A Practical Road to SaaS' in Python

Armin @mitsuhiko Ronacher
Hi, I'm Armin and I do Open Source tools of Python and SaaS.

Sentry
Flask
Flask
web development,
one drop at a time
TypeError: Cannot read property 'componentWillReceiveProps' of null

browser = Chrome 50.0  device = Other  environment = prod  level = error  logger = javascript
os = Mac OS X 10.11.4  release = a8962617411521449321efc1e8d29bc7018a287bfc64
sentry.version = 1c5ae57c8a9c1dec5ebc83331d8e6572879ea2ad
url = https://app.getsentry.com/make-me-up-co/  user = id:43428
TypeError
Cannot read property 'id' of undefined

```javascript
../app/views/projectDetails.jsx in Constructor.onProjectChange at line 90:59

85.   onProjectChange(projectIds) {
86.     if (!this.state.project) return;
87.     if (!projectIds.has(this.state.project.id)) return;
88. 
89.     this.setState({
90.         project: [...ProjectStore.GetById(this.state.Project.id)]
91.     });
92. },
93. 
94.   identifyProject() {
95.     let {params} = this.props;

Called from: ~/reflux-core/lib/PublisherMethods.js in EventEmitter.eventHandler

../app/stores/projectStore.jsx in Store.loadInitialData at line 25:9

../app/views/organizationDetails.jsx in Request.success at line 100:21

../app/api.jsx in Object.success at line 61:20

Called from: ~/jquery/dist/jquery.js in fire
```
I love Open Source
Therefore I love SaaS
Multi Tenant
But also …

On Premises?
MANAGED CLOUD?
Python
Why Python?
Python in 2017
STRONG ECOSYSTEM
Fast Iteration
Stable Environment
Powerful Metaprogramming
Fast Interpreter Introspection
Quo Vadis?
Python 2.7 / 3.6
Machine Learning
The Foundation
Flask
web development, one drop at a time
roll your own?
Application Architecture
Security First
patterns are universal
examples are Flask + Flask-SQLAlchemy
If you only take one thing away from this talk ...
Context Awareness

… or how I learned to love the thread-local bomb
from flask import g, request

def get_tenant_from_request():
    auth = validate_auth(request.headers.get('Authorization'))
    return Tenant.query.get(auth.tenant_id)

def get_current_tenant():
    rv = getattr(g, 'current_tenant', None)
    if rv is None:
        rv = get_tenant_from_request()
        g.current_tenant = rv
    return rv
def batch_update_projects(ids, changes):
    projects = Project.query.filter(
        Project.id.in_(ids) &
        Project.status != ProjectStatus.INVISIBLE
    )
    for project in projects:
        update_project(project, changes)
Automatic Tenant Scoping

class TenantQuery(db.Query):
    current_tenant_constrained = True

def tenant_unconstrained_unsafe(self):
    rv = self._clone()
    rv.current_tenant_constrained = False
    return rv

@db.event.listens_for(TenantQuery, 'before_compile', retval=True)
def ensureTenantConstrained(query):
    for desc in query.column_descriptions:
        if hasattr(desc['type'], 'tenant') and \\ query.current_tenant_constrained:
            query = query.filter_by(tenant=get_current_tenant())
    return query
from sqlalchemy.ext.declarative import declared_attr

class TenantBoundMixin(object):
    query_class = TenantQuery

    @declared_attr
    def tenant_id(cls):
        return db.Column(db.Integer, db.ForeignKey('tenant.id'))

    @declared_attr
    def tenant(cls):
        return db.relationship(Tenant, uselist=False)
Example Use

class Project(TenantBoundMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(100))
    status = db.Column(db.Integer)

    def __repr__(self):
        return '<Project name=%r>' % self.name

>>> test.Project.query.all()
[<Project name='project42']
>>> test.Project.query.tenant_unconstrained_unsafe().all()
[<Project name='project1'>, Project.name='project2', ...]
careful about backrefs!
Flask-SQLAlchemy lets you set a default query class for all things
Uses for Context
Current User
def load_user_from_request():
    user_id = session.get('user_id')
    if user_id is not None:
        return User.query.get(user_id)
    return None

def get_current_user():
    rv = getattr(g, 'current_user', None)
    if rv is None:
        rv = g.current_user = load_user_from_request()
    return rv
User Access Scope
Restrictions
def get_current_scopes():
    current_user = get_current_user()
    if current_user is None:
        all_scopes = set(['anonymous'])
    else:
        all_scopes = current_user.get_roles()
    return all_scopes & scopes_from_request_authorization()
Audit Logs
def log(action, message=None):
    data = {
        'action': action,
        'timestamp': datetime.utcnow()
    }
    if message is not None:
        data['message'] = message
    if request:
        data['ip'] = request.remote_addr
    user = get_current_user()
    if user is not None:
        data['user'] = User
    db.session.add(LogMessage(**data))
i18n/l10n
def get_current_language():
    user = get_current_user()
    if user is not None:
        return user.language
    if request and request.accept_languages:
        return request.accept_languages[0]
    return 'en_US'
Design as you go
Build first, then evolve
Sentry is still non-sharded Postgres
Python helps with Prototype to Production
Operating Python
CPython: Refcounting
PyPy: GC
sys._getframe()
ZeroDivisionError: integer division or modulo by zero
mitsuhiko at herzog in ~/Development/sentry on git:feature/symbolserver? workon sentry
$ sentry shell
from sePython 2.7.10 (default, Jul 30 2016, 19:40:32)
[ GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34) ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from sentry.models import Project
>>> p = Project.objects.get(slug='internal')
>>> p.name
u'Internal'
>>> p.slug
u'internal'
>>>
Process and Data
deploy in seconds
be unable to screw up
and if you do: instant rollbacks
commit
→
review
→
integration
→
deploy
requires good test coverage
requires good local setup
makes it easier for newcomers
lint on commit!

mitsuhiko at herzog in ~/Development/sentry on git:master+
$ workon sentry
$ git ci -am 'Performance improvements to the data scrubber.'
src/sentry/utils/data_scrubber.py:147:1: F401 'unused' imported but unused
FLAKE8 & CUSTOM LINTERS
master is stable
(HOW TO)

AVOID DOWNTIME
bidirectional compatibility
My Opinion: Invest into Fast Iteration rather than Scalability
Duck-Typing Helps Here
Quick Release Cycles
large systems are organisms
not all things run the same code at the same time
BREAK UP FEATURES
FEATURE FLAG THEM
Make Prod & Dev Look Alike
On Prem?
TWO RELEASE CYCLES

HOURLY SAAS

SIX-WEEK ON-PREM
CONSIDER SHIPPING WIP FEATURE FLAG IT AWAY
class Feature(object):
    def __init__(self, key, scope, enable_chance=None, default=False):
        self.key = key
        self.scope = scope
        self.enable_chance = enable_chance
        self.default = default

    def evaluate(self):
        scope = self.scope(self)
        value = load_feature_flag_from_db(self.key, scope)
        if value is not None:
            return value
        if self.enable_chance:
            if hash_value(scope) / float(MAX_HASH) > self.enable_chance:
                return True
        return self.default
def ip_scope(feature):
    if request:
        return 'ip:%s' % request.remote_addr

NEW_SIGN_IN_FLOW = Feature(
    key='new-sign-in-flow',
    scope=ip_scope,
    enable_chance=0.9,
    allow_overrides='admin',
    default=False,
)
User Features

def new_dashboard_default():
    tenant = get_current_tenant()
    if tenant.creation_date > datetime(2017, 1, 1):
        return True
    return False

NEW_DASHBOARD = Feature(
    key='new-dashboard',
    scope=user_scope,
    allow_overrides='user',
    default=new_dashboard_default,
)
Testing Features

if is_enabled(NEW_DASHBOARD):
...

• Cache
• Prefetch
• Easier Grepping
Mastering Deployments
Build Wheels
then follow up with Docker Images