

Projekt: Kontakte

Vorlesung: Aktuelle Themen der Dienstleistungsinformatik

Markus Marzotko, Thorben Seeland, Dominic Wirkner

Prof. Dr. Bernhard Steffen, Dipl.-Inf. Markus Doedt

Technische Universität Dortmund

22. Januar 2013

- 1 Das Projekt
- 2 SAP Connector
- 3 Google Connector
- 4 SIB-Programmierung
- 5 jABC-Modell

- 1 Das Projekt
- 2 SAP Connector
- 3 Google Connector
- 4 SIB-Programmierung
- 5 jABC-Modell

Orchestrierung von Webservices



Die Aufgabenstellung

- **Datenmigration:** Kontakte von SAP zu Google übertragen
- **Strategie:** nicht einfach alle kopieren → Karteileichen
 - Kontakte nur bei Bedarf übertragen!
 - zunächst bei Google anfragen
 - erst dann Migration von SAP
 - sonst manuelle Eingabe
- **erste Überlegungen:**
 - Aufteilung in drei Bereiche: Google, SAP, SIBs
 - GUI Elemente für Nutzereingabe notwendig

Die Aufgabenstellung

- **Datenmigration:** Kontakte von SAP zu Google übertragen
- **Strategie:** nicht einfach alle kopieren → Karteileichen
 - Kontakte nur bei Bedarf übertragen!
 - zunächst bei Google anfragen
 - erst dann Migration von SAP
 - sonst manuelle Eingabe
- **erste Überlegungen:**
 - Aufteilung in drei Bereiche: Google, SAP, SIBs
 - GUI Elemente für Nutzereingabe notwendig

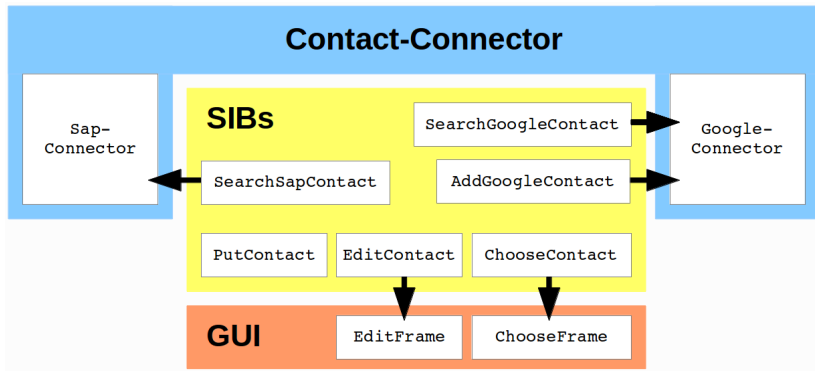
Die Aufgabenstellung

- **Datenmigration:** Kontakte von SAP zu Google übertragen
- **Strategie:** nicht einfach alle kopieren → Karteileichen
 - Kontakte nur bei Bedarf übertragen!
 - 1 zunächst bei Google anfragen
 - 2 erst dann Migration von SAP
 - 3 sonst manuelle Eingabe
- **erste Überlegungen:**
 - Aufteilung in drei Bereiche: Google, SAP, SIBs
 - GUI Elemente für Nutzereingabe notwendig

Die Aufgabenstellung

- **Datenmigration:** Kontakte von SAP zu Google übertragen
- **Strategie:** nicht einfach alle kopieren → Karteileichen
 - Kontakte nur bei Bedarf übertragen!
 - 1 zunächst bei Google anfragen
 - 2 erst dann Migration von SAP
 - 3 sonst manuelle Eingabe
- **erste Überlegungen:**
 - Aufteilung in drei Bereiche: Google, SAP, SIBs
 - GUI Elemente für Nutzereingabe notwendig

Projekt-Struktur



Eingesetzte Tools

- **Grundlage:** automatisierte Tests (JUnit), Versionsverwaltung (git)
- **Apache Maven:** Build-Management-Tool
 - unterstützt Software-Lebenszyklus → automatische Ausführung der Tests
 - einfaches Einbinden von Abhängigkeiten (externe Pakete)
 - verpackt alle Komponenten zu einer JAR-Datei → jABC-Projekt

Eingesetzte Tools

- **Grundlage:** automatisierte Tests (JUnit), Versionsverwaltung (git)
- **Apache Maven:** Build-Management-Tool
 - unterstützt Software-Lebenszyklus → automatische Ausführung der Tests
 - einfaches Einbinden von Abhängigkeiten (externe Pakete)
 - verpackt alle Komponenten zu einer JAR-Datei → jABC-Projekt

- 1 Das Projekt
- 2 SAP Connector**
- 3 Google Connector
- 4 SIB-Programmierung
- 5 jABC-Modell

Aufgabe

- erhalte Kontaktobjekt mit Angaben zu Typ, Vorname, Nachname, Firma..
- filtere aus Datenbank entsprechende Datensätze
- gib aufbereitete Liste aller zutreffenden Kontakte zurück

Verwendete WSDLs

Lieferant

- Find Supplier by Name and Address
- Read Supplier Basic Data

Mitarbeiter

- Find Employee by Elements
- Find Employee Address by Employee

Kunde

- Find Customer by Elements

Programmablauf

- Art des Filterobjekts überprüfen und entsprechenden Webservice aufrufen
 - IDs auslesen und anderen Webservice für alle IDs (einzeln) aufrufen
 - Rückgabeobjekte auslesen und Daten geordnet zurückgeben
- Bei Kunde reicht ein Webserviceaufruf

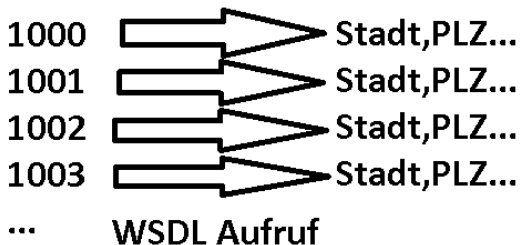
Lieferant, Kunde und Mitarbeiter: 3 unterschiedliche Welten

- Lieferant, Kunde und Mitarbeiter verwenden völlig unterschiedliche Klassen
 - lediglich die Übergabe von Passwort und Username ist gleich
 - Employee kommt aus dem SAP Human Resources (HR) Bereich
- Selbst einfachste Zuweisungen verkommen hier zur Akkordarbeit

Problematik bei Lieferant und Kunde

- Webservices geben hier nur Liste von IDs und Namen zurück
- für Adressinformationen weiterer Aufruf mit anderem Webservice nötig
- **PROBLEM:** Aufruf geschieht für jede ID einzeln

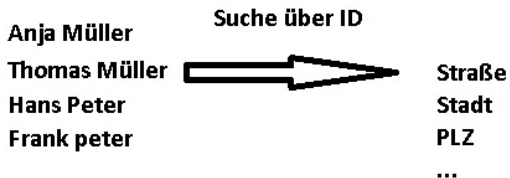
Problemdarstellung



Lösungsansatz

- GUI so erstellen, dass zunächst nur Name/Firma angezeigt werden
- Name/Firma sind bereits nach erstem Webservice Aufruf vorhanden
- erst nach Klick auf Namen werden Adressdaten via Webservice angefordert

Lösungsansatz

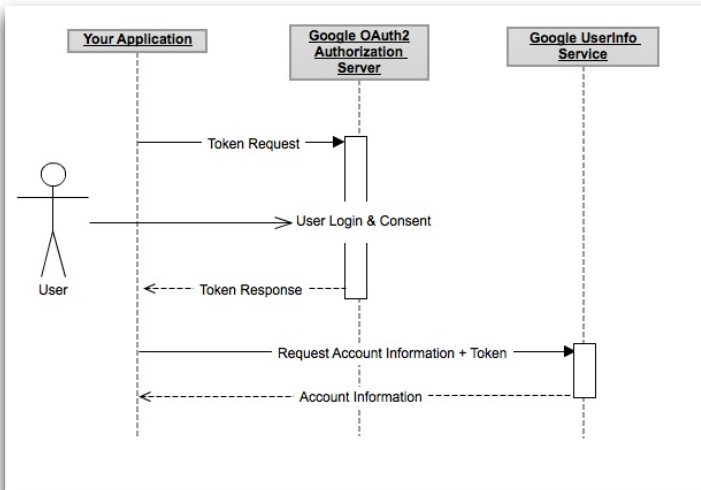


- 1 Das Projekt
- 2 SAP Connector
- 3 Google Connector**
- 4 SIB-Programmierung
- 5 jABC-Modell

Die *gdata*-Bibliothek

- frei verfügbar (s. <http://code.google.com/p/gdata-java-client/downloads/list>)
- benötigt Account (dli.ides.api@gmail.com)
- kapselt Google-Webservices komplett in *Java*-Klassen
- enthält alle benötigten Klassen und Pakete als JAR-Archive

Google-Service erstellen (1)



(Quelle: <https://developers.google.com/accounts/docs/OAuth2>)

Google-Service erstellen (2)

Username und Password

```
ContactsService myService;  
myService = new ContactsService(servicename);  
try {  
    myService.setUserCredentials(username, password);  
} catch (AuthenticationException e) {  
    e.printStackTrace();  
}
```


Einen Kontakt erstellen (1)

Kontakt-Objekt erstellen

```
// Create the entry to insert
ContactEntry contact = new ContactEntry();
contact.setTitle(new
    PlainTextConstruct(contactInfo.getFirstname()
+ contactInfo.getLastname()));
```

Einen Kontakt erstellen (2)

Namen in ein Kontakt-Objekt einfügen

```
// Name
Name name = new Name();
name.setFamilyName(new
    FamilyName(contactInfo.getLastname(), null));
name.setGivenName(new
    GivenName(contactInfo.getFirstname(), null));
contact.setName(name);
```

Einen Kontakt erstellen (3)

Benutzerdefinierte Einträge zu einem Kontakt-Objekt hinzufügen

```
// Firma
if (contactInfoCopy.getCompany() != null) {
    ExtendedProperty company = new ExtendedProperty();
    company.setName("Company");
    company.setValue(contactInfo.getCompany());
    contact.addExtendedProperty(company);
}
```

Einen Kontakt erstellen (4)

Das Kontakt-Objekt senden

```
// Kontakt senden
URL postUrl = new
    URL("https://www.google.com/m8/feeds/contacts/
dli.ides.api@gmail.com/full");
return myService.insert(postUrl, contact);
```

Kontakte suchen mit *Queries*

Kontakte suchen mit *Queries*

```
URL feedUrl = new
    URL("https://www.google.com/m8/feeds/contacts
/dli.ides.api@gmail.com/full");
Query myQuery = new Query(feedUrl);
ContactFeed resultFeed = null;
// Gruppe
myQuery.setStringCustomParameter("group",
    "http://www.google.com/m8/feeds/groups
/dli.ides.api%40gmail.com/base/587c880e884cdacb");
// submit request
resultFeed = myService.query(myQuery,
    ContactFeed.class);
```

Kontakte holen

Kontakte holen ohne *Queries*

```
URL feedUrl = new
    URL("https://www.google.com/m8/feeds/contacts
/dli.ides.api@gmail.com/full");
resultFeed = myService.getFeed(feedUrl,
    ContactFeed.class);
```

- 1 Das Projekt
- 2 SAP Connector
- 3 Google Connector
- 4 SIB-Programmierung**
- 5 jABC-Modell

SIB-Programmierung



Vorüberlegungen...

- **3 Sorten von SIBs:** Google, SAP, GUI
- **Google-SIBs:** Kontakt suchen und hinzufügen
- **SAP-SIB:** Kontakt suchen
- **GUI-SIBs:**
 - Eingabe von Kontakt-Attributen
 - Auswahl aus einer Liste von Kontakten

Vorüberlegungen...

- **3 Sorten von SIBs:** Google, SAP, GUI
- **Google-SIBs:** Kontakt suchen und hinzufügen
- **SAP-SIB:** Kontakt suchen
- **GUI-SIBs:**
 - Eingabe von Kontakt-Attributen
 - Auswahl aus einer Liste von Kontakten

Vorüberlegungen...

- **3 Sorten von SIBs:** Google, SAP, GUI
- **Google-SIBs:** Kontakt suchen und hinzufügen
- **SAP-SIB:** Kontakt suchen
- **GUI-SIBs:**
 - Eingabe von Kontakt-Attributen
 - Auswahl aus einer Liste von Kontakten

Google + SAP: suche Kontakt

- **Input:** eine Instanz der Klasse `Contact`
 - dient als Filter für die Suche
- **Output:** Liste von `Contact`-Objekten
- **Branches:**
 - **found:** mindestens ein Kontakt gefunden
 - **not found:** keine Ergebnisse
 - **error:** es wurde eine `Exception` geworfen

Google: Kontakt hinzufügen

- **Input:** eine Instanz der Klasse `Contact`
- **Output:** keiner
- **Branches:**
 - **default:** Kontakt erfolgreich hinzugefügt
 - **error:** es wurde eine `Exception` geworfen

GUI: Kontakt-Daten eingeben

- **Input:** eine Instanz der Klasse Contact
 - Formular wird entsprechend befüllt
 - zudem Parameter für Fenstertitel und Validierung der Eingabe
- **Output:** geänderte(!) Instanz der Klasse Contact
 - wenn Button „OK“ geklickt wurde
- **Branches:**
 - **ok:** Eingabe bestätigt mit Button „OK“
 - **cancel:** Eingabe abgebrochen mit Button „CANCEL“
 - **error:** es wurde eine `Exception` geworfen, oder `UNKNOWN`

GUI: Kontakt-Daten eingeben

- **Input:** eine Instanz der Klasse Contact
 - Formular wird entsprechend befüllt
 - zudem Parameter für Fenstertitel und Validierung der Eingabe
- **Output:** geänderte(!) Instanz der Klasse Contact
 - wenn Button „OK“ geklickt wurde
- **Branches:**
 - **ok:** Eingabe bestätigt mit Button „OK“
 - **cancel:** Eingabe abgebrochen mit Button „CANCEL“
 - **error:** es wurde eine `Exception` geworfen, oder UNKNOWN

GUI: Kontakt-Daten eingeben

- **Input:** eine Instanz der Klasse `Contact`
 - Formular wird entsprechend befüllt
 - zudem Parameter für Fenstertitel und Validierung der Eingabe
- **Output:** geänderte(!) Instanz der Klasse `Contact`
 - wenn Button „OK“ geklickt wurde
- **Branches:**
 - **ok:** Eingabe bestätigt mit Button „OK“
 - **cancel:** Eingabe abgebrochen mit Button „CANCEL“
 - **error:** es wurde eine `Exception` geworfen, oder `UNKNOWN`

GUI: Kontakt-auswählen

- **Input:** Liste von `Contact`-Objekten
 - zudem Parameter für Fenstertitel
- **Output:** eine Instanz der Klasse `Contact`
 - wenn Button „OK“ geklickt wurde
- **Branches:**
 - **ok:** Eingabe bestätigt mit Button „OK“
 - **cancel:** Eingabe abgebrochen mit Button „CANCEL“
 - **error:** es wurde eine `Exception` geworfen, oder `UNKNOWN`

Besonderheit der GUI-Programmierung

- **warten auf Eingabe:** wie `trace()`-Methode anhalten?
 - Swing-Frame läuft in einem eigenem Thread!
- **Lösung hier:** mittels `synchronized`-Block in Frame und SIB

Thread-Synchronisation

```
public String trace(ExecutionEnvironment env) {  
    ...  
    synchronized (frame) {  
        frame.wait();  
    }  
    ...  
}
```

- **Vorgehen:**
 - `trace()` erstellt den Frame und wartet auf ein `notify()`
 - `notify()` wird vom Action-Listener der Buttons aufgerufen
 - im Anschluss kann `trace()` die Eingabe im Frame abfragen

Besonderheit der GUI-Programmierung

- **warten auf Eingabe:** wie `trace()`-Methode anhalten?
 - Swing-Frame läuft in einem eigenem Thread!
- **Lösung hier:** mittels `synchronized`-Block in Frame und SIB

Thread-Synchronisation

```
public String trace(ExecutionEnvironment env) {  
    ...  
    synchronized (frame) {  
        frame.wait();  
    }  
    ...  
}
```

- **Vorgehen:**
 - `trace()` erstellt den Frame und wartet auf ein `notify()`
 - `notify()` wird vom Action-Listener der Buttons aufgerufen
 - im Anschluss kann `trace()` die Eingabe im Frame abfragen

Eigener Code im jABC

- **Unterschiede in der Laufzeitumgebung:**
 - Code läuft ausserhalb von jABC...
 - ABER: Ausführung des Graphen wirft `Exception`?
- **Lösung hier:** mittels „*bad Practice*“

Änderung von Systemeigenschaften

```
...
System.setProperty("javax.xml.parsers.SAXParser", ...);
System.setProperty("...parsers.SAXParserFactory", ...);
System.setProperty("oracle.xml.parser.v2.SAXParser", ...
...
```

- **Effekt:**
 - konfiguriert aktiv laufende JVM
 - auch nach Ausführung des Modells weiterhin wirksam
 - sollte in einem produktivem System nicht werden!

Eigener Code im jABC

- **Unterschiede in der Laufzeitumgebung:**
 - Code läuft ausserhalb von jABC...
 - ABER: Ausführung des Graphen wirft `Exception`?
- **Lösung hier:** mittels „*bad Practice*“

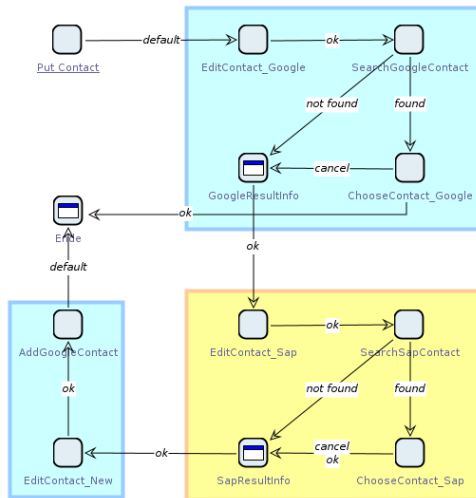
Änderung von Systemeigenschaften

```
...  
System.setProperty("javax.xml.parsers.SAXParser", ...);  
System.setProperty("...parsers.SAXParserFactory", ...);  
System.setProperty("oracle.xml.parser.v2.SAXParser", ...  
...
```

- **Effekt:**
 - konfiguriert aktiv laufende JVM
 - auch nach Ausführung des Modells weiterhin wirksam
 - sollte in einem produktivem System nicht werden!

- 1 Das Projekt
- 2 SAP Connector
- 3 Google Connector
- 4 SIB-Programmierung
- 5 jABC-Modell**

Der Modell-Graph im jABC



Vielen Dank für Ihre Aufmerksamkeit!

- 1 Das Projekt
- 2 SAP Connector
- 3 Google Connector
- 4 SIB-Programmierung
- 5 jABC-Modell