Patterns in Frontendtests

Used Patterns

- Page Object
- Loadable Component
- Dependency Injection
- Page Factory

Page Object Pattern

- entkoppelt Testcode von UI
  - nur das PO sollte Details der Page kennen
- PO repräsentiert "Dienste" einer Page
- bietet API für
  - alles was ein User sieht
  - alles was ein User tun kann
- POs bilden Hierarchie die der Page-Hierarchie entspricht
  - Komposition, keine Vererbung!
- http://code.google.com/p/selenium/wiki/PageObjects

Loadable Component Pattern

- Basisklasse für Page Objects
- konzentriert Boilerplate
  - load()
  - isLoaded()
  - getBaseUrl() // base URL of component
  - getDriver()
  - getTitle()
  - implicitlyWait()
  - ...
- alle POs erben von dieser Klasse
- http://code.google.com/p/selenium/wiki/LoadableComponent

Example

- Platform (http://www.kwick.de)
  - Logged In
    - Email lesen
    - Email senden
    - ...
  - Mitglieder suchen
  - ...

Example

```
class PlatformPage extends LoadableComponent {
    PlatformPage(URI uri) { ... }
}

class AuthPage extends LoadableComponent {
    AuthPage(PlatformPage parent) { ... }
}

class SendEmailPage extends LoadableComponent {
    SendEmailPage(AuthPage parent) { ... }
}
```

Example *mäh* BLOATED!1elf

```
class SendEmailPageTest {
    private SendEmailPage sut;

    @BeforeClass public static preparePage() {
        PlatformPage platform =
            new PlatformPage(new URI("http://www...."));
        AuthPage auth = new AuthPage(platform);
        sut = new SendEmailPage(auth);
    }

    @Test public void sendAnEmail() {
        sut.setSubject("Hello");
        ...
    }
}
```

Example w/ Guice DI

```
class SendEmailPageTest {
    private SendEmailPage sendEmailPage

    @BeforeClass public static preparePage() {
        sendEmailPage =
            PageFactory.create(SendEmailPage.class);
    }

    @Test public void sendAnEmail() {
        ...
    }
}
```

Example w/ Guice DI

```
class PlatformPage extends LoadableComponent {
    @Inject
    PlatformPage(URI uri) { ... }
}

class AuthPage extends LoadableComponent {
    @Inject
    AuthenticatedPage(PlatformPage parent) { ... }
}

class SendEmailPage extends LoadableComponent{
    @Inject
    SendEmailPage(AuthPage parent) { ... }
}
```

Page Module (Guice DI)

- konfiguriert die Abhängigkeiten für Guice (DI)
- überall wo ein @Inject dran steht agiert Guice
- Guice baut für uns den kompletten Objektgraphen
- Besipiele:
  - bind(WebDriver.**class**).to(FirefoxDriver.**class**);
  - bild(Type.class).toInstance(anObjectOfType);
  - @Provide WebDriver provideDriver() { ... }

Example

```
class SendEmailPageTest {
    private SendEmailPage sendEmailPage

    @BeforeClass public static preparePage() { ... }

    @Test public void sendAnEmail() {
        // NO selectors or IDs, NO knowledge of UI!
        sendEmailPage.setSubject("Hello");
        sendEmailPage.setBody("...");
        ...
        sendEmailPage.submit();
        assertTrue(sendEmailPage.isSubmitSuccessful());
        ...
    }
}
```

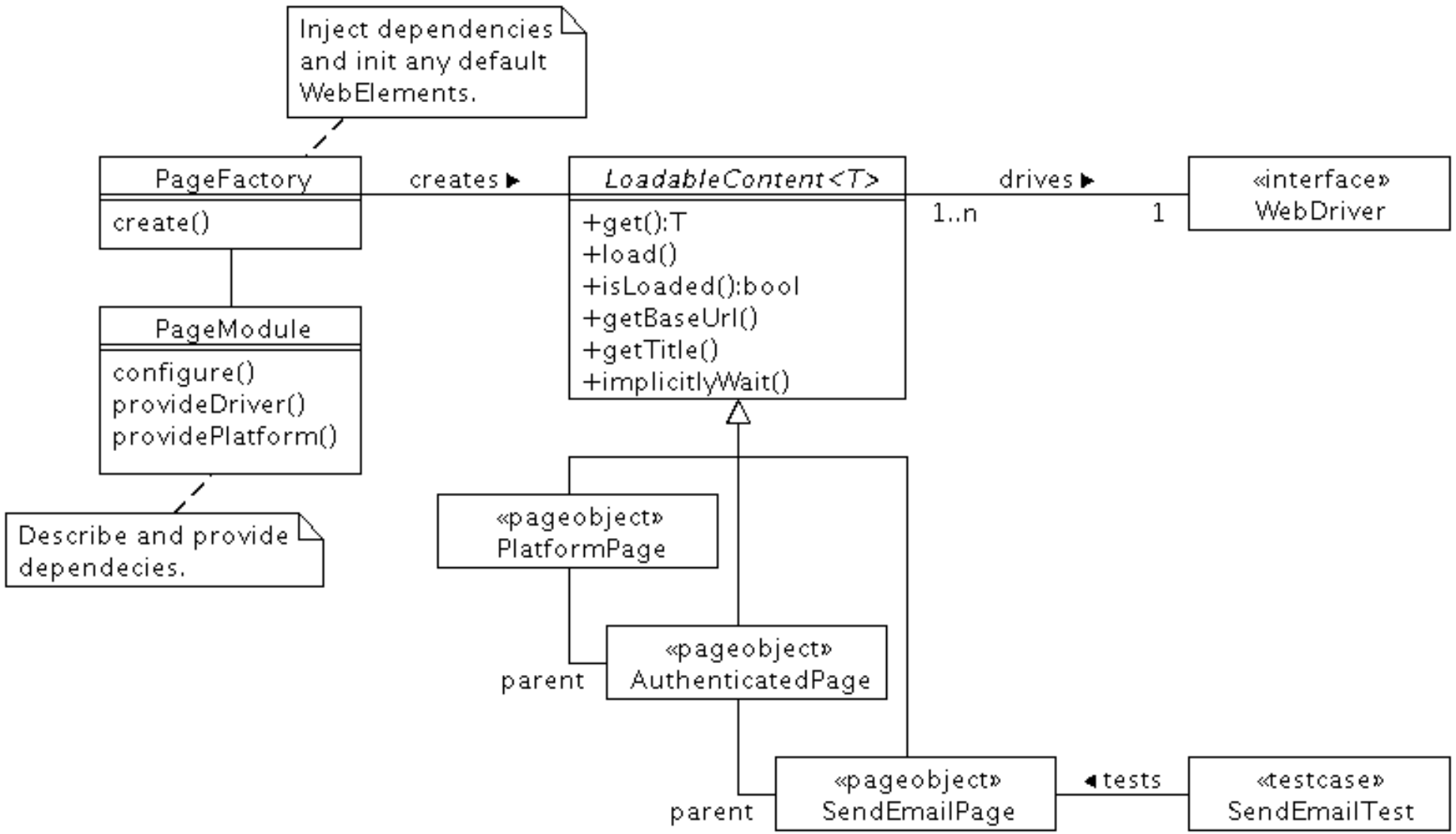
Page Facotry

- initialisiert Page Module
- initialisiert Guice Injector mit Page Module
- erzeugt PageObjects mit allen Abhängigkeiten
- initialisiert "automagische" WebElements des POs

"automagische" WebElements

```
class SendEmailPage extends LoadableComponent{

    WebElement submit; // id or name

    @FindBy(css = "a.cancelLink") WebElement cancel;


    @Inject
    SendEmailPage(AuthPage parent) { ... }

    public void setSubject(String subject) { ... }
}
```

Big Picture


Inject dependencies and init any default WebElements.
PageFactory
create()
creates
LoadableContent<T>
+get():T
+load()
+isLoaded():bool
+getBaseUrl()
+getTitle()
+implicitlyWait()
drives
1..n
1
«interface»
WebDriver
PageModule
configure()
provideDriver()
providePlatform()
Describe and provide dependecies.
«pageobject»
PlatformPage
parent
«pageobject»
AuthenticatedPage
parent
«pageobject»
SendEmailPage
tests
«testcase»
SendEmailTest