Pyramid
The Start Small, Finish Big
Stay Finished Framework

pyrasample

- Walkthough jednoduchou aplikací:

  https://github.com/petrblahos/pyrasample
- Prohlížení jakékoliv databáze

- Traversal
- Mapování kontextu na view
- Databáze

pyrasample

```
python -m venv PYRASAMPLE
cd PYRASAMPLE
git clone https://github.com/petrblahos/pyrasample.git
. bin/activate
cd pyrasample
pip install U pip setuptools
pip install -e .
```

# Editujte development.ini: sqlalchemy.url

```
pserve –reload development.ini
```

Struktura

```
pyrasample
pyrasample/__init__.py
pyrasample/resources.py
pyrasample/views.py
pyrasample/templates/layout.mako
pyrasample/templates/home.mako
pyrasample/templates/...
pyrasample/model/__init__.py
pyrasample/model/meta.py
pyrasample/static/...
MANIFEST.in
setup.py
development.ini
production.ini
```

pyrasample/__init__.py

```
def main(global_config, **settings):
    """
    This function returns a
    Pyramid WSGI application.
    """
    config = Configurator(settings=settings,
                          root_factory=TopContext)
    config.add_static_view('static', 'static',
                           cache_max_age=3600)

    config.include('pyramid_mako')
    config.include('pyrasample.model')

    config.scan()

    return config.make_wsgi_app()
```

model/__init__.py

```
def get_tm_session(session_factory, transaction_manager):
    dbsession = session_factory()
    zope.sqlalchemy.register(
        dbsession, transaction_manager=transaction_manager)
    return dbsession

def includeme(config):
    settings = config.get_settings()
    config.include('pyramid_tm')
    engine = sqlalchemy.engine_from_config(
        settings, "sqlalchemy.")
    # [ ... ]
    session_factory = get_session_factory(engine)
    config.registry['dbsession_factory'] = session_factory

    config.add_request_method(
        # r.tm is the transaction manager used by pyramid_tm
        lambda r: get_tm_session(session_factory, r.tm),
        'db',
        reify=True
    )
```

model/__init__.py

```
def get_tm_session(session_factory, transaction_manager):
    dbsession = session_factory()
    zope.sqlalchemy.register(
        dbsession, transaction_manager=transaction_manager)
    return dbsession

def includeme(config):
    settings = config.ge
    config.include('pyram
    engine = sqlalchemy.engine_          ig(
        settings, "sqlalchemy.")
    # [ ... ]
    session_factory = get_session_ actory(engine)
    config.registry['dbsession_factory'] = session_factory

    config.add_request_method(
        # r.tm is the tran                                  tm
        lambda r: get_tm_sessi
        'db',
        reify=True
    )
```

Ve view:
`request.registry['dbsession_factory']`

Ve view:
`request.db`
je db session

add_request_method

- SQLAlchemy session
- Lokalizace:
  - `request.translate`
  - `request.pluralize`
  - `request._`
- Mongo DB connection

- reified

pyrasample/__init__.py

```python
def main(global_config, **settings):
    """
    This function returns a
    Pyramid WSGI application.
    """
    config = Configurator(settings=settings,
                          root_factory=TopContext)
    config.add_static_view('static', 'static',
                           cache_max_age=3600)

    config.include('pyramid_mako')
    config.include('pyrasample.model')

    config.scan()

    return config.make_wsgi_app()
```

Traversal

- Způsob mapování URL na view
- URL → Context
- Context → View

- (virtuální filesystém)

Traversal

- URL path: /customers/CustomerId=6/name
- Resource Tree
- Root Context: Znáš kontext `customers` ?
  - Ne: Vyvolá `KeyError` , vyhledávání končí
  - Ano: Vrátí kontext pro `customers`
    - Znáš kontext pro `CustomerId=6` ?
      - Ne: Vyvolá `KeyError`
      - Ano: Vrátí kontext pro `CustomerId=6`
        - Znáš kontext pro `name` ?
          - ...

Traversal

```
class TopContext(object):
    __name__ = ""
    __parent__ = None

    def __init__(self, request):
        self.request = request
        self.tables = metadata.tables

    def __getitem__(self, key):
        return DBContext(self, key)
```

Traversal

```
class DBContext(object):
    def __init__(self, parent, name):
        self.__parent__ = parent
        self.__name__ = self.table_name = name
        self.request = parent.request
        self.model = metadata.tables[self.table_name]

    def __getitem__(self, name):
        # Schema: column1=value1&column2=value2
        params = parse_qs(name)
        q = self.request.db.query(self.model)
        for pk in self.model.primary_key:
            q = q.filter(pk == params[pk.name][0])
        return ItemContext(self, name, q.one())
        # FIXME: nevyvolá KeyError - vyjasnit
```

Na konci traversal

- Máme:
  - request
  - context
  - něco co se nespotřebovalo: view_name
- View:

```
@view_config(
    context="pyrasample.resources.TopContext",
    renderer="templates/home.mako"
)
def home(context, request):
    # context je typu TopContext
    # Máme k dispozici: context.tables
    return {}
```

Jiné view

```
@view_config(
    context="pyrasample.resources.DBContext",
    renderer="templates/dbtable.mako",
)
def dbtable(request):
    # context je taky v request.context
    # a je to DBContext
    # takže máme: request.context.model
    #             request.context.get_query
    return {}
```

Další view

```
@view_config(
    context="pyrasample.resources.ItemContext",
    renderer="templates/dbitem.mako"
)
def dbitem(request):
    return {}
```

Predikáty

```
@view_config(
    context="pyrasample.resources.ItemContext",
    renderer="templates/dbitem.mako", name="edit",
    permission="edit", request_method="GET",
)
def dbitem_edit(context, request):
    form = context.item.make_form(request)
    return {"form": form}

@view_config(
    context="pyrasample.resources.ItemContext",
    renderer="templates/dbitem.mako", name="edit",
    permission="edit", request_method="POST",
)
def dbitem_save(context, request):
    # zpracuj POST
    return HTTPFound(
        location=request.resource_url(context))
```

Nejlepší kousky

- View handler:
  - má všechna data
- Co jste neviděli
  - můžeme dělat kontrolu request params mimo View
  - request params jako parametry funkce – usnadní testování
  - predikáty – vestavěné i uživatelsky definované
  - object level security

pyrasample/__init__.py

```
def main(global_config, **settings):
    """
    This function returns a
    Pyramid WSGI application.
    """
    config = Configurator(settings=settings,
                          root_factory=TopContext)
    config.add_static_view('static', 'static',
                           cache_max_ag

    config.include('pyramid_mako')
    config.include('pyrasample.model')

    config.scan()

    return config.make_wsgi_app()
```

direktivy jako
@view_config
@subscriber

Zdroje

- https://trypyramid.com/
- https://pyramid.readthedocs.org/
- https://docs.pylonsproject.org/projects/pyramid-cookbook/en/latest/index.html

- https://github.com/petrblahos/pyrasample