Министерство образования и науки РФ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Уральский федеральный университет

имени первого Президента России Б. Н. Ельцина»

Институт фундаментального образования

Кафедра интеллектуальных информационных технологий

**РАЗРАБОТКА САЙТА**

**C ФЭНТЕЗИЙНЫМ КОНТЕНТОМ**

Курсовая работа по дисциплине

«WEB-программирование»

Пояснительная записка

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Преподаватель | |  | |  |  | Свинцов Д. В. | |
|  | |  | | подпись преподавателя |  |  | |
|  | |  | |  |  |  | |
| Студентка гр. ФО-331002 | |  | |  |  | Бердюгина А. О. | |
|  | |  | | подпись студента |  |  | |
| Дата сдачи | |  | | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2016 г. | | | |
|  | |  | |  | | | |

Екатеринбург

2016

Оглавление

|  |
| --- |
| Введение……………………………………………………………………………3 |
| Описание проекта………………………………………………………………….4 |
| Обоснование проекта……………………………………………………………...5 |
| Инструкция пользователя…………………………………………………………6 |
| Инструкция программиста………………………………………………….…….7 |
| Описание модулей…………………………………………………………………8 |
| Выводы……………………………………………………………………………14 |
| Список литературы………………………………………………………………15 |

Введение

В начале семестра нас осведомили, что нужно будет выбрать тему для курсовой работы. Мой выбор был в пользу разработки сайта для «любителей» фэнтези.

Для создания нужно было использовать определенные технологии и архитектуру, прописанные в требованиях к курсовой работе.

Описание проекта

Сайт состоит из четырех основных страниц и страницы авторизации:

* Главная(index.jinja2) – содержит в себе общую информацию о сайте;
* Книги(bookPage.jinja2) – содержит информацию о фэнтезийных книгах, так же можно выбрать коллекцию, новинку и популярное;
* Фильмы(filmPage.jinja2) – содержит информацию о фэнтезийных фильмах;
* Статьи(historyPage.jinja2) – содержит статьи, рецензии;
* Авторизация(avtorization.jinja2) – содержит несколько полей и кнопку.

Он должен предоставлять пользователям доступ к информации, посредствам перехода между страницами(вкладками).

Обоснование проекта

Сайт мог бы быть полезен людям, которые ищут, что почитать, посмотреть в стиле фэнтези на просторах интернета. Экономической выгоды как таковой нет, если только не покупать права у авторов и не продавать их произведения.

Инструкция пользователя

Сайт состоит из четырех основных страниц и одной страницы авторизации, переход на каждую из которых производится при нажатии на соответствующий заголовок или ссылку.

Инструкция программиста

Установка

Для установки требуется версия Python 3.5, пакеты Jinja2, PyramidFramework, SQLAlchemy.

В командной строке необходимо создать виртуальной пространство и перейти в папку проекта: *cd C:\Users\lenovo\Documents\WEB\myproject (для Windows)*

Инициализация базы данных

Для инициализации базы данных введите в командной строке:

*"C:\Program Files\Python 3.5\Scripts\initialize\_myproject\_db.exe" development.ini (для Windows)*

Запуск проекта

Для запуска проекта введите в командной строке:

*"C:\Program Files\Python 3.5\Scripts\pserve.exe" development.ini –reload (для Windows)*

Авторизация

Для того, чтобы зайти на сайт в качестве администратора, нужно перейти на страницу с формой авторизации, нажав на ссылку в правом верхнем углу. Логин и пароль указаны в файле security.py.

Если пользователь авторизован, логин указывается в левом верхнем углу сайта.

Описание модулей

* \_init\_.py – модуль который содержит в себе настройки авторизации, аутентификации, роуты страниц;

from pyramid.config import Configurator

from sqlalchemy import engine\_from\_config

from pyramid.authentication import AuthTktAuthenticationPolicy

from pyramid.authorization import ACLAuthorizationPolicy

from .security import groupfinder

from .models import ( DBSession, Base, User, Collection, Book, Film, History, )

def main(global\_config, \*\*settings):

""" This function returns a Pyramid WSGI application.

"""

engine = engine\_from\_config(settings, 'sqlalchemy.')

DBSession.configure(bind=engine)

Base.metadata.bind = engine

authn\_policy = AuthTktAuthenticationPolicy(

'secret', callback=groupfinder, hashalg='sha512')

authz\_policy = ACLAuthorizationPolicy()

config = Configurator(settings=settings, root\_factory='myproject.mymodel.Access')

config.set\_authentication\_policy(authn\_policy)

config.set\_authorization\_policy(authz\_policy)

config.include('pyramid\_jinja2')

config.add\_static\_view('static', 'static')

config.add\_route('index', '/index')

config.add\_route('book', '/book')

config.add\_route('film', '/film')

config.add\_route('history', '/history')

config.add\_route('login', '/login')

config.add\_route('logout', '/logout')

config.include('pyramid\_sacrud',route\_prefix='admin')

settings = config.registry.settings

settings['pyramid\_sacrud.models'] = (('Project', [User, Collection, Book, Film, History]),)

config.scan()

return config.make\_wsgi\_app()

* view.py – модуль который содержит представления для страниц сайта в том числе и авторизацию;

from pyramid.response import Response

from pyramid.view import view\_config

from sqlalchemy.exc import DBAPIError

from .models import(DBSession, User, Collection, Book, Film, History, Access)

from pyramid.httpexceptions import(HTTPFound, HTTPNotFound)

from pyramid.view import(view\_config, forbidden\_view\_config)

from pyramid.security import(remember, forget)

from .security import USERS

@view\_config(route\_name='index', renderer='myproject/templates/index.jinja2')

def index(request):

if request.authenticated\_userid == "admin":

return {'username': request.authenticated\_userid, 'project': 'Myproject'}

return {'project': 'Myproject'}

@view\_config(route\_name='bookPage', renderer='myproject/templates/bookPage.jinja2')

def bookPage(request):

if request.authenticated\_userid == "admin":

return {'username': request.authenticated\_userid, 'project': ''}

return {'project': 'Myproject'}

@view\_config(route\_name='filmPage', renderer='myproject/templates/filmPage.jinja2')

def filmPage(request):

return {'username': request.authenticated\_userid, 'project': 'Myproject'}

@view\_config(route\_name='historyPage', renderer='myproject/templates/historyPage.jinja2')

def historyPage(request):

if request.authenticated\_userid == "admin":

return {'username': request.authenticated\_userid, 'project': ''}

return {'project': 'Myproject'}

conn\_err\_msg = """\

You have problem!

"""

@view\_config(route\_name='login', renderer='myproject/templates/avtorization.jinja2')

@forbidden\_view\_config(renderer='myproject/templates/avtorization.jinja2')

def login(request):

if 'submitted' in request.params:

login = request.params['login']

password = request.params['password']

if USERS.get(login) == password:

headers = remember(request, login)

return HTTPFound(location = 'home', headers = headers)

else:

return {'message': "Incorrect login or password", 'project': 'Myproject'}

return {'project': 'Myproject'}

@view\_config(route\_name='logout')

def logout(request):

headers = forget(request)

return HTTPFound(location = request.referrer,

headers = headers)

* mymodel.py – модуль в котором описаны таблицы базы данных и права для групп пользователей;

#описание таблиц базы данных#

from sqlalchemy import (

Column,

Index,

Integer,

Text,

ForeignKey,

)

from .meta import Base

from sqlalchemy.orm import backref, relationship

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy.orm import (

relationship,

scoped\_session,

sessionmaker,

)

from pyramid.security import (

Allow,

Everyone,

)

from zope.sqlalchemy import ZopeTransactionExtension

DBSession = scoped\_session(sessionmaker(extension=ZopeTransactionExtension()))

Base = declarative\_base()

class User(Base):

\_\_tablename\_\_ = 'user'

id = Column(Integer, primary\_key = True)

name = Column(Text, primary\_key=True)

def \_\_repr\_\_(self):

return self.name or ''

class Collection(Base):

\_\_tablename\_\_ = 'collection'

id = Column(Integer, primary\_key=True)

name = Column(Text, primary\_key=True)

info = Column(Text)

parent\_id = Column(Integer, ForeignKey(id))

parent\_collection = relationship("Collection", remote\_side="Collection.id")

def \_\_repr\_\_(self):

return self.name or ''

class Book(Base):

\_\_tablename\_\_ = 'book'

author = Column(Text, primary\_key=True)

title = Column(Text, primary\_key=True)

year = Column(Integer)

collection\_id = Column(Integer,ForeignKey('collection.id'))

collection = relationship("Collection", backref='book')

def \_\_repr\_\_(self):

return self.author + ' - ' + self.title or ''

class Film(Base):

\_\_tablename\_\_ = 'film'

title = Column(Text, primary\_key=True)

year = Column(Integer)

collection = relationship("Collection", backref='film')

def \_\_repr\_\_(self):

return self.author + ' - ' + self.title or ''

class History(Base):

\_\_tablename\_\_ = 'history'

title = Column(Text, primary\_key=True)

data = Column(Integer)

def \_\_repr\_\_(self):

return self.author + ' - ' + self.title or ''

#права для пользователей

class Access(object):

\_\_acl\_\_ = [ (Allow, 'group:editors', ('pyramid\_sacrud\_home', 'pyramid\_sacrud\_create', 'pyramid\_sacrud\_update', 'pyramid\_sacrud\_delete', 'pyramid\_sacrud\_list')),]

def \_\_init\_\_(self, request):

pass

* secuirity.py – модуль содержит группы пользователей и пароль для админа.

USERS = {'admin':'password',

'foo':'12345'}

GROUPS = {'admin':['group:editors'], 'foo':['group:editors']}

def groupfinder(userid, request):

if userid in USERS:

return GROUPS.get(userid, [])

Выводы

Я научилась создавать базу данных с помощью SQLAlchemy, создавать структуру проекта с помощью Pyramid framework, немного верстать сайт с помощью jinja2. Изучила способ авторизации пользователей.

Если я пойду работать web-разработчиком, то та база, которая была изучена в процессе выполнения работы поможет мне совершенствоваться в этой области.

Список литературы

1. http://docs.pylonsproject.org