1 Merkblatt Django

1.1 Die Idee hinter django

Django wurde entwickelt mit dem Ziel, ein Framework zu schaffen, welches auf einfache Art verschiedenste Inhaltstypen einfach erstellen, verwalten und kombinieren lässt.

Es wurde anfangs vor allem für inhaltsreiche Webseiten wie zum Beispiel grossen Zeitungswebseiten eingesetzt.

Das Grundprinzip von Django ist DRY (Don't repeat yourself) sprich alles sollte wiederverwend- und veränderbar sein.

Der Aufbau folgt dem OOP Paradigma MVC (Model View Controler). Die Idee davon ist das alle Drei Komponenten einzeln austausch- und veränderbar sind. In Django wurde aber ein webbasierter Ansatz gewählt, was leider zu Verwirrungen führt. Da: Model = Model, View = Template und Controller = View

1.2 Ein Projekt

Jedes Django Projekt startet mit einem Projekt Ordner.

Dieser kann dynamisch erstellt werden mit Hilfe von:

```
# django-admin.py startprojekt <name>
```

Anschliessend gibt es folgende Dateien:

settings.py – alle globalen Settings des Projektes

urls.py – alle Applikationen bekommen innerhalb der Webseite einen Standort zugewiesen. Z.B. Das Blog kommt unter /blog/

1.2.1 Die Settings

Alle settings sind in Python geschrieben, es wurde auf Grund der Performance so wie der Einfachheit darauf verzichtet, ein zusätzliches Settingsformat zu verwenden.

1.2.1.1 Die wichtigsten Settings

1.2.1.1.1 INSTALLED_APPS

Hier werden alle internen wie externen Django - Applikationen in eine tuple definiert.

1.2.1.1.2 TEMPLATE_DIRS

Definiert ein tuple, welches alle Ordner mit Templates definiert. Im Normalfall empfiehlt sich nur ein Verzeichnis für Templates zu definieren

1.2.1.1.3 MEDIA_ROOT

Definiert den Ordner unter dem sich alle statischen Dateien befinden (css, javascript, Bilder, Flash usw.).

Z.B. <ProjektDir>/media/

1.2.1.1.4 MEDIA_URL

Definiert die weburl unter dem sich alle statischen Dateien befinden.

Z.B. /media/

1.2.1.1.5 ADMIN_MEDIA_PREFIX

Definiert die weburl unter dem sich alle statischen Dateien für das Admininterface befinden.

Z.B. /media/admin/

1.2.1.1.6 MIDDLEWARE_CLASSES

Definiert alle Middlewareklassen welche im Projekt integriert werden. Middlewareklassen, sind Klassen, welche an verschiedenen Punkten während dem Programmablauf, Werte verändern oder hinzufügen, bzw. zusätzliche Operationen ausführen wie zum Beispiel logging.

Weitere Informationen zu Middlewareklassen findet ihr unter:

http://docs.djangoproject.com/en/dev/topics/http/middleware/

1.2.1.2 Alle verfügbaren Djangosettings

Django bietet eine Vielfalt von weiteren Einstellungsmöglichkeiten. Alle sind dokumentiert unter:

http://docs.djangoproject.com/en/dev/ref/settings/

Es ist hierbei aber auch zu beachten das viele Applikationen auch die Projektsettings benutzten um Werte abzufragen, die nicht zu Django gehören, hierzu findet ihr mehr unter den jeweiligen Applikationsdokumentationen.

Z.B. Für django-tagging (http://code.google.com/p/django-tagging/)

findet ihr die settings unter: http://django-tagging.googlecode.com/svn/trunk/docs/overview.txt

1.2.2 Die Urls

Die Projekt urls.py Datei sollte eigentlich nur dazu benutzt werden, die Applikationen innerhalb deiner Webseite zu positionieren.

Es ist aber für Spezialfälle sehr wohl möglich direkt views einzubinden – siehe 1.3.2

Beispiel:

```
urlpatterns = patterns('',
    #(r'^agenda/', include('cal.urls')),
    #(r'^agenda/', include('eventcal.urls')),
    (r'^wiki/', include('djiki.urls')),
    (r'^schule/kurse', include('school.urls.courses')),
    (r'^tinymce/', include('tinymce.urls')),
    (r'^mini/', include('school.urls.my')),

    # Uncomment the admin/doc line below and add 'django.contrib.adm
    # to INSTALLED_APPS to enable admin documentation:
    (r'^admin/doc/', include('django.contrib.admindocs.urls')),
    (r'^admin/filebrowser/', include('filebrowser.urls')),

    # Uncomment the next line to enable the admin:
    (r'^grappelli/', include('grappelli.urls')),
    (r'^admin/stats/', include('django_statistics.urls')),
    (r'^admin/(.*)', admin.site.root),
    (r'^markitup/', include('markitup.urls')),
    (r'^accounts/', include('registration.urls')),
    (r'^mitteilungen/', include('notification.urls')),
    (r'^stats/', include('django_statistics.urls')),
    (r'^kommentare/', include('django.contrib.comments.urls')),
    (r'^', include('django.contrib.flatpages.urls'))
)
```

Um das include Statement zu verwenden musst du folgendes importieren:

```
from django.conf.urls.defaults import include
```

oder im Normalfall

```
from django.conf.urls.defaults import *
```

1.3 Eine Applikation

Eine Applikation unter django ist ein python module, welches für eine spezifische Funktionalität bereit stellt.

Z.B. Ein Blog, ein CMS, ein Forum usw.

Eine Applikation teilt sich in mehrere Bereiche auf:

model, view, urls, templates, admin, forms

1.3.1 Das Model

Ein Model definiert eine Datenstruktur in der Datei models.py.

Eine Applikation kann mehrere Modelle beinhalten.

Ein Model definiert eine bis mehrere Tabellen in einer Datebank.

Web: http://docs.djangoproject.com/en/dev/topics/db/models/

Die Felder eines Models werden mit Hilfe von sogenannten fields definiert. Fields haben wiederum verschiedene Eigenschaften.

Web: http://docs.djangoproject.com/en/dev/ref/models/fields/

Mit Hilfe der Models ist es möglich auf einfache Art und Weise Datenbankabfragen zu erstellen.

Beispiel:

```
from blog.models import Entry
entries = Entry.objects.all()
catentries = entries.filter(categorie="TestCat").filter(published=True)
```

Web: http://docs.djangoproject.com/en/dev/topics/db/queries/

1.3.2 Die Url

Die urls.py Datei definiert, welche View mit welchen Parameter aufgeführt wird wenn folgender Ausdruck auf die aufgerufene URL zutrifft.

Für kleinere nicht interaktive Applikation genügen häufig bereits die in Django standardmässig mitgelieferten generic_views, womit nur eine URL Definition notwendig ist.

Beispiel generic_views und eigene views vermischt:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'^$', 'django.views.generic.date_based.archive_index'),
    (r'^(?P<year>\d{4})/(?P<month>[a-z]{3})/$',
'django.views.generic.date_based.archive_month'),
    (r'^tag/(?P<tag>\w+)/$', 'weblog.views.tag'),
)
```

Generic View Dokumentation:

http://docs.djangoproject.com/en/dev/ref/generic-views/

URLS Dokumentation:

http://docs.djangoproject.com/en/dev/topics/http/urls/

1.3.3 Die View

Die Views werden in views.py definiert.

Eine View generiert aufgrund von Parametern eine Auswahl an Objekten und verwandelt damit ein Template für die danach folgende graphische Ausgabe. Wobei im Template das Design definiert und nicht die View. Die View definiert nur die zu Verfügung gestellten Daten.

Web: http://docs.djangoproject.com/en/dev/topics/http/views/

1.3.4 Templates

Ein Template generiert aus einer Vielzahl von Objekten eine HTML Seite. Django bietet hierfür eine eigene sehr umfangreiche und erweiterbare Templatesprache, welche das verwandeln von Objekten in HTML möglichst einfach machen sollte.

Die Templates werden von verschiedenen Orten geladen. Die meisten bereits vorhandenen Applikation bieten von Haus aus eigene Templates. Wenn man diese überschreiben will kopiert man diese am besten in sein im Projekt definiertem Templatedir und verändert diese nach seinen eigenen Bedürfnissen

Web: http://docs.djangoproject.com/en/dev/topics/templates/

1.3.5 Admin

In der optionalen Datei admin.py wird das Verhalten der Applikation im Admininteface definiert.

Web: http://docs.djangoproject.com/en/dev/ref/contrib/admin/

1.3.6 Forms

Häufig definieren Applikationen auch Forms, diese repräsentieren HTML Formulare und bieten zusätzlich die Möglichkeit die Daten zu validieren und direkt in ein definiertes Model zu speichern.

Von der Programmierung und dem Handling sind Forms sehr vergleichbar mit Models.

Web: http://docs.djangoproject.com/en/dev/topics/forms/