Nina Weber

Freie Universität Berlin

Einstiegserleichterung für die Weiterentwicklung und Erweiterung der JavaScript- und HTML-GUI von Saros

Abschlussvortrag Bachelorarbeit

Gliederung

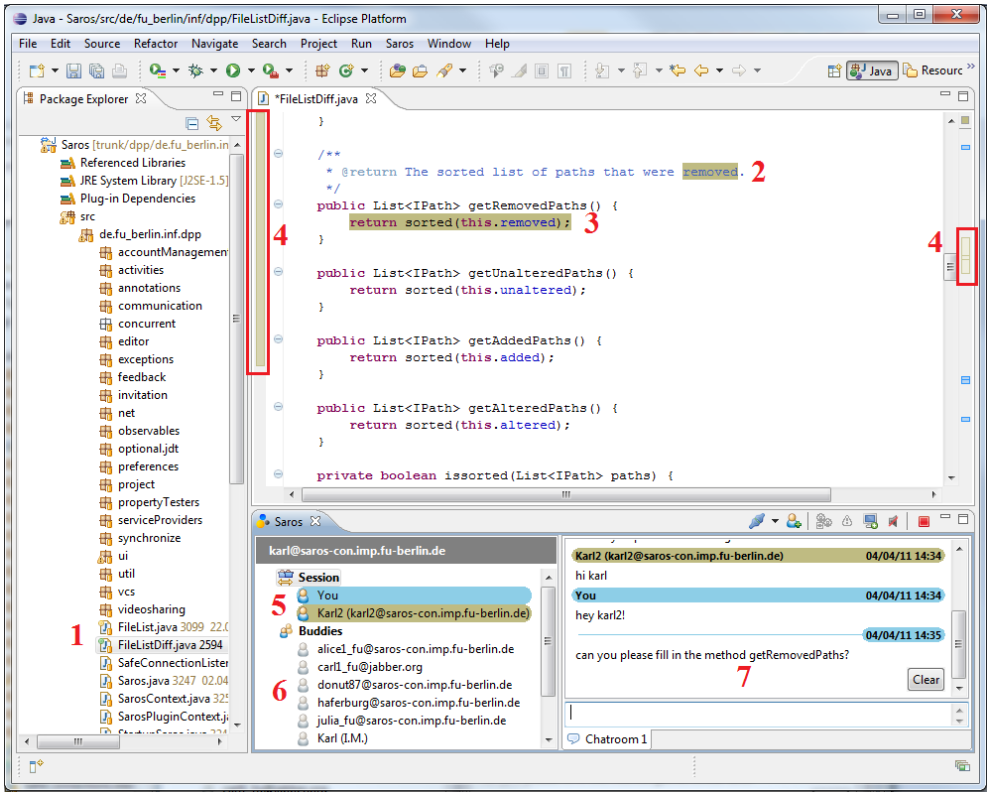
1. Einführung
   - Kontext
   - Aufgabenstellung
   - Überblick
2. Hauptteil
   - JavaScript-Framework
   - Build-Skript
3. Fazit

Freie Universität Berlin

EINFÜHRUNG

Kontext
Saros

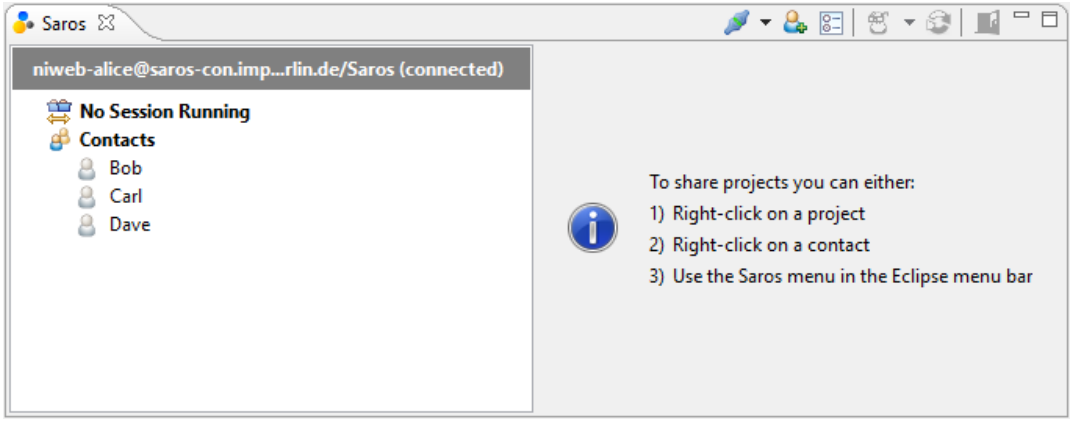
- Plug-in für Eclipse
- AG Software Engineering
- Gemeinsame Software-Entwicklung
  - Verteilte Paarprogrammierung



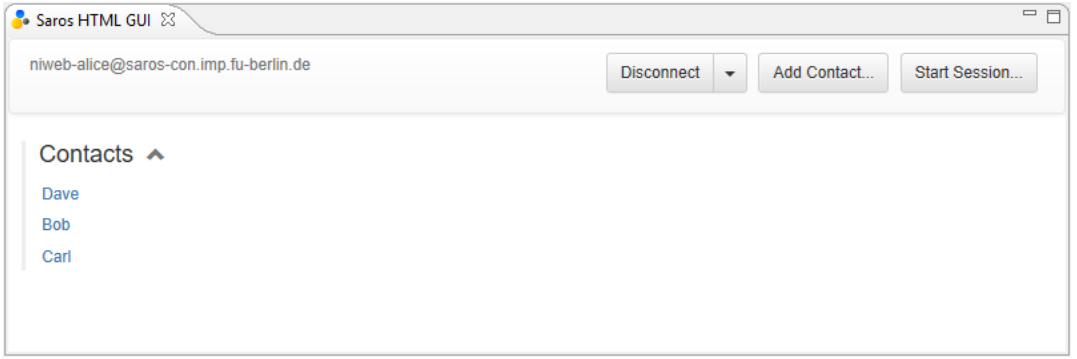
Quelle: http://www.saros-project.org/screenshots, 09.11.2016

Kontext

- Saros für weitere IDEs
- IntelliJ IDEA
  - Nutzt Swing, nicht SWT
  - DRY-Prinzip



- Browser einbetten
  - HTML/CSS/JS
- Bereits begonnen



Aufgabenstellung

- Nutzerorientierte Erweiterung der „HTML-GUI“
- Erleichterung des Einstiegs für nachfolgende Entwickler

Freie Universität Berlin

Überblick

- Einstiegserleichterungen
  - JavaScript-Framework
  - Ordnerstruktur
  - Build-Skript
  - Jade-Templates

- Erweiterungen
  - StartSessionWizard
  - JoinSessionWizard

Freie Universität Berlin

HAUPTTEIL

JavaScript-Framework
Ampersand.js

- Wenig Dokumentation
- Kaum Lernmaterialien
- Kleine Community

  ⇒ Schwer Erlernbar ⇒ Große Hürde

JavaScript-Framework
Alternativen

- AngularJS
  - Viel Material
  - Große Community
- Angular 2
  - Angular 1 bald veraltet
  - Großes Werkzeug, kleines Problem
- Reines JS
  - Keine Framework-Probleme
  - Viel selbst zu bauen
- Ampersand.js
  - Baukastensystem
  - Nah an reinem JS
  - Codedokumentation

Build-Skript
Erklärung

- Production Build
- Lange Ausführungszeit
  - Nach jeder Änderung

  ⇒ Verzögert Einarbeitung

```
$ time npm run build

> @ prebuild d:\Saros\saros\de.fu_berlin.inf.dpp.ui.frontend\html
> mkdirp bundle dist/bundle/ dist/css dist/fonts


> @ build d:\Saros\saros\de.fu_berlin.inf.dpp.ui.frontend\html
> npm run build:jade -s && npm run build:js -s


> @ postbuild d:\Saros\saros\de.fu_berlin.inf.dpp.ui.frontend\html
> cp *.html dist/ && cp css/*.* dist/css/ && cp fonts/*.* dist/fonts/ && cp bundle/bundle.js dist/bundle/bundle.js


real    0m19.379s
```

Build-Skript
Aufbau

1. **prebuild**
   - Erstellung diverser Ordner (*bundle*, *dist/bundle*, *dist/css*, *dist/fonts*)
2. **build**
   - **build:jade**
     - Erstellt *template.js* aus allen Templates
   - **build:js**
     - Erstellt *bundle.js* aus allen JS-Dateien
3. **postbuild**
   - Kopiert HTML, CSS, Fonts, Bilder & *bundle.js* in *dist*-Verzeichnis

Build-Skript
Optimierungen

- Auslagerung
  - *createdist* und *copysrcs*
  - Ausführung wenn benötigt

- Build
  - ~~"build:jade": "**templatizer** -d templates/ -o src/templates.js"~~
  - "build:js": "**browserify** -d src/app.js"
  - jadeify: -t **jadeify**
  - watchify: „watch mode for browserify“
    - Automatische Rekompilation
  - "watch": "**watchify** -d -t jadeify src/app.js -o dist/bundle/bundle.js"
  - "prewatch": "npm run *createdist* && npm run *copysrcs*"

Build-Skript
Ergebnis

```
$ npm run watch

> @ prewatch d:\Saros\saros\de.fu_berlin.inf.dpp.ui.frontend\html
> npm run createdist && npm run copysrcs


> @ createdist d:\Saros\saros\de.fu_berlin.inf.dpp.ui.frontend\html
> mkdirp dist/bundle/ dist/css dist/fonts dist/images


> @ copysrcs d:\Saros\saros\de.fu_berlin.inf.dpp.ui.frontend\html
> cp src/pages/*/*.html dist/ && cp src/css/*.* dist/css/ && cp src/fonts/*.* dist/fonts/ && cp src/images/*.* dist/images/


> @ watch d:\Saros\saros\de.fu_berlin.inf.dpp.ui.frontend\html
> watchify -v -d -t hbsfy src/app.js -o dist/bundle/bundle.js

3648005 bytes written to dist/bundle/bundle.js (5.26 seconds)
3648057 bytes written to dist/bundle/bundle.js (0.58 seconds)
3647975 bytes written to dist/bundle/bundle.js (0.64 seconds)
3648057 bytes written to dist/bundle/bundle.js (0.63 seconds)
3647975 bytes written to dist/bundle/bundle.js (0.63 seconds)
```

Freie Universität Berlin

FAZIT

Freie Universität Berlin

DANKE