Zope3 fuer Entwickler

...oder: *“besseres Leben durch Komponenten, ein Tutorial fuer Zope3-Einsteiger mit Zope2-Erfahrung”*

Peter Sabaini, peter@sabaini.at

Warum Zope3?

- Leichtere Programmierbarkeit
- Bessere Wiederverwendbarkeit
- Einfacherer Code
- Bessere Dokumentation
- Mehr (Unit)Tests

Gib mir mein Zope3

- Checkout Zope3 Source, build:

```
$ svn co \
    svn://svn.zope.org/repos/main/Zope3/trunk \
    Zope3
$ cd Zope3
$ make
$ cp sample_principals.zcml principals.zcml
```

- principals.zcml anpassen
- Start:

```
$ bin/zopectl
zopectl> fg
```

Hello, World!

Hello, World!

- helloworld package
- hello.py module
- configure.zcml, package-includes

Debugging Hello World

```
$ bin/zopectl
zopectl> debug
>>> root = debugger.root()
>>> myfolder = root['myfolder']
>>> from helloworld.hello import HelloWorld
>>> h = HelloWorld()
>>> h.greeting = "Hello again!"
>>> myfolder['again'] = h
>>> print myfolder['again'].greeting
Hello again!
>>> import transaction
>>> transaction.commit()
>>> ^D
zopectl>
```

Zope3 Component Architecture

Zope3 Component Architecture

- Komponenten sind zentraler Bestandteil von Zope3
- Komponenten sind Objekte, die mittels Interfaces an andere Objekte angekoppelt werden koennen
- Bekannte Architekturen: Corba, KDE KParts, Mozilla API, ...

Component Architecture: Interfaces

- Definiert API
- Klasse mit API-Methoden/Attributen
- Deklaration und Dokumentation

Component Architecture: Schemas

- Erweiterung des Interface-Konzepts
- Deklariert ein Datenmodell
- Interface mit Attributen

Component Architecture: Adapter

- aus A mach B mach C ...
  - Objekt a wird ueber ein Interface zu einem Objekt b umgemodelt
- Eigene Klassen, werden global registriert
- Interfaces koennen Objekte adaptieren:
  - foo_obj = IFoo(bar_obj)

Component Architecture: Services, Utilities

- statische Logik, zB. DB-Konnektor, Authentifizerung, Caching, ...
- Oft Singletons

Component Architecture: ZCML

- Konfigurationssprache
- XML-Dialekt
- Bindet Komponenten, “glue”
  - mittels Interfaces
  - zB. Content an Views
- Registriert Adapter, Views, ...

Model - View - Controller

- Model: Content Components
- View: Adapter, adaptieren Request und (zB.) Model zu Output
- Controller: Ablaufsteuerung – Utilities, Services etc.

Das ein wenig nuetzlichere Beispiel

- Ein MessageBoard
- Wieder eigenes Package
  - book/messageboard/

Design

- 2 Content-Objekte
  - MessageBoard
  - Message
  - Beide muessen als Container agieren

MessageBoard: Interface

- book/messageboard/interface.py

MessageBoard: Content Components

- book/messageboard/message.py
- book/messageboard/messageboard.py

MessageBoard: Content Component Registrierung

- book/messageboard/configure.zcml

MessageBoard: Views

- book/messageboard/browser/configure.zcml

MessageBoard: Registrieren / Konfigurieren

- package-includes/messageboard-configure.zcml

MessageBoard Demo

- im ZMI

Unit testing

Unit testing

- Softwarequalitaet durch automatisierte Tests
- Wesentlicher Bestandteil von Zope3
- Verwendet Python unittest Modul
- eigener Testrunner

Unit testing

- MessageBoard
  - messageboard/tests
  - DocTests
- Ausfuehren

```
$ cd ~/Zope3
$ python test.py -vpu -dir \
    src/book/messageboard
```

Weitere Views hinzufuegen

- Dzt. nur EditForm
  - simples Display fuer User ohne Editrechte
  - Template: browser/details.pt
    - Page Template
  - Support-Klasse: browser/message.py
    - stellt Displaylogik zur Verfuegung

View konfigurieren

- browser/configure.zcml
- <page ... /> Direktive
  - for: zu welchem Interface gehoert page
  - class: View-Klasse, die gebunden werden soll
  - template: Template, das aufgerufen werden soll
  - menu, title: Tab im ZMI

Weitere View: Thread

- View soll Message Threads anzeigen
  - Messages und Follow-ups in Baumdarstellung
- thread.Thread
  - listContentInfo(): geht rekursiv durch enthaltene Messages

MessageBoard, neue View

- Demo im ZMI

Zusammenfassung

- Zope3 als Framework fuer mittlere bis groesser Applikationen
  - Leichtere Programmierung
  - Bessere Codequalitaet
- Overhead Komponentenarchitektur
  - (sehr) kleine Applikationen profitieren kaum

Fragen, Credits

- Fragen? Antworten?
- Credits
  - Stephan Richter, Verfasser “Zope3 Developers Handbook”
  - Jim Fulton, Verfasser “Programmers Tutorial”