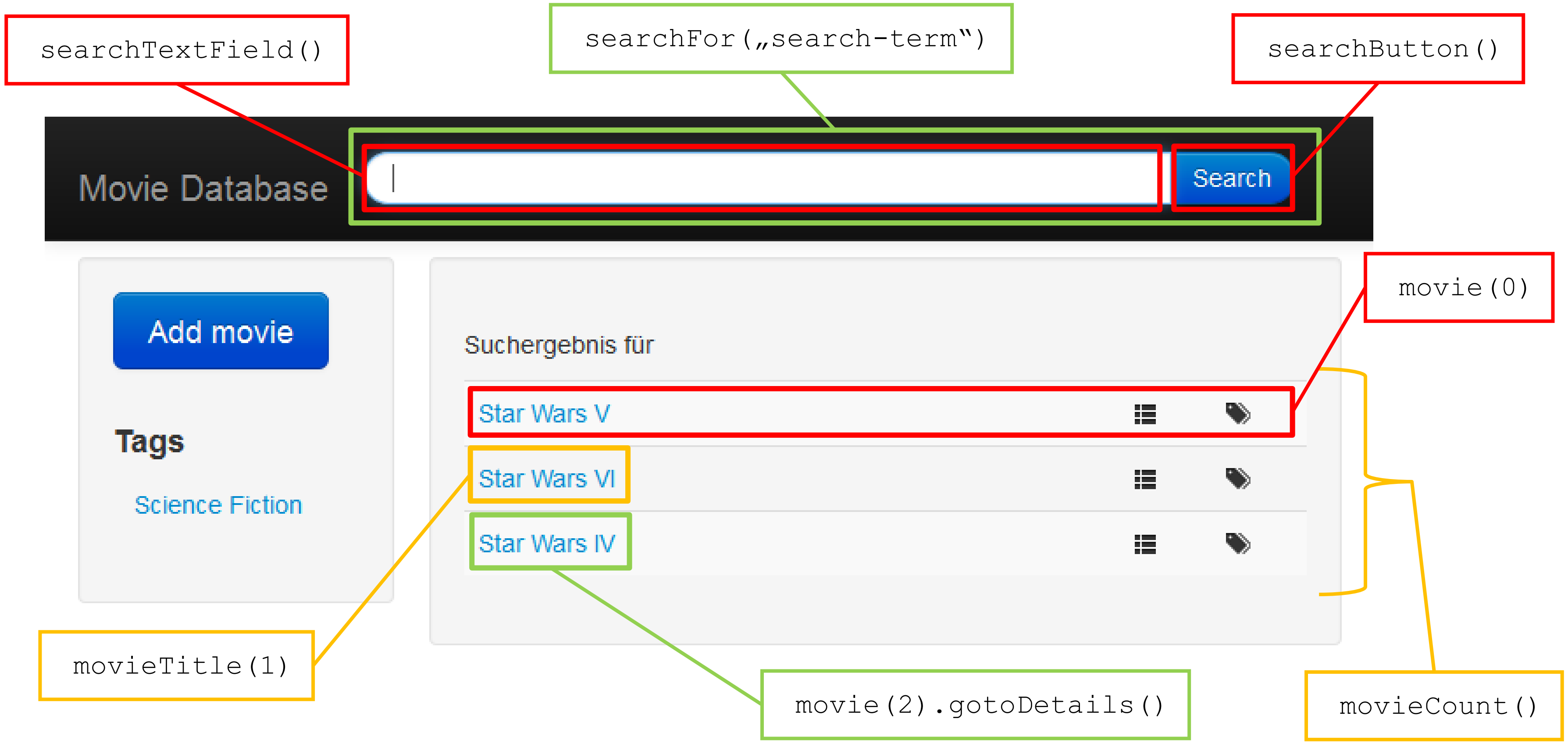
PAGE OBJECT PATTERN – EXAMPLE

Legend:

Element

Information

Action



PAGE OBJECT PATTERN – IMPLEMENTATION

```groovy
class MovieDatabaseHomePage extends Page {
    static content = {
        // content templates are defined using a simple dsl:
        // <name> (<options map>) { <definition> }
        searchForm { $("#searchbar .form-search") }

        // the <definition> is evaluated against the page-instance,
        // so calling other templates is possible
        searchTextField { searchForm.searchString }

         // templates can be used to execute actions, too
         searchSubmitButton { searchForm.find(type: "submit") }

         // it is also possible to pass arguments to templates
         searchFor { searchTerm ->
             searchTextField = searchTerm
             searchSubmitButton.click()
         }

         // if a template returns a Navigator object, geb normally throws an
         // RequiredPageContentNotPresent, when the no content is found
         // setting the option "required" to false disables this behaviour
         movies(required: false) { $("#pageContent tr") }

         // required=true is the default
         movie(required: true) { index -> movies.find(index) }

         // templates can return arbitrary data
         movieCount() { movies().size() }
         movieTitle { index -> movie(index).text() }
    }
}
```

PAGE OBJECT PATTERN - USAGE

```
class SearchSpec extends GebReportingSpec {

    def "search for movie"() {
        given: "a user at the moviedatabase homepage"
        // set the current page object to an instance of MovieDatabaseHomePage
        to MovieDatabaseHomePage

        when: "the user searches for 'Star'"
        // equivalent to: browser.page.searchFor("Star")
         searchFor("Star")

         then: "the search result contains the movie 'Star Wars'"
         // equivalent to: assert browser.page.movieTitle(0) == "Star Wars"
         movieTitle(0) == "Star Wars"
     }

     def "search for nonexistant movie"() {
         given: "a user at the moviedatabase homepage"
         to MovieDatabaseHomePage

         when: "the user searches for 'Foo'"
         searchFor("Foo")

         then: "the search result is empty"
         movieCount() == 0
     }

}
```

PAGE OBJECT PATTERN – FURTHER TOPICS

- At checking
  - Check if the displayed web page matches a certain Page Object type
  - Conditions defined inside Page definition
  - Execute check inside test using „at“-method
- Modules
  - „Small“ Page Objects
  - Model parts of a web page
  - Can be included in Page Objects
- Lifecycle Hooks
  - onLoad, onUnload
- Inheritance
- Parametrized Page Objects

ASYNCHRONOUS ELEMENT SELECTION

CODING DEMO

RESOURCES

- Geb Homepage: http://www.gebish.org
- Book of Geb: http://www.gebish.org/manual/0.9.0-RC-1/
- Introductory Blog Post: http://blog.codecentric.de/en/2013/02/browser-automation-and-acceptance-testing-with-geb/
- Example projects on GitHub
  - https://github.com/geb/geb-example-maven
  - http://github.com/geb/geb-example-grails
  - http://github.com/geb/geb-example-gradle
  - https://github.com/mlex/movie-database

FRAGEN?

Michael Lex

codecentric AG
Kölner Landstraße 11
40591 Düsseldorf

michael.lex@codecentric.de

www.codecentric.de
www.mbg-online.de
blog.codecentric.de
www.meettheexperts.de