GTUG Bremen

Guice

Dependency Injection mit Google

Was ist Guice?

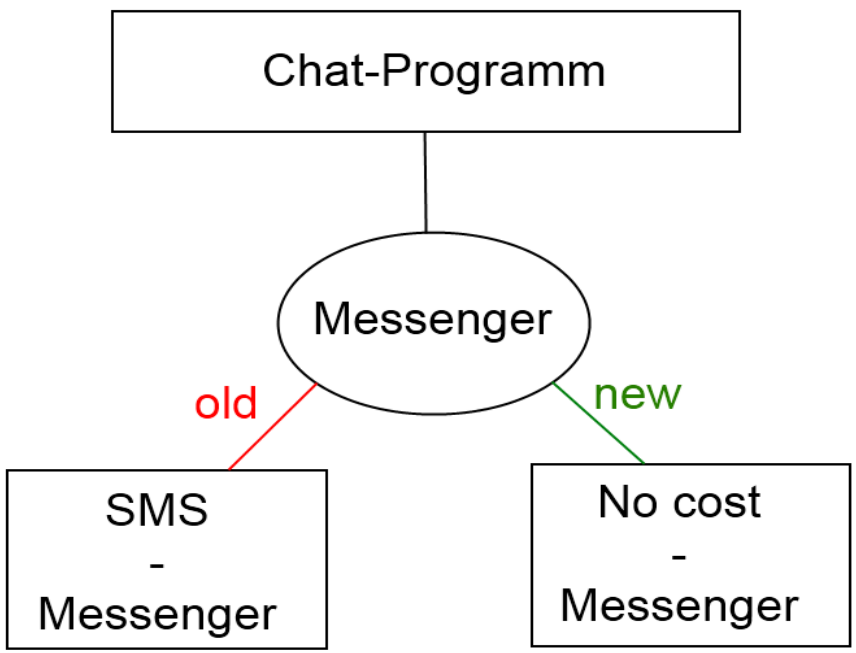
- Java Framework
- Entwickelt von Google
- Unter Apache Lizenz veröffentlicht
- Ermöglicht Dependency Injection
- *Erschienen:* 8. März 2007 in der Version 1.0
- *Aktuelle Version:* 3.0 seit 24. März 2011

Wozu brauch man Guice?

"Guice helps you design better APIs"
- Zitat von der Google Code Site von Guice

- Die Schnittstellenverwaltung wird einfacher
  - Durch Nutzung von Dependency Injection

- Beispiel:



Vorteile von Guice?

- Bei anderen Frameworks muss man für die Nutzung von Dependency Injection XML-Dateien erstellen
  - Mit Guice entfällt dies
- Schnittstellenverwaltung liegt zentral im Programm
- Detaillierte Fehlerausgabe
- Testen wird einfacher
  - Erstellung der Testumgebung ist einfacher

Ein Einblick

Verschiedene Injection-Arten

- Constructor Injection

```
private final Accent accent;

@Inject
public German(Accent accent) {
  this.accent = accent;
}
```

- Field Injection

```
@Inject
@Named("lang1")
private Language lang1;
```

- Method Injection

```
private Language lang1;

@Inject
public void setLanguage1(@Named("lang1") Language lang) {
  this.lang1 = lang;
}
```

Beeinflussung der Injection

- Nutzung von Modules
  - Werden mit Guice.createInjection(Modules ...) in Guice einbezogen
  - In einem Module muss in der Methode configure die Klassen gebunden werden (bind.to)

- Nutzung von Providern
  - Umfassendere Bindings können in Providern ausgelagert werden

Scopes

- Unscoped
  - In Tutorial ausschließlich genutzt
  - Für jeden Aufruf wird ein neues Objekt erstellt
- Singleton
  - Eine Instanz des Objekts in der gesamten Applikation
- SessionScoped
  - Eine Instanz pro Session z.B. HTML-Session
- RequestScoped
  - Eine Instanz pro Request z.B. HTML-Request
- Eigenen Scope mit CustomScopes erstellen
  - Wird nicht empfohlen

Wie nutze ich Scopes?

- Mit Annotation (z.B. @Singleton) über der Klasse
- Oder durch Binding im Module
  - bind(myAPI.class).to(myImpl.class).in(Singleton.class)
  - Scopes sind auf das Binding und nicht auf die Quelle bezogen
    - bind(myAPI1.class).to(myImpl.class).in(Singleton.class)
    - bind(myAPI2.class).to(myImpl.class).in(Singleton.class)
- Oder mit der Annotation @Provides über Methode in Module
  - Beispiel:
    - @Provides @Singleton
      BremenAccent provideBremenAccent(){[..]}

Was gibt es noch?

- Anstatt @Named können auch eigene Annotationen genutzt werden
  - bind(myAPI.class).
  - annotatedWith(myAnnotation.class).to(myImpl.class)
- Konstanten werden gecasted
  - bindConstant().annotatedWith(myAnnotation.class).to("4")
- u.v.m

Wo kann ich Guice nutzen?

- Java-Anwendung
  - Plattformunabhängig
- Google AppEngine
- Struts 2
- Java Persistence API
- Android
  - RoboGuice

Fragen?

Quellen

- Guice auf Google Code
  - http://code.google.com/p/google-guice/
- Google Tech Talk (26.04.2007)
  - http://www.youtube.com/watch?v=8RGhT-YySDY
- Roboguice auf Google Code
  - http://code.google.com/p/roboguice/