ОГБПОУ «ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

Отчет по практической работе №1

*УП.05.01 Объектно-ориентированное программирование*

*Тема: «Реализация тестового проекта LocalLibrary»*

Выполнил:

студент 431 группы

Литвинчук Э.А.

Проверил:  
преподаватель, высшей к.к.

Фунтиков М.Н.

г. Томск – 2024 г

Содержание

1.1 Отчёт по выполнению работы

1. catalog/urls.py
2. locallibrary/urls.py
3. views.py
4. settings.py
5. models.py
6. admin.py
7. forms.py
8. catalog/urls.py

from django.urls import path, re\_path  
from . import views  
from django.contrib.auth import views as auth\_views  
from .views import logged\_out\_view  
  
urlpatterns = [  
 path('', views.index, name='index'),  
 path('books/', views.BookListView.as\_view(), name='books'),  
 re\_path(r'^book/(?P<pk>\d+)$', views.BookDetailView.as\_view(), name='book-detail'),  
 path('authors/', views.AuthorListView.as\_view(), name='authors'),  
 re\_path(r'^author/(?P<pk>\d+)$', views.AuthorDetailView.as\_view(), name='author-detail'),  
 path('logout/', auth\_views.LogoutView.as\_view(), name='logout'),  
 path('logged\_out/', logged\_out\_view, name='logged\_out'),  
 re\_path(r'^mybooks/$', views.LoanedBooksByUserListView.as\_view(), name='my-borrowed'),  
 re\_path(r'borrowed/', views.LoanedBooksAllListView.as\_view(), name='all-borrowed'),  
 re\_path(r'^book/(?P<pk>[-\w]+)/renew/$', views.renew\_book\_librarian, name='renew-book-librarian'),  
 re\_path(r'^author/create/$', views.AuthorCreate.as\_view(), name='author\_create'),  
 re\_path(r'^author/(?P<pk>\d+)/update/$', views.AuthorUpdate.as\_view(), name='author\_update'),  
 re\_path(r'^author/(?P<pk>\d+)/delete/$', views.AuthorDelete.as\_view(), name='author\_delete'),  
 re\_path('book/create/', views.BookCreate.as\_view(), name='book-create'),  
 re\_path('book/<int:pk>/update', views.BookUpdate.as\_view(), name='book-update'),  
 re\_path('book/<int:pk>/delete', views.BookDelete.as\_view(), name='book-delete'),  
]

1. locallibrary/urls.py

from django.urls import path  
from django.contrib import admin  
from django.urls import include  
from django.views.generic import RedirectView  
from django.conf import settings  
from django.conf.urls.static import static  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', RedirectView.as\_view(url='catalog/')),  
 path('catalog/', include('catalog.urls')),  
 path('accounts/', include('django.contrib.auth.urls')),  
] + static(settings.STATIC\_URL, document\_root=settings.STATIC\_ROOT)

1. views.py

def index(request):  
  
 num\_books=Book.objects.all().count()  
 num\_instances=BookInstance.objects.all().count()  
 num\_instances\_available=BookInstance.objects.filter(status\_\_exact='a').count()  
 num\_authors=Author.objects.count()  
 num\_genre = Genre.objects.count()  
 num\_visits = request.session.get('num\_visits', 0)  
 request.session['num\_visits'] = num\_visits + 1  
  
 return render(  
 request,  
 'index.html',  
 context={'num\_books':num\_books,'num\_instances':num\_instances,'num\_instances\_available':num\_instances\_available,  
 'num\_authors':num\_authors, 'num\_genre': num\_genre, 'num\_visits':num\_visits},  
 )  
  
def logged\_out\_view(request):  
 return render(request, 'logged\_out.html')  
  
def auto\_logout\_view(request):  
 return render(request, 'auto\_logout.html')  
class BookListView(generic.ListView):  
 model = Book  
 paginate\_by = 4  
  
class BookDetailView(generic.DetailView):  
 model = Book  
  
class AuthorListView(generic.ListView):  
 model = Author  
  
class AuthorDetailView(generic.DetailView):  
 model = Author  
  
class LoanedBooksByUserListView(LoginRequiredMixin,generic.ListView):  
 model = BookInstance  
 template\_name ='catalog/bookinstance\_list\_borrowed\_user.html'  
 paginate\_by = 10  
  
 def get\_queryset(self):  
 return BookInstance.objects.filter(borrower=self.request.user).filter(status\_\_exact='o').order\_by('due\_back')  
  
class LoanedBooksAllListView(PermissionRequiredMixin, generic.ListView):  
 model = BookInstance  
 permission\_required = 'catalog.can\_mark\_returned'  
 template\_name = 'catalog/bookinstance\_list\_borrowed\_all.html'  
 paginate\_by = 4  
  
 def get\_queryset(self):  
 return BookInstance.objects.filter(status\_\_exact='o').order\_by('due\_back')  
  
@login\_required  
@permission\_required('catalog.can\_mark\_returned', raise\_exception=True)  
def renew\_book\_librarian(request, pk):  
 book\_instance = get\_object\_or\_404(BookInstance, pk=pk)  
  
 if request.method == 'POST':  
  
 form = RenewBookForm(request.POST)  
  
 if form.is\_valid():  
 book\_instance.due\_back = form.cleaned\_data['renewal\_date']  
 book\_instance.save()  
  
 return HttpResponseRedirect(reverse('all-borrowed'))  
  
 else:  
 proposed\_renewal\_date = datetime.date.today() + datetime.timedelta(weeks=3)  
 form = RenewBookForm(initial={'renewal\_date': proposed\_renewal\_date})  
  
 context = {  
 'form': form,  
 'book\_instance': book\_instance,  
 }  
  
 return render(request, 'catalog/book\_renew\_librarian.html', context)  
  
class AuthorCreate(PermissionRequiredMixin, CreateView):  
 model = Author  
 fields = ['first\_name', 'last\_name', 'date\_of\_birth', 'date\_of\_death']  
 initial = {'date\_of\_death': '24/11/2024'}  
 permission\_required = 'catalog.add\_author'  
  
class AuthorUpdate(PermissionRequiredMixin, UpdateView):  
 model = Author  
 fields = '\_\_all\_\_'  
 permission\_required = 'catalog.change\_author'  
  
class AuthorDelete(PermissionRequiredMixin, DeleteView):  
 model = Author  
 success\_url = reverse\_lazy('authors')  
 permission\_required = 'catalog.delete\_author'  
  
 def form\_valid(self, form):  
 try:  
 self.object.delete()  
 return HttpResponseRedirect(self.success\_url)  
 except Exception as e:  
 return HttpResponseRedirect(  
 reverse("author-delete", kwargs={"pk": self.object.pk})  
 )  
  
class BookCreate(PermissionRequiredMixin, CreateView):  
 model = Book  
 fields = ['title', 'author', 'summary', 'isbn', 'genre', 'language']  
 permission\_required = 'catalog.add\_book'  
  
class BookUpdate(PermissionRequiredMixin, UpdateView):  
 model = Book  
 fields = ['title', 'author', 'summary', 'isbn', 'genre', 'language']  
 permission\_required = 'catalog.change\_book'  
  
class BookDelete(PermissionRequiredMixin, DeleteView):  
 model = Book  
 success\_url = reverse\_lazy('Book')  
 permission\_required = 'catalog.delete\_Book'  
  
 def form\_valid(self, form):  
 try:  
 self.object.delete()  
 return HttpResponseRedirect(self.success\_url)  
 except Exception as e:  
 return HttpResponseRedirect(  
 reverse("Book-delete", kwargs={"pk": self.object.pk})  
 )

1. settings.py

import os  
  
# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)  
BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
  
  
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = '!5-6ph-4baxl@z$p^ge=i+mn^kc+m0kxtp\_lsrqginbh\*bo7ez'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
ALLOWED\_HOSTS = []  
  
  
# Application definition  
  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'catalog.apps.CatalogConfig',  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'locallibrary.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [os.path.join(BASE\_DIR, 'templates')],  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'locallibrary.wsgi.application'  
  
  
# Database  
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),  
 }  
}  
  
  
# Password validation  
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators  
  
AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 },  
]  
  
  
# Internationalization  
# https://docs.djangoproject.com/en/3.0/topics/i18n/  
  
LANGUAGE\_CODE = 'en-us'  
  
TIME\_ZONE = 'Asia/Tomsk'  
  
USE\_I18N = True  
  
USE\_L10N = True  
  
USE\_TZ = True  
  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/3.0/howto/static-files/  
  
STATIC\_URL = '/static/'  
  
LOGIN\_REDIRECT\_URL = '/'  
  
EMAIL\_BACKEND = 'django.core.mail.backends.console.EmailBackend'

1. models.py

from django.db import models  
from django.conf import settings  
from datetime import date  
from django.urls import reverse  
import uuid  
# Create your models here.  
  
class Genre(models.Model):  
  
 name = models.CharField(max\_length=200, help\_text="Enter a book genre (e.g. Science Fiction, French Poetry etc.)")  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class Book(models.Model):  
  
 title = models.CharField(max\_length=200)  
 author = models.ForeignKey('Author', on\_delete=models.SET\_NULL, null=True)  
 summary = models.TextField(max\_length=1000, help\_text="Enter a brief description of the book")  
 isbn = models.CharField('ISBN', max\_length=13,  
 unique=True,  
 help\_text='13 Character <a href="https://www.isbn-international.org/content/what-isbn'  
 '">ISBN number</a>')  
 genre = models.ManyToManyField(Genre, help\_text="Select a genre for this book")  
 language = models.ForeignKey('Language', on\_delete=models.SET\_NULL, null=True)  
  
 class Meta:  
 ordering = ['title', 'author']  
  
 def display\_genre(self):  
 return ', '.join([ genre.name for genre in self.genre.all()[:3] ])  
  
 display\_genre.short\_description = 'Genre'  
  
 def get\_absolute\_url(self):  
 return reverse('book-detail', args=[str(self.id)])  
  
 def \_\_str\_\_(self):  
 return self.title  
  
class BookInstance(models.Model):  
  
 id = models.UUIDField(primary\_key=True, default=uuid.uuid4, help\_text="Unique ID for this particular book across whole library")  
 book = models.ForeignKey('Book', on\_delete=models.SET\_NULL, null=True)  
 imprint = models.CharField(max\_length=200)  
 due\_back = models.DateField(null=True, blank=True)  
 borrower = models.ForeignKey(  
 settings.AUTH\_USER\_MODEL, on\_delete=models.SET\_NULL, null=True, blank=True)  
  
 @property  
 def is\_overdue(self):  
 return bool(self.due\_back and date.today() > self.due\_back)  
  
 LOAN\_STATUS = (  
 ('m', 'Maintenance'),  
 ('o', 'On loan'),  
 ('a', 'Available'),  
 ('r', 'Reserved'),  
 )  
  
 status = models.CharField(  
 max\_length=1,  
 choices=LOAN\_STATUS,  
 blank=True,  
 default='m',  
 help\_text='Book availability')  
  
 class Meta:  
 ordering = ["due\_back"]  
 permissions = (("can\_mark\_returned", "Set book as returned"),)  
  
 def get\_absolute\_url(self):  
 return reverse('bookinstance-detail', args=[str(self.id)])  
  
 def \_\_str\_\_(self):  
 return f"Book: {self.book.title}, Status: {self.status}, Due Back: {self.due\_back}, ID: {self.id}"  
  
class Author(models.Model):  
 first\_name = models.CharField(max\_length=100)  
 last\_name = models.CharField(max\_length=100)  
 date\_of\_birth = models.DateField(null=True, blank=True)  
 date\_of\_death = models.DateField('Died', null=True, blank=True)  
  
 class Meta:  
 ordering = ['last\_name']  
  
 def get\_absolute\_url(self):  
  
 return reverse('author-detail', args=[str(self.id)])  
  
 def \_\_str\_\_(self):  
 return '%s, %s' % (self.last\_name, self.first\_name)  
  
 def lifespan(self):  
 if self.date\_of\_death:  
 return f"{self.date\_of\_birth} - {self.date\_of\_death}"  
  
 return f"{self.date\_of\_birth} - Present"  
  
class Language(models.Model):  
 name = models.CharField(max\_length=200,  
 unique=True,  
 help\_text="Enter the book's natural language (e.g. English, French, Japanese etc.)")  
  
 def get\_absolute\_url(self):  
 return reverse('language-detail', args=[str(self.id)])  
  
 def \_\_str\_\_(self):  
 return self.name

1. admin.py

from django.contrib import admin  
from .models import Author, Genre, Book, BookInstance  
# admin.site.register(Book)  
# admin.site.register(Author)  
admin.site.register(Genre)  
# admin.site.register(BookInstance)  
  
class AuthorAdmin(admin.ModelAdmin):  
 list\_display = ('last\_name', 'first\_name', 'date\_of\_birth', 'date\_of\_death')  
 fields = ['first\_name', 'last\_name', ('date\_of\_birth', 'date\_of\_death')]  
  
admin.site.register(Author, AuthorAdmin)  
  
class BooksInstanceInline(admin.TabularInline):  
 model = BookInstance  
 extra = 0  
  
@admin.register(Book)  
class BookAdmin(admin.ModelAdmin):  
 list\_display = ('title', 'author', 'display\_genre')  
 inlines = [BooksInstanceInline]  
  
@admin.register(BookInstance)  
class BookInstanceAdmin(admin.ModelAdmin):  
 list\_display = ('book', 'status', 'borrower', 'due\_back', 'id')  
 list\_filter = ('status', 'due\_back')  
  
 fieldsets = (  
 (None, {  
 'fields': ('book','imprint', 'id')  
 }),  
 ('Availability', {  
 'fields': ('status', 'due\_back','borrower')  
 }),  
 )

1. forms.py

from django.core.exceptions import ValidationError  
from django.utils.translation import gettext\_lazy as \_  
import datetime  
from django import forms  
  
class RenewBookForm(forms.Form):  
 renewal\_date = forms.DateField(  
 help\_text="Enter a date between now and 4 weeks (default 3).")  
  
 def clean\_renewal\_date(self):  
 data = self.cleaned\_data['renewal\_date']  
  
 if data < datetime.date.today():  
 raise ValidationError(\_('Invalid date - renewal in past'))  
 if data > datetime.date.today() + datetime.timedelta(weeks=4):  
 raise ValidationError(  
 \_('Invalid date - renewal more than 4 weeks ahead'))  
  
 return data

1.2 Ссылка на открытый git репозиторий.

[Ссылка на репозиторий](https://github.com/Elya-Limei/django.git)

Выводы по работе

В ходе разработки тестового проекта LocalLibrary были изучены основные возможности фреймворка Django что и было целью этой работы.