Coding

- Python
    - Wordpress-compatible XML-RPC API

Python

Python

Wordpress-compatible XML-RPC API

```python
import string
from datetime import datetime
from flask import url_for
from flask.ext.xmlrpc import XMLRPCHandler, Fault
from labs import app, db
from labs.models import User, Post, Tag, Category
POST_ROOT = 'http://127.0.0.1/post/'

# MetaWeblogAPI XML-RPC
handler = XMLRPCHandler('api')
handler.connect(app, '/api')
metaweblog = handler.namespace('metaWeblog')
blogger = handler.namespace('blogger')
wordpress = handler.namespace('wp')
moveabletype = handler.namespace('mt')
@metaweblog.register
def newPost(blog_id, username, password, content, publish):
    user = db.session.query(User).filter(User.username == username).first()
    if user is None or not user.check_password(password):
        raise Fault("invalid_user",
                    "Invalid username/password, please try again.")
    post = Post(content['title'], content['description'])
    post.author = user
    post.teaser = content['mt_excerpt']
    if 'wp_slug' in content:
        post.slug = content['wp_slug']
    if 'dateCreated' in content:
        post.create_date = datetime.strptime(str(content['dateCreated']),
                                             "%Y%m%dT%H:%M:%SZ")
    if 'custom_fields' in content:
        for custom_field in content['custom_fields']:
            if custom_field['key'] == 'subtitle':
                post.subtitle = custom_field['value']
            elif custom_field['key'] == 'lead_img':
                post.lead_img = custom_field['value']
    tag_names = string.split(content['mt_tags'], ',')
    for tag_name in tag_names:
        tag = Tag.query.filter(Tag.name == tag_name).first()
        if tag is None:
            tag = Tag(tag_name)
            db.session.add(tag)
            db.session.commit()
        post.tags.append(tag)
    db.session.add(post)
    db.session.commit()
    return post.id


@metaweblog.register
def editPost(post_id, username, password, content, publish):
    user = db.session.query(User).filter(User.username == username).first()
    if user is None or not user.check_password(password):
        raise Fault("invalid_user",
                    "Invalid username/password, please try again.")
    post = Post.query.get(post_id)
    post.title = content['title']
    post.markdown = content['description']
    post.set_html()
    post.teaser = content['mt_excerpt']
    if 'wp_slug' in content:
        post.slug = content['wp_slug']
    if 'dateCreated' in content:
        post.create_date = datetime.strptime(str(content['dateCreated']),
                                             "%Y%m%dT%H:%M:%SZ")
    if 'custom_fields' in content:
        for custom_field in content['custom_fields']:
            if custom_field['key'] == 'subtitle':
                post.subtitle = custom_field['value']
            elif custom_field['key'] == 'lead_img':
                post.lead_img = custom_field['value']
    tag_names = string.split(content['mt_tags'], ',')
    tags = []
    for tag_name in tag_names:
        tag = Tag.query.filter(Tag.name == tag_name).first()
        if tag is None:
            tag = Tag(tag_name)
            db.session.add(tag)
            db.session.commit()
        tags.append(tag)
    post.tags = tags
    db.session.add(post)
    db.session.commit()
    return True


@metaweblog.register
def getPost(post_id, username, password):
    user = db.session.query(User).filter(User.username == username).first()
    if user is None or not user.check_password(password):
        raise Fault("invalid_user",
                    "Invalid username/password, please try again.")
    post = Post.query.filter(Post.id == post_id).first()
    if not post:
        raise Fault("not_found", "Post not found.")
    item = {}
    item['title'] = post.title
    item['link'] = POST_ROOT + post.slug
    item['description'] = post.markdown
    item['postid'] = post.id
    item['mt_excerpt'] = post.teaser
    item['custom_fields'] = [
        {
            'key': 'subtitle',
            'value': post.subtitle
        },
        {
            'key': 'lead_img',
            'value': post.lead_img
        }
    ]
    item['wp_slug'] = post.slug
    if post.tags:
        item['mt_tags'] = ','.join(map(lambda tag: tag.name, post.tags))
    item['dateCreated'] = post.create_date
    return item


@metaweblog.register
def getRecentPosts(blogid, username, password, numberOfPosts):
    user = db.session.query(User).filter(User.username == username).first()
    if user is None or not user.check_password(password):
        raise Fault("invalid_user",
                    "Invalid username/password, please try again.")
    posts = Post.query.order_by('create_date').all()
    response = []
    for post in posts:
        item = {}
        item['title'] = post.title
        item['link'] = POST_ROOT + post.slug
        item['description'] = post.markdown
        item['postid'] = post.id
        item['mt_excerpt'] = post.teaser
        item['wp_slug'] = post.slug
        item['custom_fields'] = [
            {
                'key': 'subtitle',
                'value': post.subtitle
            },
            {
                'key': 'lead_img',
                'value': post.lead_img
            }
        ]
        tags = []
        for tag in post.tags:
            tags.append(tag.name)
        item['mt_tags'] = ','.join(tags)
        item['dateCreated'] = post.create_date
        # if post['draft']:
        #     item['draft'] = 'Yes'
        response.append(item)
    return response


@wordpress.register
def getPages(blogid, username, password, numberOfPages):
    return []


@wordpress.register
def newCategory(blogid, username, password, new_category):
    user = db.session.query(User).filter(User.username == username).first()
    if user is None or not user.check_password(password):
        raise Fault("invalid_user",
                    "Invalid username/password, please try again.")
    category = Category.query.filter(Category.name == new_category['name']).first()
    if category is None:
        category = Category(new_category['name'])
        db.session.add(category)
        db.session.commit()
    return category.id


@wordpress.register
def getTags(blogid, username, password):
    return map(lambda tag: {
            'tag_id': tag.id,
            'name': tag.name
        }, Tag.query.all())


@wordpress.register
def getCategories(blogid, username, password):
    return map(lambda category: {
            'categoryId': category.id,
            'categoryName': category.name,
            'categoryDescription': category.description
        }, Category.query.all())


@moveabletype.register
def setPostCategories(post_id, username, password, post_categories):
    post = Post.query.get(post_id)
    for post_category in post_categories:
        category = Category.query.filter(
            Category.name == post_category['categoryName']
        ).first()
        # only single category per post supported
        post.category = category
    db.session.add(post)
    db.session.commit()
    return True
```

```python
@moveabletype.register
def getPostCategories(post_id, username, password):
    # only single per post supported
    category = Post.query.get(post_id).category
    if category is not None:
        post_category = {
            'categoryId': category.id,
            'categoryName': category.name,
            'categoryDescription': category.description
        }
        return [post_category]
    return []


@moveabletype.register
def supportedTextFilters():
    return []


@blogger.register
def deletePost(appkey, post_id, username, password, publish):
    user = db.session.query(User).filter(User.username == username).first()
    if user is None or not user.check_password(password):
        raise Fault("invalid_user",
                    "Invalid username/password, please try again.")
    post = Post.query.get(int(post_id))
    db.session.delete(post)
    db.session.commit()
    pass
```