technische universität dortmund

fakultät für informatik

DSL-Driven Generation of User Documentations for Web Applications

BACHELORARBEIT

MUKENDI MPUTU

11.06.21

Inhalt

Hintergrund und Kontext

Woher kommt die Fragestellung?

- Wie ist die Fragestellung entstanden?

  Die Benutzererfahrung ist einer der wichtigsten Aspekte der Softwareentwicklung. Somit ist eine gute Dokumentation Schlüssel für eine zufriedenstellende User eXperience (UX) solcher Anwendungen.

- Welche Fachgebiete werden von der Fragestellung berührt?

  Jedes Fachgebiete der Software Entwicklung, welche Produkte für die Benutzung durch den End User anbietet.

- Wer braucht eine (bessere) Lösung?

  Die Software-Entwickler, Softwareprodukt-Anbieter …

Problematik

Was genau ist das Problem?

- Die manuelle Erzeugung der Dokumentation ist mühselig

  Aktuell werden Nutzerdokumentationen manuell erstellt, indem Nutzersequenzen reproduziert und dabei zahlreich Screenshots aufgenommen werden.

- Software-Änderungen werden oft nicht in Dokumentation reflektiert

  Nach jedem Update der Webanwendung müssen betroffene Nutzersequenzen erneut durchgegangen werden und Screenshots aktualisiert werden. Bei komplexer Anwendungen ist die mit großem Aufwand bzw. Kosten verbunden.

Stand der Forschung
Was ist der aktuelle Stand der Forschung zum Thema?

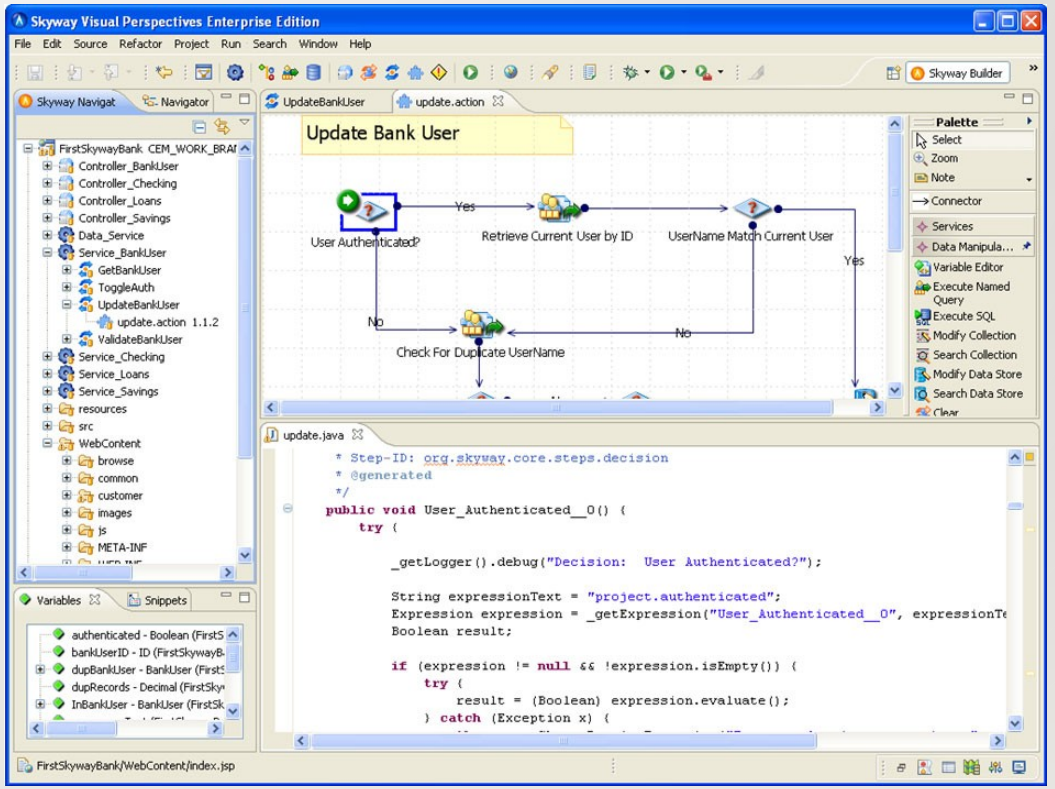
- Wer hat welche Voraussetzungen geschaffen?

  Eclipse-based Rich Client Platform mit dem Fokus auf Applikationsmodelentwicklung

- Welche vergleichbaren Ergebnisse gibt es bisher?

  GuideAutomator (Souza, Oliveira, 2017), Écrit toolkit (Descher et al., 2014)

Skyway Builder CE – Eclipse basierte Entwicklung von RIAs



https://www.eclipse.org/community/images/skyway.jpg

GuideAutomator
Mit Markdown und Selenium

- Eine Markdown-Datei muss im Voraus existieren
- Darin sind Befehle eingebettet, mit denen GuideAutomator Screenshots aufnimmt
- Selenium wird dabei benutzt, um den Browser zu treiben

Getting started

Description

Guide-automator extract `javascript` or `js` tags (```javascript guide-automator commands ``` or ```js guide-automator commands ```) from markdown file and generate manual from them. You need use our API commands in markdown file.

back to top

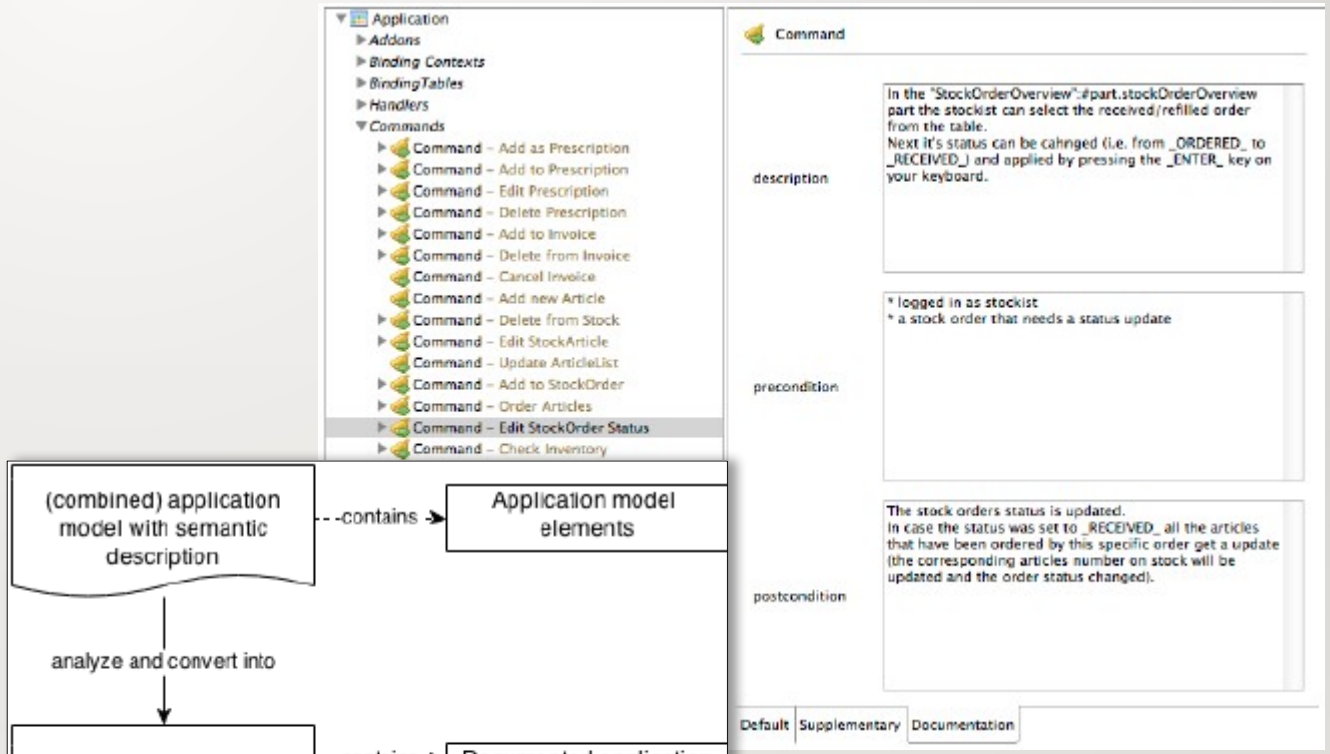
Example:

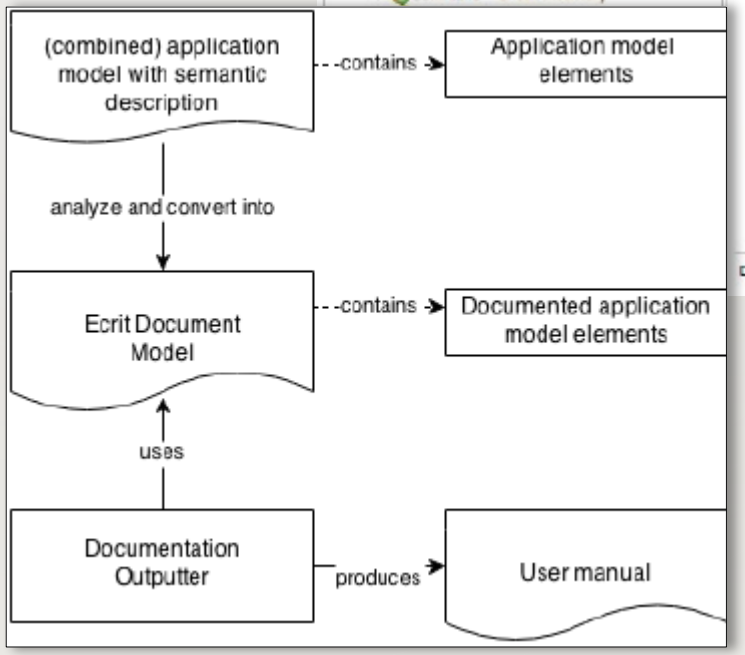
````
# This is my github

```js
  get('https://github.com/welbert');
  takeScreenshot();
  takeScreenshotOf('.avatar',false,true);

```(<- three back-ticks)
````

https://www.npmjs.com/package/guide-automator#getting-started

Écrit Toolkit
basiert auf Eclipse Applikationsmodell

- Modellierung der Anwendungsabläufe auf Basis von Eclipse RCP
- Applikationsmodell um semantische Beschreibung erweitern





https://www.researchgate.net/publication/268391920_Automated_user_documentation_generation_based_on_the_Eclipse_application_model

Stand der Forschung

Was ist der aktuelle Stand der Forschung zum Thema?

- Viele der bereits vorgeschlagenen Lösungen sind abhängig von bestimmten Entwicklungsparadigma und/oder Entwicklungsplattformen.
- Einige fokussieren sich nur auf Teilaspekte der Problematik
- Die Implementierung solcher Tools sind noch in der Entwicklung

Ziele

Was genau soll durch die Arbeit erreicht werden?

- Entwicklung eines Cinco-basierten Editors zur grafischen Modellierung von Nutzersequenzen
- Generierung einer Ausführungsumgebung, darauf basierend die Generierung einer Markdown-basierten Nutzerdokumentation
- Evaluation: Modellierung und Generierung einer Nutzerdokumentation für (Teile von) EquinOCS.

Relevanz und Nutzen

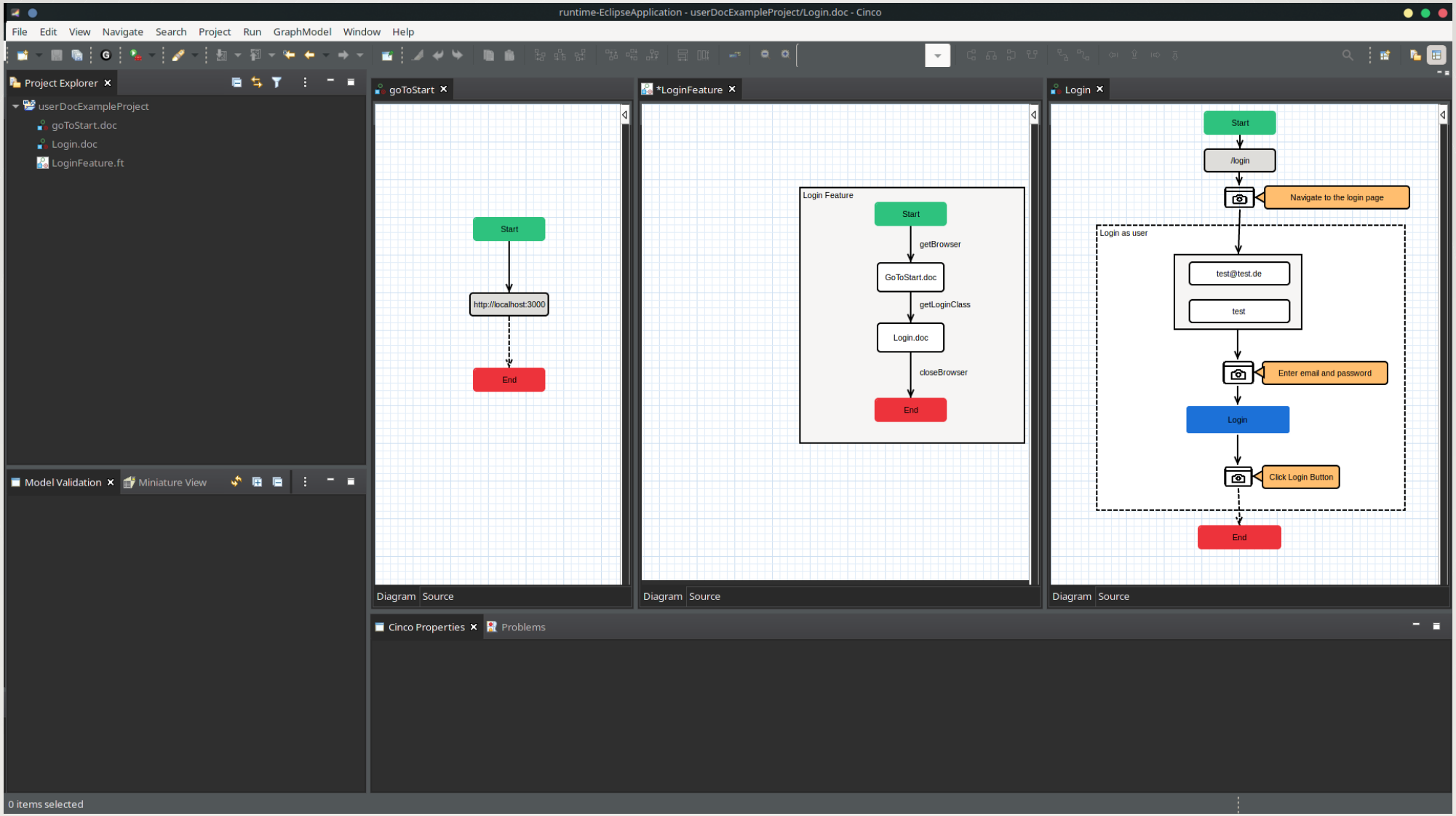
Inwiefern stellt die Arbeit einen Fortschritt dar?

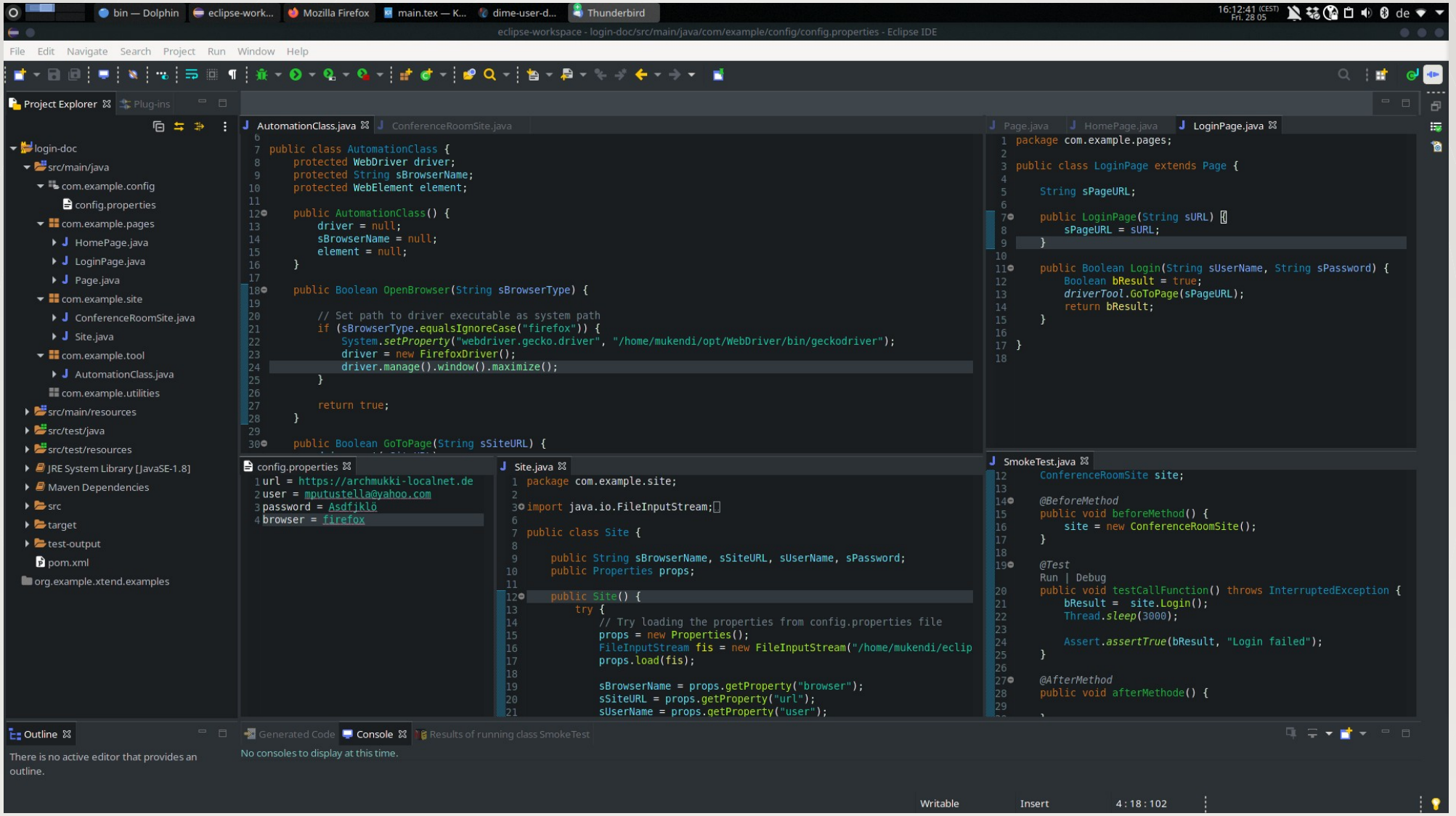
- Wie wird das Fachgebiet durch das Erreichen der Ziele vorangebracht?
  - Automatisierungsprozesse sind zeitsparend und kostensenkend. Softwareprojekte sind an Budget und Deadlines gebunden,
- Was wird dadurch z.B. schneller, effizienter, zuverlässiger ...?
  - … daher ist es ein großer Fortschritt die User Dokumentation schnell und kostengünstig erzeugen zu können.
- Wie profitieren Anwender davon?
  - Nicht nur die Entwickler profitieren davon, sondern auch der Endbenutzer der stets eine aktuelle Dokumentation der Anwendung zur Verfügung gestellt bekommt.

Methodologie

Wie sollen die Ziele erreicht werden?

- CINCO SCCE Meta Tooling Suite
  - Ermöglicht die Erzeugung von graphbasierten Modellierungswerkzeugen aus High-Level-Spezifikationen
  - Reine Programmierkenntnisse sind nicht mehr unbedingt notwendig
- Selenium WebDriver
  - Ermöglicht es den Browser so zu bedienen, wie es der Endbenutzer tun würde.
  - Deckt eine Vielzahl von Browserprogrammen ab.
- VuePress
  - Für die Entwicklung von Single Page Application
  - Optimiert für die Erstellung von technischen Dokumentationen


runtime-EclipseApplication - userDocExampleProject/Login.doc - Cinco
File Edit View Navigate Search Project Run GraphModel Window Help
Project Explorer
userDocExampleProject
goToStart.doc
Login.doc
LoginFeature.ft
Model Validation
Miniature View
goToStart
Start
http://localhost:3000
End
Diagram Source
*LoginFeature
Login Feature
Start
getBrowser
GoToStart.doc
getLoginClass
Login.doc
closeBrowser
End
Diagram Source
Login
Start
/login
Navigate to the login page
Login as user
test@test.de
test
Enter email and password
Login
Click Login Button
End
Diagram Source
Cinco Properties
Problems
0 items selected


bin — Dolphin
eclipse-work...
Mozilla Firefox
main.tex — K...
dime-user-d...
Thunderbird
eclipse-workspace - login-doc/src/main/java/com/example/config/config.properties - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explorer
Plug-ins
login-doc
src/main/java
com.example.config
config.properties
com.example.pages
HomePage.java
LoginPage.java
Page.java
com.example.site
ConferenceRoomSite.java
Site.java
com.example.tool
AutomationClass.java
com.example.utilities
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
test-output
pom.xml
org.example.xtend.examples
AutomationClass.java
ConferenceRoomSite.java
public class AutomationClass {
protected WebDriver driver;
protected String sBrowserName;
protected WebElement element;
public AutomationClass() {
driver = null;
sBrowserName = null;
element = null;
}
public Boolean OpenBrowser(String sBrowserType) {
// Set path to driver executable as system path
if (sBrowserType.equalsIgnoreCase("firefox")) {
System.setProperty("webdriver.gecko.driver", "/home/mukendi/opt/WebDriver/bin/geckodriver");
driver = new FirefoxDriver();
driver.manage().window().maximize();
}
return true;
}
public Boolean GoToPage(String sSiteURL) {
Page.java
HomePage.java
LoginPage.java
package com.example.pages;
public class LoginPage extends Page {
String sPageURL;
public LoginPage(String sURL) {
sPageURL = sURL;
}
public Boolean Login(String sUserName, String sPassword) {
Boolean bResult = true;
driverTool.GoToPage(sPageURL);
return bResult;
}
}
config.properties
url = https://archmukki-localnet.de
user = mputustella@yahoo.com
password = Asdfjklö
browser = firefox
Site.java
package com.example.site;
import java.io.FileInputStream;
public class Site {
public String sBrowserName, sSiteURL, sUserName, sPassword;
public Properties props;
public Site() {
try {
// Try loading the properties from config.properties file
props = new Properties();
FileInputStream fis = new FileInputStream("/home/mukendi/eclip
props.load(fis);
sBrowserName = props.getProperty("browser");
sSiteURL = props.getProperty("url");
sUserName = props.getProperty("user");
SmokeTest.java
ConferenceRoomSite site;
@BeforeMethod
public void beforeMethod() {
site = new ConferenceRoomSite();
}
@Test
Run | Debug
public void testCallFunction() throws InterruptedException {
bResult = site.Login();
Thread.sleep(3000);
Assert.assertTrue(bResult, "Login failed");
}
@AfterMethod
public void afterMethode() {
Outline
There is no active editor that provides an outline.
Generated Code
Console
Results of running class SmokeTest
No consoles to display at this time.
Writable
Insert
4 : 18 : 102

Projektplanung

Was ist zu tun?

03. Juni 2021
- ✓Abgabe des Exposés

07. Juni 2021
- ✓Anmeldung der Arbeit

Bis 30. Juni 2021
- ✓Endgültige Gliederung
- ✓1. Termin mit Prüfer

Bis 31. August
- ✓Dummyprogramm implementieren
- ✓Tests durchführen
- ✓2. Termin mit Prüfer

Anfang September
- ✓Bugfix und evtl. Features
- ✓3. Termin mit Prüfer

12. September 2021
- ✓Rohfassung der schriftl. Arbeit

15. September 2021
- ✓Lektorat und Verbesserungen einarbeiten

23. September 2021
- ✓Abgabe der Arbeit

Mitte Oktober 2021
- ✓Festlegen der Inhalte des Kolloquiums
- ✓4. Termin mit Prüfer

Referenzen

- Amalfitano, D., Fasolino, A. R. and Tramontana, P. (2011) „Using Dynamic Analysis for Generating End User Documentation for Web 2.0“
- M. Descher, T. Feilhauer, L. Amann (2014) „Automated user documentation generation based on the Eclipse application model“
- Oliveira, A. d. S. (2017) „GuideAutomator: Automated User Manual Generation with Markdown“
- Gök, O., Ersoy, P., Börühan, G. (2019) „The effect of user manual quality on customer satisfaction: the mediating effect of perceived product quality”

Vielen Dank!