Django in Enterprise World

PyCon4

9 May, 2010

Simone Federici

s.federici@gmail.com

Obiettivo

Django è sulla bocca di tutti, è un full stack framework che ha messo una pietra di storia nel mondo python, e non solo.

Ma perchè, in Italia, le grandi aziende per il web non riescono a uscire dal tunnel J2EE/JSF o, peggio, Struts?

In questo talk porto alla luce differenze sostanziali di approccio alle problematiche comuni di applicazioni web, e ad architetture più complesse Enterprise. Ma cosa significa Enterprise? Tutto ciò che i clienti definiscono Enterprise lo è veramente?

Come javista che pythonista, parlerò di cosa il mondo java rimprovera al mondo python e come ho scoperto che le accuse fanno acqua...

Agenda

- *Mini Django Overview (ma dovreste già conoscerlo)*

- *Enterprise World*
  - *Multi Tiers (client, web, business, EIS)*
  - *Conteiners, WebServices (SOA)*
  - *Transactions*

- *Development*
- *Deployments*
- *Scalability, Accessibility, and Manageability*

- *Q?*

Django mini overview (1)

```
class Reporter(models.Model):
    full_name = models.CharField(max_length=70)
    def __unicode__(self):
        return self.full_name

class Article(models.Model):
    pub_date = models.DateTimeField()
    headline = models.CharField(max_length=200)
    content = models.TextField()
    reporter = odels.ForeignKey(Reporter)

    def __unicode__(self):
        return self.headline
```

Class definitions

Field Validation

Model's Relations

```
manage.py syncdb
```

database syncronization

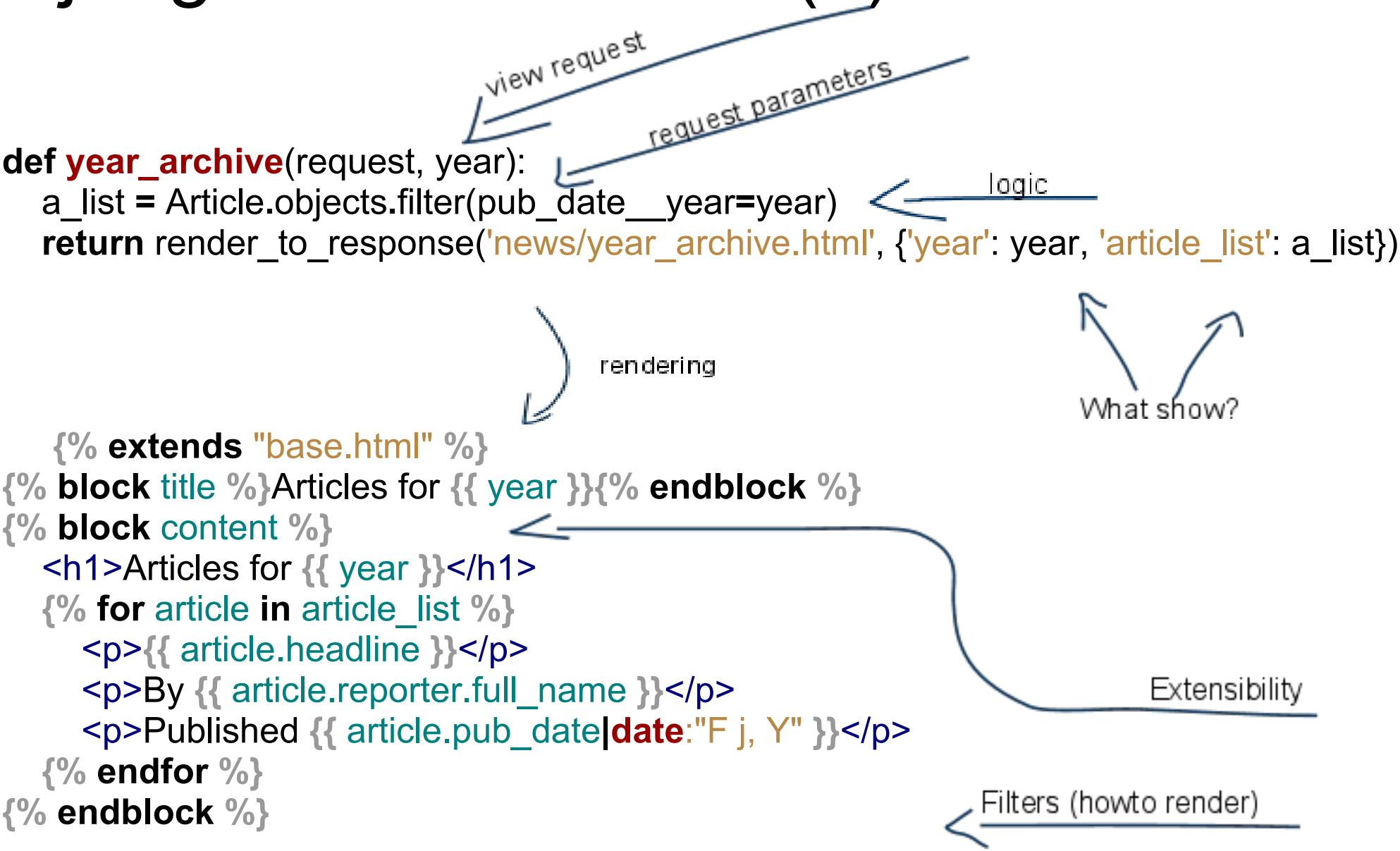
```
from django.conf.urls.defaults import *
urlpatterns = patterns('',
    (r'^articles/(\d{4})/$', 'mysite.views.year_archive'),
    (r'^articles/(\d{4})/(\d{2})/$', 'mysite.views.month_archive'),
    (r'^articles/(\d{4})/(\d{2})/(\d+)/$', 'mysite.views.article_detail'), )
```

Views urls

```
import models from django.contrib import admin
admin.site.register(models.Article)
```

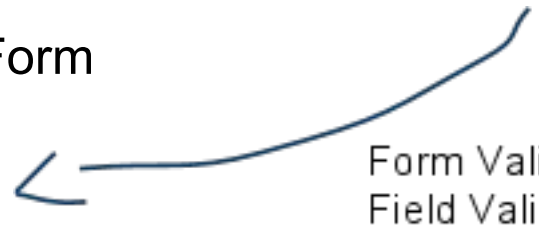
Admin urls

Django mini overview (2)



Django mini overview (4)

```
from django.forms import ModelForm
    class ArticleForm(ModelForm):
   class Meta:
       model = Article
       exclude = ('title',)
```



Form Validation (clean())
Field Validation (clean_date())
Model (save())

```
ArticleForm(request.POST).save()
ArticleForm(instance=Article.objects.get(pk=1)).save()
ArticleForm(request.POST, instance=Article.objects.get(pk=pk)).save()
```

```
<form action="/form/" method="post">
    {{ form.as_p }}
    <input type="submit" value="Submit" />
</form>
```

Django mini overview (4)

Authentication backends
Middleware
Validators
Commands
Custom model fields
Custom template tags
Custom template filters
Custom storage system
PDF, CVS, JSON, XML
Jython
Deploying
Legacy database
Error reporting via e-mail
Initial data

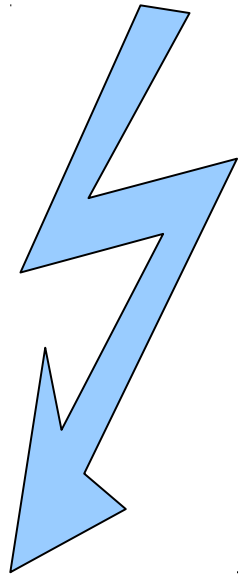
Managing files
Testing Django applications
Django's cache framework
Conditional View Processing
Sending e-mail
Internationalization and localization
Pagination
Serializing Django objects
Django settings
Signals

A lot of pluggable applications...

South (Data and DDL migrations)

New Django 1.2 Features

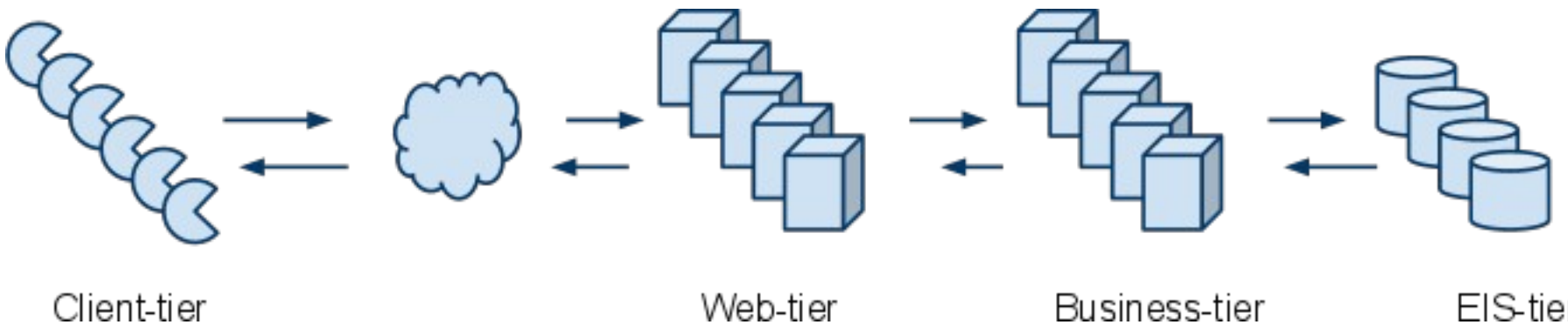
- Model Validation
- Multi Database

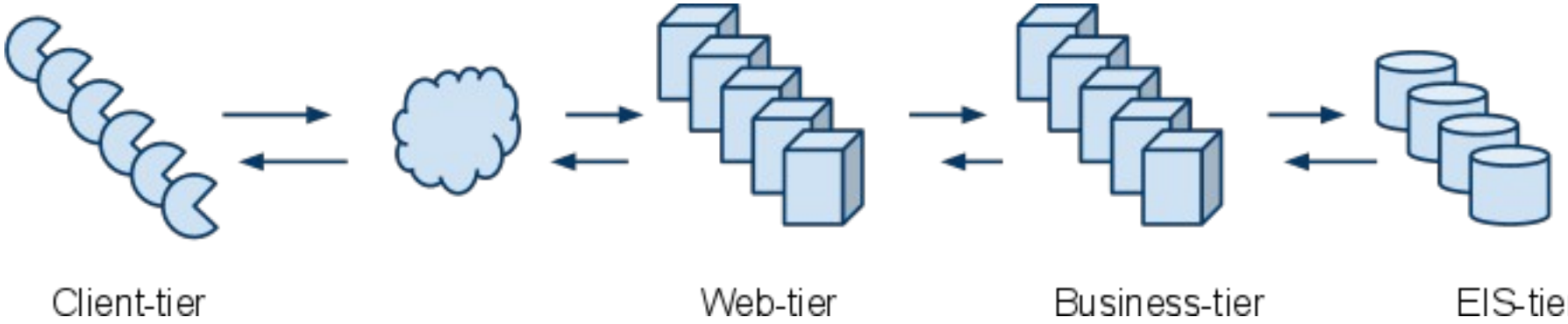


Enterprise World

Defines an architecture for implementing services as multitier applications that deliver the scalability, accessibility, and manageability needed by enterprise-level applications.

Distributed multitiered application model





- Web
  - browser
  - ajax
  - flash
  - flex
- GUI
- Batch

- Presentation-oriented
  - GET,
  - HEAD,
  - POST,
  - PUT,
  - DELETE
- Service-oriented (WS)
  - XML-RPC
  - SOA
  - REST
  - JSON

- Persistence Entities
- Session Beans
- Message-Driven Beans

- DB
- Legacy
- NO SQL
- FTP
- Remote
- JMS!

EE Containers
Centralized Configuration

- JNDI
- Datasource e Connection Pool (DB tuning)
- Mail Server (SMTP configuration)
- Enterprise JavaBeans (EJB) container
- Web container
- Application client container
- Applet container

Web Tier

Web Applications

- Servlet
- Filters
- Session Listner
- CustomTag
- Locale
- JSF (why?)

Web Services

- XML
- SOAPTransport Protocol
- WSDL Standard Format
- UDDI and XML Standard Formats
- Attachments
- Authentication

Business Tier

Local and Remote

Enterprise bean is a server-side component that encapsulates the business logic of an application. For several reasons, enterprise beans simplify the development of large, distributed applications.

- the bean developer can concentrate on solving business problems because the container is responsible for system-level services such as transaction management and security authorization.
- the client developer can focus on the presentation of the client. The client developer does not have to code the routines that implement business rules or access databases.
- because enterprise beans are portable components, the application assembler can build new applications from existing beans.

When

- The application must be scalable
- Transactions must ensure data integrity
- The application will have a variety of clients

Type

- Session (stateful/stateless)
- Message-Driven
- Asyncronous

Persistence
ORM

- provides an object/relational mapping facility to developers for managing relational data in applications.
  - The query language
    - **Finding Entities**
    - Persisting Entity Instances
  - Object/relational mapping metadata
    - **Entities, Multiplicity,** *One-to-one*, *One-to-many, Many-to-one, Many-to-many*
    - **Cascade Deletes**

Services
Authentication

- Sucurity
  - Initial Authentication
  - URL Authorization
  - Invoking Business Methods (remotly)
- Realms,Users,Groups, and Roles
  - LDAP or DB
- SSL and Certificates

JMS

Asynchronous Messagges

- Allows applications to create, send, receive, and read messages using reliable, asynchronous, loosely coupled communication.
- Messaging is a method of communication between software components or applications. A messaging system is a peer-to-peer facility: A messaging client can send messages to, and receive messages from, any other client. Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages.
- Messaging enables distributed communication that is loosely coupled. A component sends a message to a destination, and the recipient can retrieve the message from the destination.

- **Asynchronous**:

provider can deliver messages to a client as they arrive; a client does not have to request messages in order to receive them.

- **Reliable**:

can ensure that a message is delivered once and only once. Lower levels of reliability are available for applications that can afford to miss messages or to receive duplicate messages.

Transactions

- A typical enterprise application accesses and stores information in one or more databases.
  - **TransactionTimeouts**
  - **Updating Multiple Databases**

Reset Enterprise (DEVEL)

- Web
- WebServices
- Remote Invocation
- Distribuite Task
- Messaging
- Pooling
- Transaction
- Security

Reset Enterprise (DEVEL)

Keep It Simple Stupid

- Web
- WebServices
- Remote Invocation
- Distribuite Task
- Messaging
- Pooling
- Transaction
- Security

- **django / web2py / ecc..**
- **SOAPpy / jsonrcp**
- **PyRo / RPyC**
- **Celery / wh y prefer?**
- **Stomp / JMS Bridge**
- **What's you need?**
- **PEP: 249**
- **django / pattern / mind**

It's python!

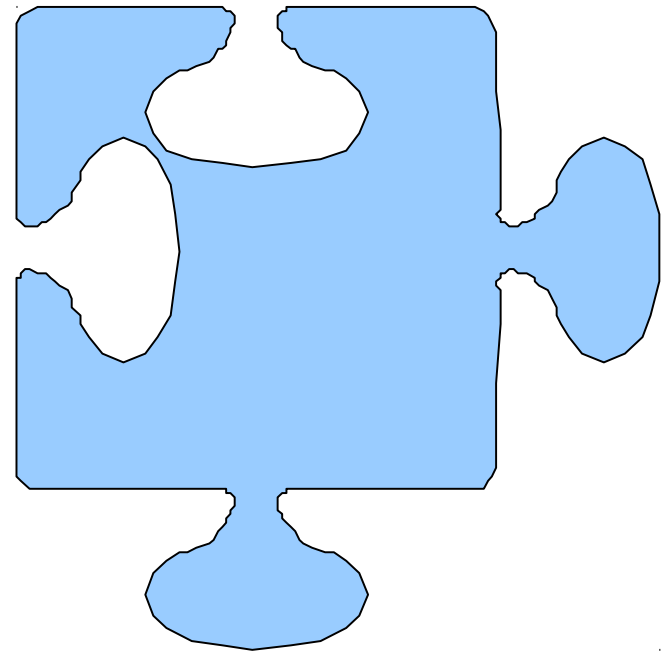
```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Why people ask for Struts or JSF Programmers?

- Because they want:
  - XML programming
  - No simple URL mappings (is XML not RE)
  - @nnotations with no logic inside
  - S****d Beans with getter and setter
  - the validation written in XML
  - Some complex container to inject A->B
  - An enterprise language...
  - Easy deploy: any fool can do this!

Packaging, Distributions, Deployments

- egg
- setuptools
- easy_install
- mod_wsgi
- mod_python
- apache restart

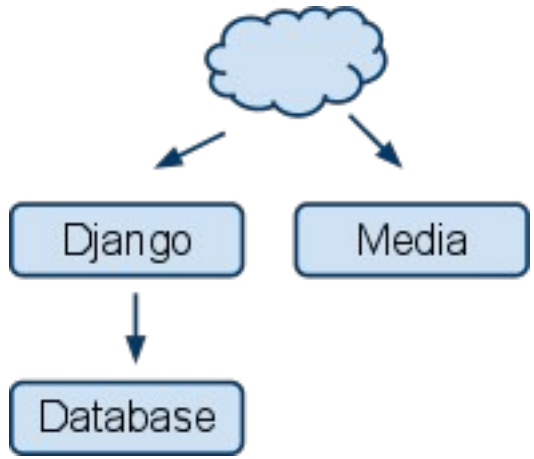


Interview

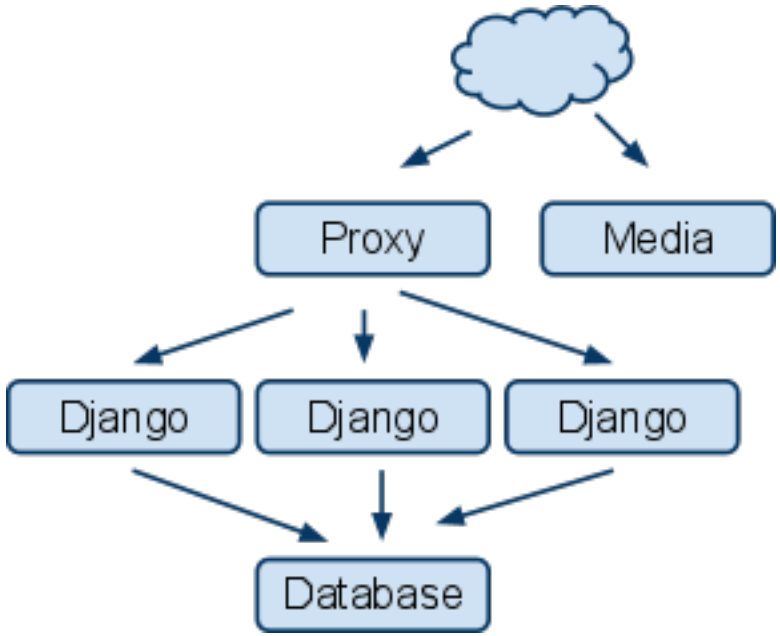
When develop an enterprise application?

- High number of users
- Distribuite Applications
- Transactional
- Clustering
  - High Reliability
  - Faul Tolerance
- Load Balancing (plugin)
  - Stiky sessions

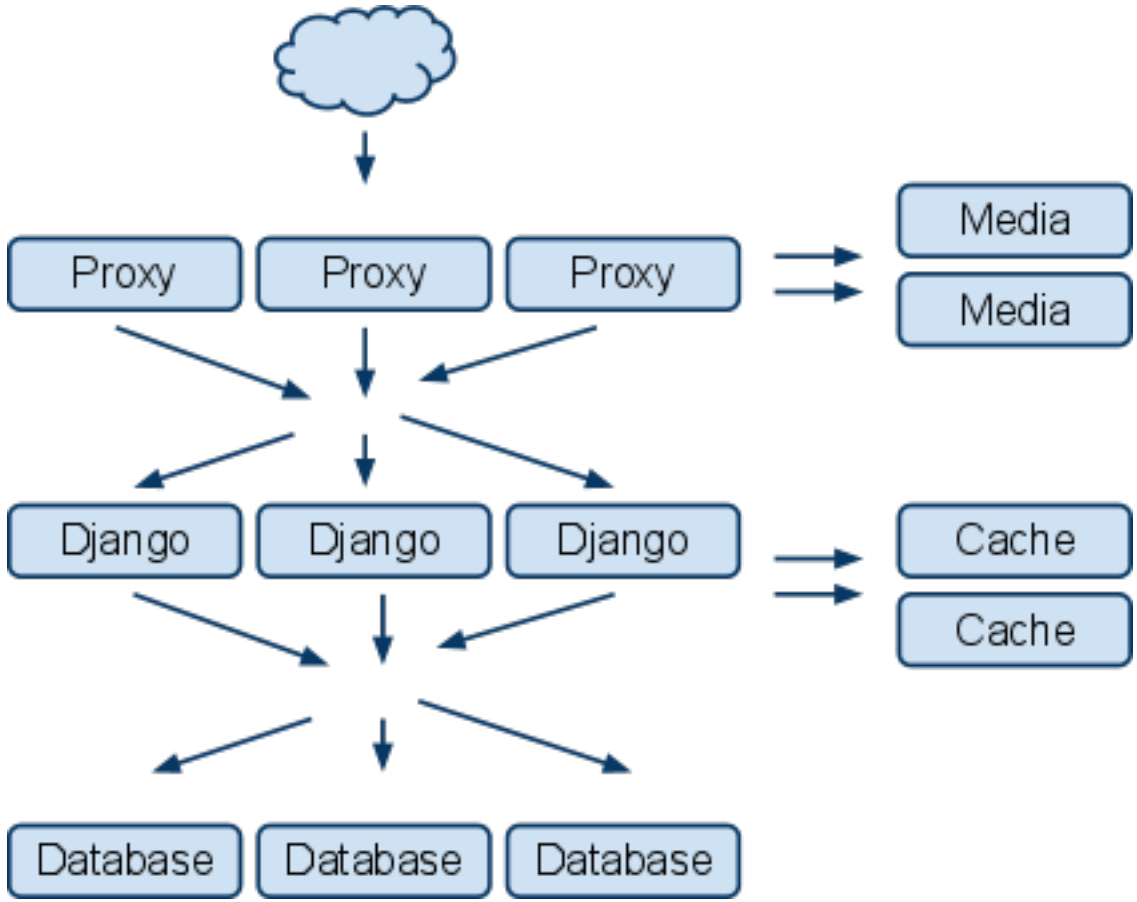
Scalability, Accessibility, and Manageability



Scalability, Accessibility, and Manageability



Scalability, Accessibility, and Manageability



Enterprise Performance Management

Just an open point....

monitoring applications
in production...

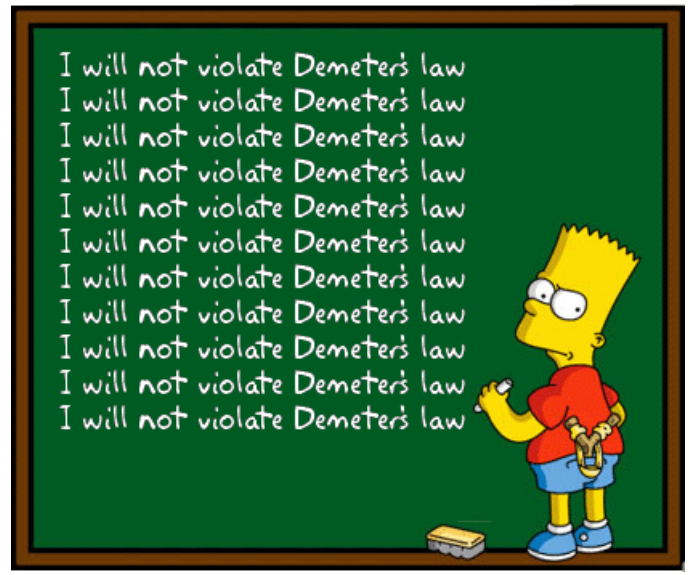
Java

- Wily Introscope
- Dynatrace
- JXInsight

Python

?

Questions?



s.federici@gmail.com