Freie Universität Berlin

Erweiterung des Saros Test Frameworks für die HTML-GUI

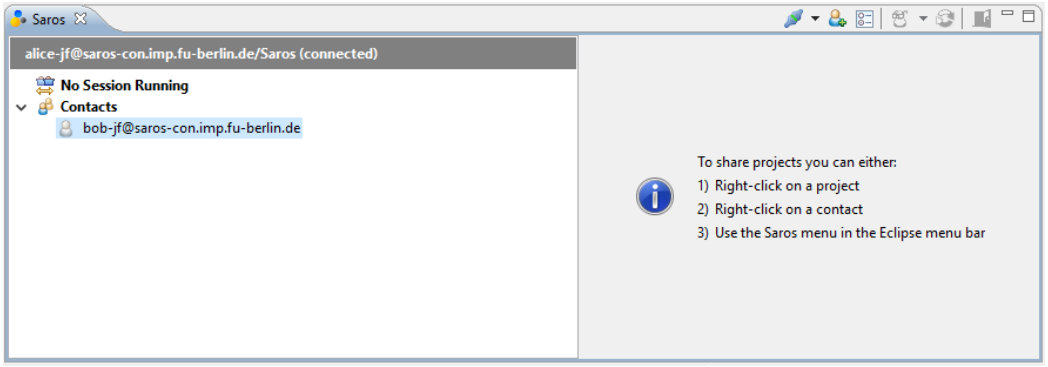
Masterarbeit

Jannis Fey

17.12.18

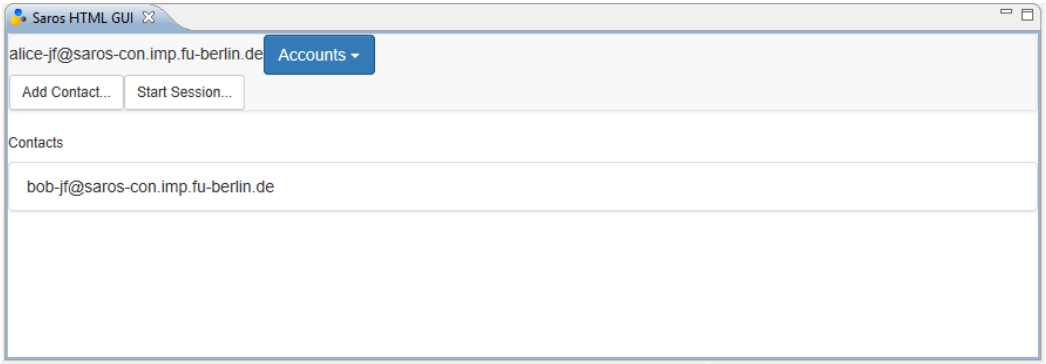
Was ist Saros?

- Plugin für Eclipse & IntelliJ
- Ermöglicht verteilte Paarprogrammierung in Echtzeit



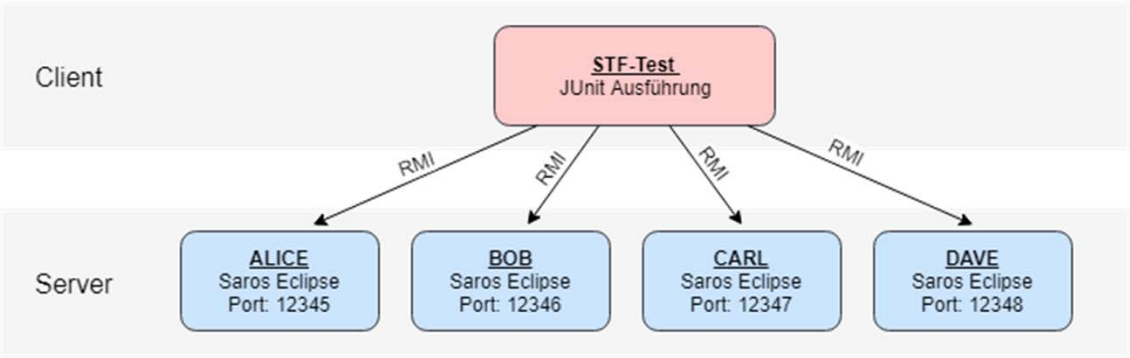
HTML-GUI

- IDE unabhängig
- In React geschrieben
- Wird in einem eingebetteten Browser angezeigt



Saros Test Framework (STF)

- Eigenes Framework um Saros zu testen
- Ermöglicht Ende-zu-Ende Test
- Interaktion mit HTML-GUI war bisher nicht möglich



Ziele

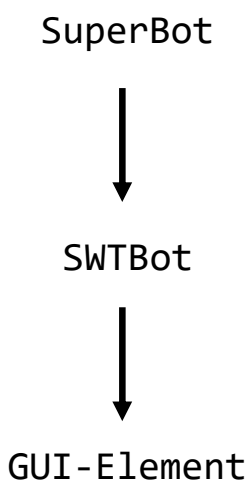
1. Interaktion mit HTML-Elemente
2. Modularisierung
3. Bestehende STF-Tests wiederverwenden
4. STF für IntelliJ starten

1

Interaktion mit HTML-Elementen

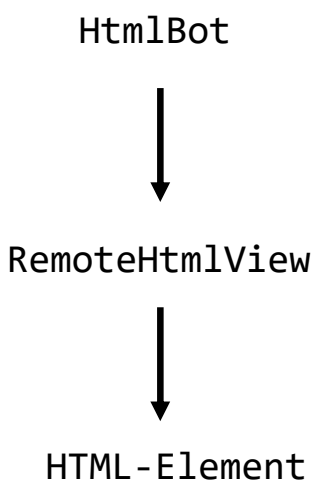
STF-Test bisher

```
@Test
public void addContact() throws Exception {
    Util.removeTestersFromContactList(ALICE, BOB);

    assertFalse(ALICE.superBot().views().sarosView().isInContactList(BOB.getJID()));
    assertFalse(BOB.superBot().views().sarosView().isInContactList(ALICE.getJID()));

    ALICE.superBot().views().sarosView().addContact(BOB.getJID());

    BOB.superBot().confirmShellRequestOfSubscriptionReceived();
    ALICE.superBot().confirmShellRequestOfSubscriptionReceived();

    assertTrue(ALICE.superBot().views().sarosView().isInContactList(BOB.getJID()));
    assertTrue(BOB.superBot().views().sarosView().isInContactList(ALICE.getJID()));
}
```



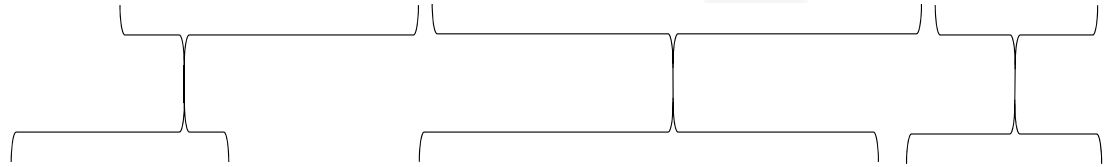
STF-Test für die HTML-GUI

```java
@Test
public void addContactViaHTML() throws Exception {
    assertFalse(ALICE.htmlBot().getContactList(MAIN_VIEW).contains(BOB.getBaseJid()));
    assertFalse(BOB.htmlBot().getContactList(MAIN_VIEW).contains(ALICE.getBaseJid()));

    ALICE.htmlBot().view(MAIN_VIEW).button( id: "add-contact").click();
    ALICE.htmlBot().view(ADD_CONTACT).inputField( name: "jid").enter(BOB.getBaseJid());
    ALICE.htmlBot().view(ADD_CONTACT).button( id: "add-contact").click();

    BOB.superBot().confirmShellRequestOfSubscriptionReceived();
    ALICE.superBot().confirmShellRequestOfSubscriptionReceived();

    assertTrue(ALICE.htmlBot().getContactList(MAIN_VIEW).contains(BOB.getBaseJid()));
    assertTrue(BOB.htmlBot().getContactList(MAIN_VIEW).contains(ALICE.getBaseJid()));
}
```



Im Detail

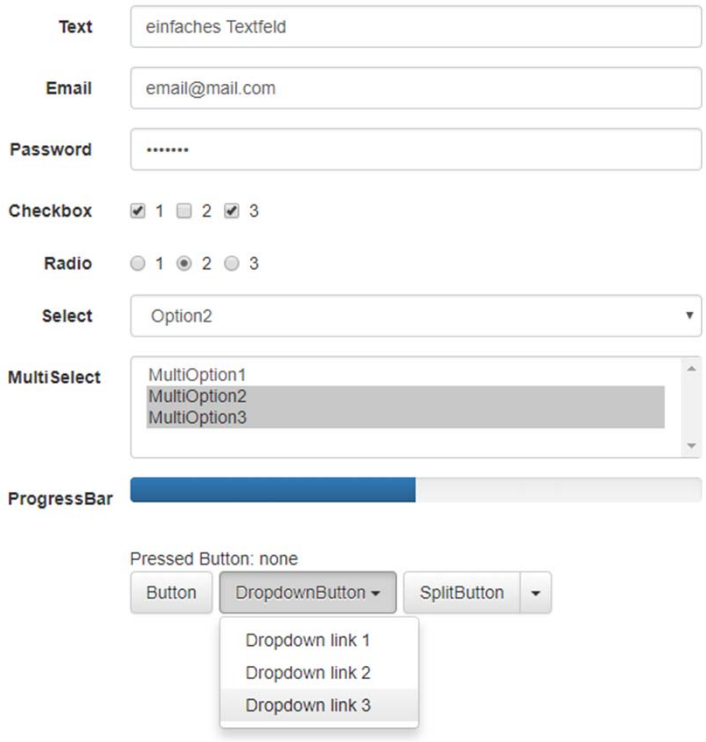
```
ALICE.htmlBot().view(MAIN_VIEW).button( id: "add-contact").click();
```



```
browser.run( “$(‘add-contact‘).click()“ );
```

Durch eine Java-Fluent-API wird Javascipt-Code zusammengesetzt, welcher dann in dem eingebetteten Browser in Eclipse ausgeführt wird.

BasicWidgetTestView

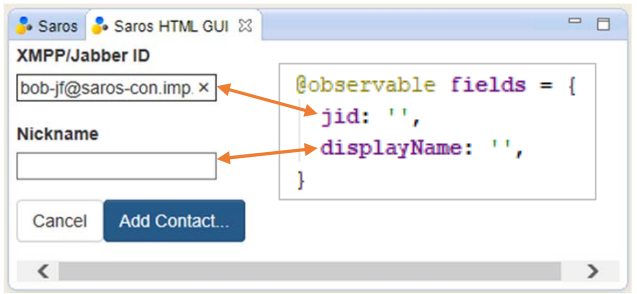



```java
@Test
public void shouldTestInputFields() throws Exception {
    IRemoteHTMLView view = ALICE.htmlBot().view(BASIC_WIDGET_TEST);

    assertTrue(view.hasElementWithName("text"));
    view.inputField( name: "text").enter( text: "Some random text");
    assertTrue(view.inputField( name: "text")
        .getValue().equals("Some random text"));
    view.inputField( name: "text").clear();
    assertTrue(view.inputField( name: "text")
        .getValue().equals(""));
}
```

- Dient nur für Testzwecke
- Nur über das STF erreichbar

Probleme

- Inhalt der Eingabefelder wird von React anders weiterarbeitet
  - Reactive Programmierung -> Observable
  - Einfaches schreiben in ein Feld reicht nicht aus
- Lösung: Direkter Zugriff auf React-Variable



- Selektoren nicht so nutzbar wie in Chrome
  - `$("button-id").click();`
  - `$("button-id")[0].click();`

Regelwerk für die HTML-GUI

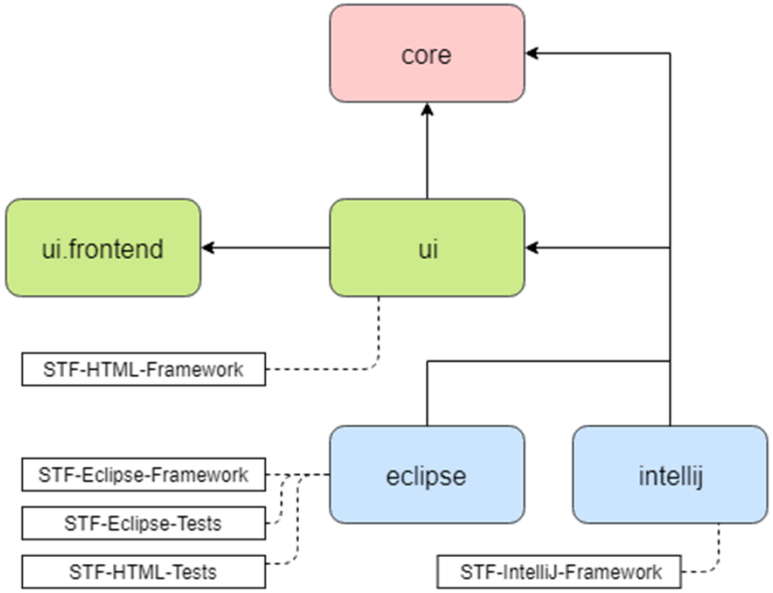
- Für jedes HTML-Eingabefeld wird eine React-Variable benötigt
  - Gleicher Name
  - OnChangeHandler
  - Getter & Setter

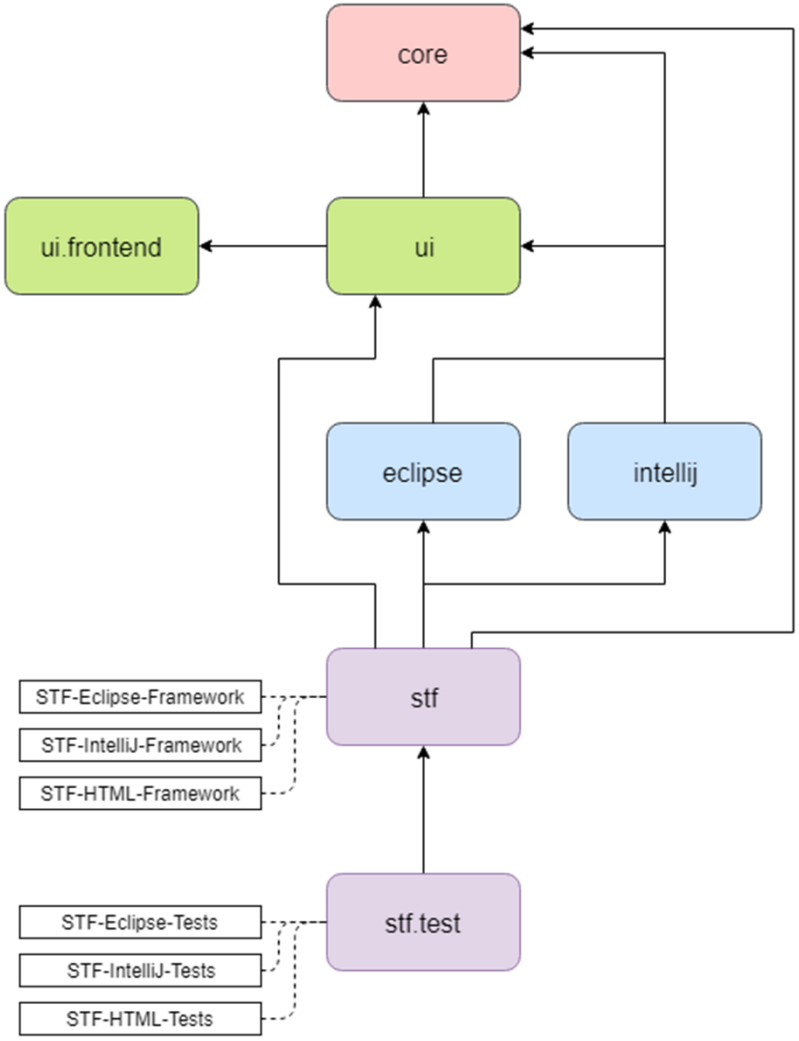
- BasicWidgetTestView dient als Vorlage für HTML-Elemente

```
@observable fields = {var1: '', var2: '', var3: ''};

@action onChangeField = (e) => {
    this.setFieldValue(e.target.name, e.target.value)
};

@action setFieldValue = (name, value) => {
    this.fields[name] = value
};

@action getFieldValue = (name) => {
    return this.fields[name]
};
```

2

Modularisierung



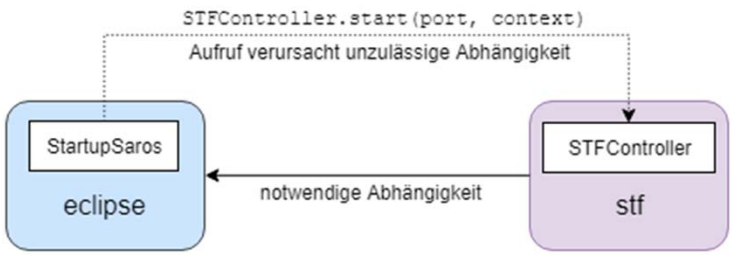


2 neue Komponenten

Problem: zyklische Abhängigkeit

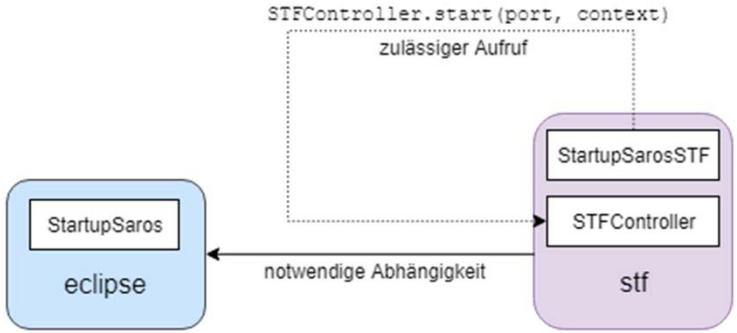
- STF-Komponente abhängig von Eclipse-Komponente
- Eclipse-Komponente benötigt STF-Komponente zum starten des STF

- Start des STF muss verlagert werden



Lösung: neues Plugin

- STF-Komponente ist ein eigenes Plugin
  - Eclipse-Plugin und gleichzeitig IntelliJ-Plugin
- STF wird von STF-Komponente gestartet

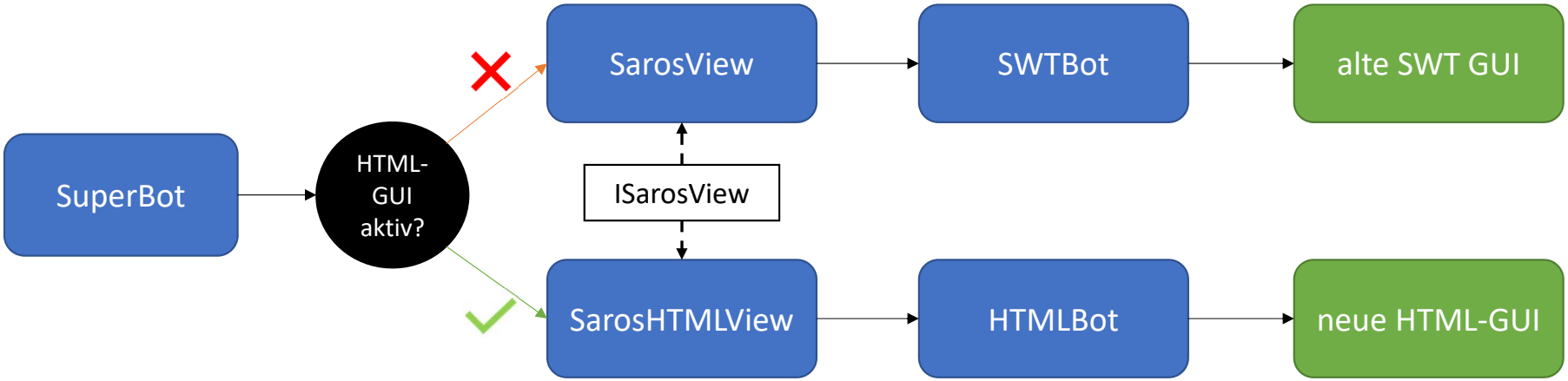


3

Bestehende STF-Tests wiederverwenden

Idee

- Bestehende Methoden auf neue HTML-GUI umleiten



Umsetzung

```java
@Override
public ISarosView sarosView() throws RemoteException {
    if (Saros.useHtmlGui()) {
        IHTMLBot htmlBot = HTMLBotImpl.getInstance();
        EclipseHTMLWorkbenchBot.getInstance().openSarosBrowserView();
        htmlBot.view(View.MAIN_VIEW).open();
        return SarosHTMLView.getInstance().setBot(htmlBot);
    } else {
        SWTWorkbenchBot bot = new SWTWorkbenchBot();
        RemoteWorkbenchBot.getInstance().openViewById(VIEW_SAROS_ID);
        bot.viewByTitle(VIEW_SAROS).show();
        return SarosView.getInstance().setView(bot.viewByTitle(VIEW_SAROS));
    }
}
```

HTML-GUI

SWT-GUI

Beispiel Aufruf: `ALICE.superBot().views().sarosView().connect();`

Ergebnis

- In der HTML-GUI fehlen viele Features von Saros
- Nur wenige Methoden des `ISarosView` Interface für HTML-GUI implementierbar
  - „Method is not yet implemented“

- Aber: Prinzipiell können alte STF-Test wiederverwendet werden um über die HTML-GUI zu testen

4

STF für IntelliJ starten

Wie geht es für Eclipse?

```java
public class StartupSarosSTF implements IStartup {

    private static final Logger LOG = Logger.getLogger(StartupSarosSTF.class);

    @Inject
    private IContainerContext context;

    public StartupSarosSTF() { SarosPluginContext.initComponent( instance: this); }

    @Override
    public void earlyStartup() {

        Integer port = Integer.getInteger("de.fu_berlin.inf.dpp.testmode");

        try {
            STFController.start(port, context);
        } catch (RemoteException e) {
            LOG.error( message: "starting STF controller failed", e);
        }
    }
}
```

Die Context Variable wird über Dependecy Injection (PicoContainer) geladen

Der Aufruf von `initComponent(this)` sorgt für die Initialisierung der Context Variable

Der Context wird zum starten des STF benötigt

Herangehensweise

- **Problem:** Das STF benötigt Zugriff auf den Context von Saros/I
- **Beispiel:** minimaler Nachbau der Problemstellung
- **Hilfe:** Problemstellung auf Stackoverflow und im IntelliJ-Forum
- **Experimentiert:** 5 verschiedene Varianten

5 Varianten ohne Erfolg!

- Variante 1: Kontext statisch laden
  - `IncompatibleClassChangeError`
  - Problem mit den ClassLoadern

```
IntellijProjectLifecycle.getInstance(project).getSarosContext();
```

- Variante 2: ServiceManager
  - IntelliJ-Platform-SDK
  - stellt sicher das nur eine Instanz existiert
  - PicoContainer in PicoContainer!

```
ServiceManager.getService(ContainerContext.class);
```

5 Varianten ohne Erfolg!

- Variante 3: ExtensionPoints
  - IntelliJ-Platform-SDK
  - Wenig Dokumentation
  - Konfiguration über XML war für Saros/I nicht möglich

- Variante 4: Neuer Lifecyle
  - Kontext erst in STF-Komponente erzeugen
  - Problem mit den ClassLoadern (wie Variante 1)

5 Varianten ohne Erfolg!

- Variante 5: Nur BrowserManager laden
  - Nur BrowserManager für HTML-STF-Tests notwendig
  - Wie in Variante 2 über ServiceManager laden
  - Problem: Neue Instanz des BrowserManagers wird erzeugt

```
ServiceManager.getService(BrowserManager.class);
```

- **Ergebniss:** Keine der 5 Varianten funktionierte
  - ➢ keine neuen Ideen/Ansätze mehr

Fazit

Was lief schlecht?

- Ziel 4 konnte nicht erreicht werden 🙁
- Zu viel Zeit in das IntelliJ-Problem investiert -> ohne Erfolg
- Dadurch Vernachlässigung von anderen Baustellen
- Offene Pull Requests für den Master aus den Augen verloren

Was lief gut?

- HTML-Elemente können über das STF angesteuert werden
- Bestehende STF-Test können wiederverwendet werden
- Gute Zusammenarbeit im Team (mir wurde immer geholfen)
- Ich habe viel Neues (kennen)gelernt
  - React, RMI, Dependecy Injection, PicoContainer, Eclipse/IntelliJ-Plugins, Arbeiten im Team mit Pull-Requests / Gerrit Reviewsystem

Vielen Dank

Jetzt ist Zeit für Fragen und Feedback