Technische Universität München

Python and Django – The elephant in the room (Part 2)

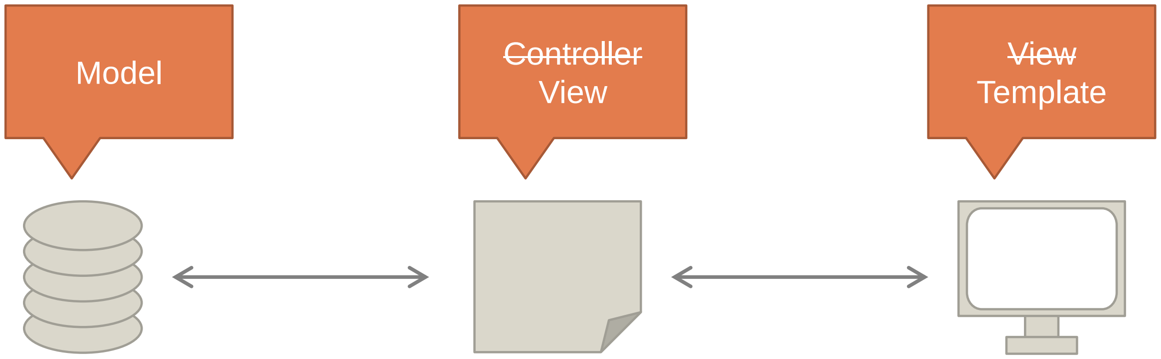
Janosch Maier, 09.04.2015, Room BC1 2.02.01

Software Engineering for Business Information Systems (sebis)
Department of Informatics
Technische Universität München, Germany

wwwmatthes.in.tum.de

What we know already?

Develop a basic application



Model

~~Controller~~
View

~~View~~
Template

How to proceed

- Clone the Application if you want…
- Take the app "webtech-skeleton"

https://github.com/Phylu/webtech-django2.git

Have a Look at some code…

Overview

1. **Object Relational Mapping**
   - Retrieving Objects
   - Filtering
   - Aggregation & Annotation

2. **Tests**
   - Unittests
   - Webserver Response Tests

3. **Middleware**
   - Messages
   - Sessions

4. **Translation (If still time)**
   - Defining Translation
   - Translation Files

Object Relational Mapping

ORM is a technique to map objects to a relational database.
Objects can be easily added, retrieved, changed and deleted.
Model is mapped to table. Attribute is mapped to column.

Backend Storage

```
sqlite> select * from
scorecard_course;

id l_id  course_title  votes
-  ----  ------------  -----
1  2     Webtech       5
2  1     Global Softw  1
3  1     Web Applicat  0
4  2     Gründung und  1
5  4     Software Eng  0
```

Access

```
c = Course.objects.get(pk=1)

c = get_object_or_404(Course, pk=1)

for c in Course.objects.all():
    print (c)
```

Object Relational Mapping

- Retrieving Objects:

```
Course.objects.get(pk=1)
Course.objects.all()
Course.objects.order_by('-votes')
```

- Filtering:

```
Course.objects.filter(votes__gte=42)

Lecturer.objects.filter(
       first_name__exact=F('last_name'))

Lecturer.objects.filter(
       Q(first_name__contains='Alex') |
       Q(last_name__contains='Shumaiev'))
```

Write some code...

Object Relational Mapping

- Aggregation:

  ```
  Course.objects.aggregate(Avg('votes'))[
         'votes__avg']
  ```

- Annotation:

  ```
  Course.objets.values('lecturer').
         annotate(avg_votes=Avg('votes'))
  ```

- Raw SQL: Check Documentation

Write some more code...

Overview

1. **Object Relational Mapping**
   - Retrieving Objects
   - Filtering
   - Aggregation & Annotation

2. **Tests**
   - Unittests
   - Webserver Response Tests

3. **Middleware**
   - Messages
   - Sessions

4. **Translation (If still time)**
   - Defining Translation
   - Translation Files

Testing

- As real software engineers, we want automated testing.

- Djangos test framework does:
  - Unittests
    - Direct Test of algorithmic
    - Mainly to test models and views
  - Client–Webserver Tests
    - Test how webserver answers requests
    - Mainly to test views and templates
  - Selenium Tests (not covered here)
    - Test how the page looks
    - Test integration of all parts

Testing

- Productive data is not touched

- Testing environment based on your needs:

```
def setUp(self):
  """ Create some courses & lecturers """
  pass
```

- Run: `./manage.py test`

Write even more code...

Overview

1. **Object Relational Mapping**
   - Retrieving Objects
   - Filtering
   - Aggregation & Annotation

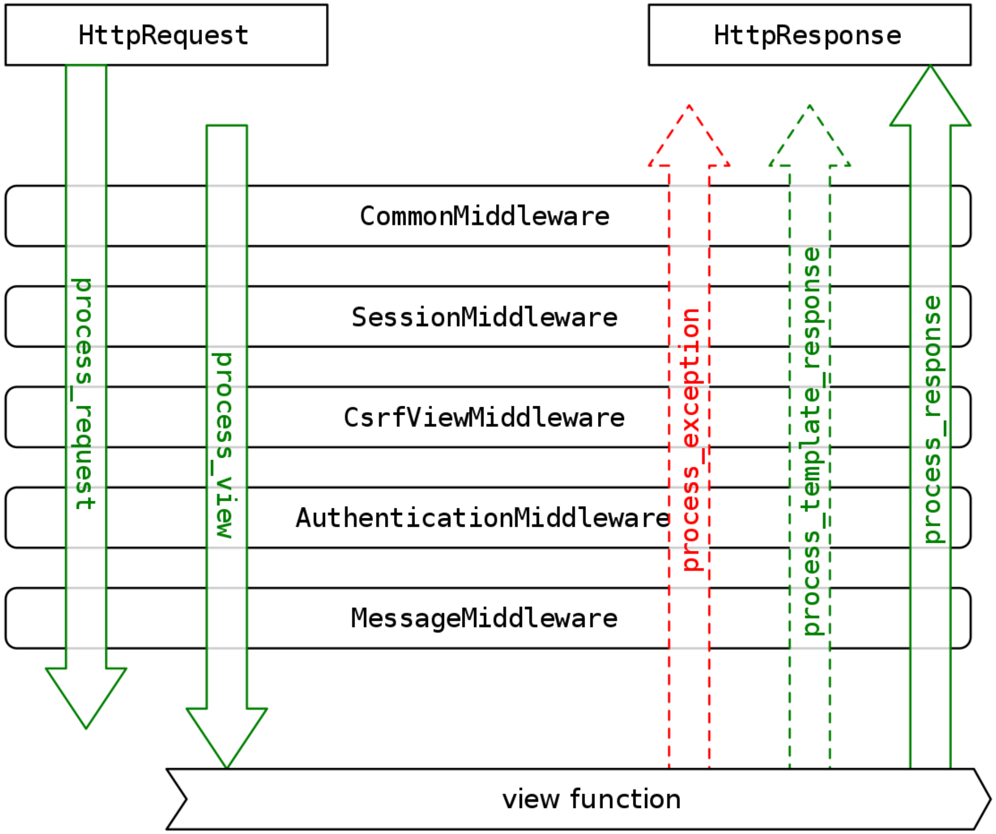
2. **Tests**
   - Unittests
   - Webserver Response Tests

3. **Middleware**
   - Messages
   - Sessions

4. **Translation (If still time)**
   - Defining Translation
   - Translation Files

Middlewares

- Middlewares hook in between HttpRequest/Response and Django view



Source: https://docs.djangoproject.com/en/1.8/topics/http/middleware/

Middlewares

- Middlewares add functionality to request handling:

- Auth:

```
if request.user.is_authenticated():
    …          # Do something for authenticated users.
else:
    ...        # Do something for anonymous users.
```

- Message:

```
messages.add_message(request, messages.ERROR,
       "You have already voted!")
```

- Session:

```
request.session['has_voted'] = True
request.session.get('has_voted', False)
```

Write code, again...

Middlewares

Want to write your own Middleware?

- Just implement (one of):
  process_request(request)
  process_view(request, view_func, view_args, view_kwargs)
  process_template_response(request, response)
  process_response(request, response)
  process_exception
- Then rigster your middleware in the config

Comparison

Django vs. other Web Frameworks

Pro Django

- It is Python! (e.g. easy readable)
- Very defensive (no standard null fields)
- Apps and namespaces

Contra Django

- Probably overloaded for small pages (defaults)

Thank you for your time!

Any Questions?

Janosch Maier
B.Sc.

Technische Universität München
Department of Informatics
Chair of Software Engineering for
Business Information Systems

Boltzmannstraße 3
85748 Garching bei München

maierj@in.tum.de
wwwmatthes.in.tum.de

Overview [We still have too much time…]

1. **Object Relational Mapping**
   - Retrieving Objects
   - Filtering
   - Aggregation & Annotation

2. **Tests**
   - Unittests
   - Webserver Response Tests

3. **Middleware**
   - Messages
   - Sessions

4. **Translation (If still time)**
   - Defining Translation
   - Translation Files

Translation

- Internationalizing your django application for an international audience? Define your translation strings

- Views:

```
from django.utils.translation import
       ugettext as _
...
messages.add_message(request, messages.ERROR,
       _("You have already voted!"))
```

- Templates:

```
{% load i18n %}
{% trans "Home" %}
{% blocktrans count counter=course.votes %}
       {{ counter }} vote.
{% plural %}
       {{ counter }} votes.
{% endblocktrans %}
```

Code the last example...

Translation

- Create Translation File:
  `django-admin.py makemessages -l de`

- This creates "/locale/de/LC_MESSAGES/django.py"

```
#: templates/scorecard/base.html:20
msgid "Home"
msgstr "Start"

#: templates/scorecard/index.html:10
#, python-format
msgid "%(counter)s vote."
msgid_plural "%(counter)s votes."
msgstr[0] "%(counter)s Stimme."
msgstr[1] "%(counter)s Stimmen."
```

Add some code…

Translation

- Compile the translation:

```
django-admin.py compilemessages
```

- Django determines the language by:
  - Cookie
  - Browser settings
  - Django application default language

Just a last command in the terminal…

Comparison

Django vs. other Web Frameworks

Pro Django

- It is Python! (e.g. easy readable)
- Very defensive (no standard null fields)
- Apps and namespaces

Contra Django

- Probably overloaded for small pages (defaults)

Thank you for your time!

Any Questions?

Janosch Maier
B.Sc.

Technische Universität München
Department of Informatics
Chair of Software Engineering for
Business Information Systems

Boltzmannstraße 3
85748 Garching bei München

maierj@in.tum.de
wwwmatthes.in.tum.de