

Rozwiązania szkieletowe w tworzeniu aplikacji WWW

Tomasz Łukaszuk

Katedra Oprogramowania
Wydział Informatyki
Politechnika Białostocka

Temat wykładu:

Moduł administracyjny



Fundusze Europejskie
dla Rozwoju Społecznego



Rzeczpospolita
Polska

Dofinansowane przez
Unię Europejską



NCBR
Narodowe Centrum Badań i Rozwoju

Materiały przygotowane w ramach projektu "PB 5.0 – dostosowanie oferty dydaktycznej Politechniki Białostockiej do potrzeb nowoczesnej gospodarki oraz zielonej i cyfrowej transformacji" (nr umowy FERS.01.05-IP.08-0327/23-00) w ramach programu Fundusze Europejskie dla Rozwoju Społecznego 2021-2027 współfinansowanego ze środków Europejskiego Funduszu Społecznego Plus.

Publikacja jest udostępniona na licencji Creative Commons - Uznanie autorstwa 4.0 (CC BY 4.0). Pełna treść licencji dostępna na stronie creativecommons.org.

Moduł administracyjny: Agenda

- 1 Django admin
 - Podstawy
 - ModelAdmin Media
 - Niestandardowe templatki
 - ModelAdmin/ModelForm hacking
 - Niestandardowe widoki

- 2 Rails admin

Django admin

Podstawy

Aktywacja strony admina

- Moduł admin jest włączony w domyślnym szablonie projektu używanym przez startproject.
- Aplikacja `django.contrib.admin` w `INSTALLED_APPS` w `settings.py`
- Wpis w `urls.py`

```
1  # settings.py
2  INSTALLED_APPS = [
3      'django.contrib.admin',
4      'django.contrib.auth',
5      'django.contrib.contenttypes',
6      'django.contrib.sessions',
7      'django.contrib.messages',
8      'django.contrib.staticfiles',
9  ]
10
11 # urls.py
12 from django.contrib import import admin
13 from django.urls import path
14
15 urlpatterns = [
16     path('admin/', admin.site.urls),
17 ]
```

Aktywacja strony admina

Feast your eyes with the django site!

Django administration

Username:

Password:

Django administration

Welcome, [adrian](#). [Documentation](#) / [Change password](#) / [Log out](#)

Site administration

Auth	
Groups	Add Change
Users	Add Change
Site	
Sites	Add Change

Recent Actions
My Actions
None available

Rejestracja modeli i dostosowanie formularzy admina

- Plik `admin.py` w katalogu aplikacji:

```
1 from django.contrib import admin
2 from myproject.myapp.models import Author
3
4 admin.site.register(Author)
```

- Dostosowanie formularza admina:

```
1 from django.contrib import admin
2 from myproject.myapp.models import Author
3
4 class AuthorAdmin(admin.ModelAdmin):
5     pass
6 admin.site.register(Author, AuthorAdmin)
7
8 @admin.register(Author)
9 class AuthorAdmin(admin.ModelAdmin):
10     pass
```

Rejestracja modeli i dostosowanie formularzy admina

Browse/edit your models!

Django administration Welcome, adrian. Documentation / Change password / Log out

Site administration

Auth	Add Change
Groups	Add Change
Users	Add Change
Sites	Add Change
Sites	Add Change
Polls	Add Change
Polls	Add Change

Recent Actions

My Actions
None available

Django administration Welcome, adrian. Documentation / Change password / Log out

Home > Polls

Select poll to change Add poll. +

Poll

What's up?

1 poll

Django administration Welcome, adrian. Documentation / Change password / Log out

Home > Polls > What's up?

Change poll History

Question:

Date published: Date: 2005-04-01 Today Calendar

Time: 00:00:00 Now Clock

[Delete](#) [Save and add another](#) [Save and continue editing](#) [Save](#)

Wygląd interfejsu admin

Select user to change

Action: 0 of 3 selected

☐	USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
☐	adrian	adrian@example.com	Adrian	Holovaty	
☐	jacob	jacob@example.com	Jacob	Kaplan-Moss	
☐	simon	simon@example.com	Simon	Willison	

3 users

ADD USER +

FILTER

By staff status

All

Yes

No

By superuser status

All

Yes

No

By active

All

Yes

No

Wygląd interfejsu admin

Add flat page

URL:

Example: '/about/contact/'. Make sure to have leading and trailing slashes.

Title:

Content:

Sites:

example.com



Hold down "Control", or "Command" on a Mac, to select more than one.

Opcje dostosowywania

- 1 ModelAdmin media
- 2 Niestandardowe templatki
- 3 ModelAdmin/ModelForm hacking
- 4 Niestandardowe funkcje widoków

ModelAdmin Media

Dodawanie niestandardowych mediów

Dodaj media w ten sam sposób jak w przypadku klasy `Form`:

```
1 class ArticleAdmin(admin.ModelAdmin):  
2     class Media:  
3         css = {  
4             "all": ("my_styles.css",)  
5         }  
6         js = ("my_code.js",)
```

Należy pamiętać, że będą one poprzedzane `MEDIA_URL`

Przykłady ModelAdmin Media

- JavaScript
 - WYSIWYG Editor (e.g. TinyMCE)
 - AJAX
 - Fancy Inlines (drag and drop, dynamic add/delete)
 - <http://tinyurl.com/add-remove-inlines>
 - <http://www.djangosnippets.org/snippets/1053/>
 - Inject HTML
- CSS
 - Colors
 - Layout

Za i przeciw ModelAdmin Media

- Za
 - Łatwe do jednorazowych projektów
- Przeciw
 - Wymaga JavaScript
 - Działa tylko dla ChangeForm
 - Trudne w przypadku aplikacji wielokrotnego użytku

Niestandardowe templatki

Niestandardowe templatki

- `django.contrib.admin` jest "aplikacją wielokrotnego użytku"
- Kluczowe templatki
 - `admin/base.html`
 - `admin/index.html`
 - `admin/change_form.html`
 - `admin/change_list.html`

Templatki dla Project/App/Model

Templatki mogą być nadpisane:

- W ramach całego projektu:
`admin/change_form.html`
- W ramach aplikacji:
`admin/<my_app>/change_form.html`
- Dla konkretnego modelu:
`admin/<my_app>/<my_model>/change_form.html`

Można również zastąpić niektóre templatki poprzez ustawienie odpowiednich atrybutów `ModelAdmin`: `change_form_template`, `change_list_template`, `delete_confirmation_template`, `delete_selected_confirmation_template`, `object_history_template`

Przykład niestandardowej templatki

```
1 {% extends "admin/change_list.html" %}
2 {% block object-tools %}
3     <h1 class="errornote">
4         Look Here!
5     </h1>
6     {{ block.super }}
7 {% endblock %}
```

Django administration

Home > Demo_app > Posts

Select post to change

 Look Here!

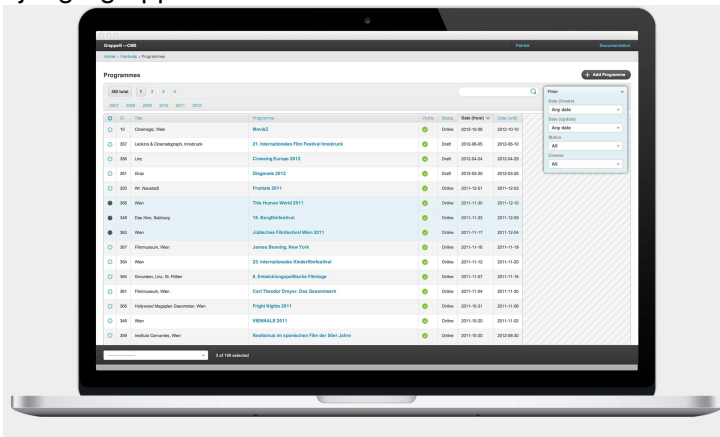
0 posts

Porady do niestandardowych tamplatek

- Rozszerzaj, nie zastępuj
- Użyj `{{ block.super }}` do rozszerzenia bloku
- Jeśli potrzebujesz dodatkowych bloków, utwórz własny szablon, a następnie rozszerz go.
- Rozszerz `{% extrahead block %}` w `base.html` do mediów w całym adminie

Przykłady niestandardowych templatek

django-grappelli



<https://grappelliproject.com/>

Przykłady niestandardowych templatek

django-suit

The screenshot displays the Django Suit admin interface. The top navigation bar includes the 'Django Suit' logo, the current date and time (Friday, 7th May 2021, 16:13), the site URL (Djangosuit.com), and user information (Welcome, Demo). The left sidebar contains a menu with 'Home', 'Examples', 'Countries', 'Continents', 'Kitchen sinks', 'Integrations', 'Django CMS', 'Filter', 'Auth', and 'Custom view'. The 'Examples' item is selected. The main content area shows the 'Kitchen sinks' view, which includes a search bar, a table of data, and a 'Add kitchen sink' button. The table has columns for 'Name', 'Help text', 'Choices', 'Horizontal choices', and 'Boolean'. The data rows are: 'Testing sinks' (Fridge and microwave), 'One more time' (Same text), and 'Mega form entry' (Some help or whatever). The bottom of the interface shows the 'Django Suit v0.2.12' version and copyright information.

Django Suit

Friday, 7th May 2021 16:13

Djangosuit.com

Welcome, Demo

Change password | Log out

Home > Examples > Kitchen sinks

Keyword: Choices: Date: Country: Search

Add kitchen sink

☐	Name	Help text	Choices	Horizontal choices	Boolean
☐	Testing sinks	Fridge and microwave	Short	Awesome	☒
☐	One more time	Same text	Short	Awesome	☒
☐	Mega form entry	Some help or whatever	Short	Good	☒

0 of 3 selected

1 - 3 / 3 kitchen sinks

Go

Save

Support Licence Report a bug

Django Suit v0.2.12

Copyright © 2013 Djangosuit.com
Developed by Djangosuit.com

<https://djangosuit.com/>

Za i przeciw niestandardowym templatkom

- Za
 - Łatwość zastosowania
 - Dotyczą każdego widoku administratora
 - Brak dodatkowych prac w kierunku wielokrotnego wykorzystanie aplikacji
- Przeciw
 - Głównie kosmetyczne zmiany, nie funkcjonalne

ModelAdmin/ModelForm hacking

Opcje ModelAdmin

- Wiele opcji w ModelAdmin może być zastąpionych własnymi ustawieniami
- Wiele modyfikacji jest możliwych bez ingerowania w kodzie źródłowym
- Zmiana domyślnych fields/fieldsets w "change views" (`fields`, `fieldsets`)
- Zmiana domyślnych opcji wyszukiwania fields/filters w "list views" (`list_filter`, `search_fields`)
- Zmiana domyślnych kontroltek (`filter_horizontal`, `filter_vertical`)
- Zmiana domyślnego sortowania (`ordering`)
- Edycja powiązanych modeli (`inlines`)
- ... i wiele więcej!

Opcje ModelAdmin

```
1 class MyUserAdmin(ModelAdmin):
2     list_display = ('username', 'name', 'account_type', 'is_active', 'number_of_visits')
3     list_filter = ('account_type', 'research_groups', 'is_active')
4     fieldsets = (
5         (None, {'fields': ('username', 'password',)}),
6         (_('Personal data'), {'fields': ('name', 'date_of_birth', 'email',)}),
7         (_('Permissions'), {'fields': ('account_type', 'research_groups', 'is_active',)}),
8         (_('Login information'), {'fields': ('last_login', 'number_of_visits',)}),
9     )
10    readonly_fields = ('last_login', 'number_of_visits',)
11    search_fields = ('username', 'name',)
12    ordering = ('username',)
13    filter_horizontal = ()
14
15    admin.site.register(MyUser, MyUserAdmin)
```

Przerejestrowanie ModelAdmin

Rozszerzenie podklasy ModelAdmin z innych aplikacji jest proste, np. `django.contrib.auth`:

```
1 from django.contrib import admin
2 from django.contrib.auth.admin import UserAdmin
3 from demo_app.models import UserProfile
4
5 class UserProfileInline(admin.TabularInline):
6     model = UserProfile
7     fk_name = 'user'
8     max_num = 1
9
10 class CustomUserAdmin(UserAdmin):
11     inlines = [UserProfileInline, ]
12
13 admin.site.unregister(User)
14 admin.site.register(User, CustomUserAdmin)
```

Metody w ModelAdmin

- Istnieje szereg metod w ModelAdmin, które są wykorzystywane do przetwarzania żądań
- Możliwe jest rozszerzenie tych metod, aby dodać określone funkcje
- Można dodawać atrybuty do `request` w czasie wykonywania
- Metody używane do renderowania strony administratora:
 - `ModelAdmin.add_view(self, request, form_url="", extra_context=None)`
 - `ModelAdmin.change_view(self, request, object_id, extra_context=None)`
 - `ModelAdmin.changelist_view(self, request, extra_context=None)`
 - `ModelAdmin.delete_view(self, request, object_id, extra_context=None)`
 - `ModelAdmin.history_view(self, request, object_id, extra_context=None)`

Rozszerzanie danych kontekstowych

```
1 class MyModelAdmin(admin.ModelAdmin):
2
3     # A template for a very customized change view:
4     change_form_template = 'admin/myapp/extras/openstreetmap_change_form.html'
5
6     def get_osm_info(self):
7         # ...
8
9     def change_view(self, request, object_id, extra_context=None):
10         my_context = {
11             'osm_data': self.get_osm_info(),
12         }
13         return super(MyModelAdmin, self).change_view(request, object_id,
14                                                         extra_context=my_context)
```

Uprawnienia na poziomie wiersza

```
1 class ArticleAdmin(admin.ModelAdmin):
2
3     def save_model(self, request, obj, form, change):
4         obj.user = request.user
5         obj.save()
6
7     def queryset(self, request):
8         qs = self.model._default_manager.filter(user=request.user)
9         return qs
```

ModelForms

- Wiele z funkcjonalności ModelAdmin jest "wrapperem" dla ModelForm
- Jeśli czegoś nie można tego zrobić w ModelAdmin, są szanse, że można to zrobić w ModelForm

Przykład ModelForms

```
1 class AuthorForm(forms.ModelForm):
2     exclude_states = ['AS', 'GU', 'MP', 'VI',]
3
4     def __init__(self, *args, **kwargs):
5         super(AuthorForm, self).__init__(*args, **kwargs)
6         w = self.fields['state'].widget
7         choices = []
8         for key, value in w.choices:
9             if key not in self.exclude_states:
10                 choices.append((key, value))
11         w.choices = choices
12
13 class AuthorAdmin(admin.ModelAdmin):
14     form = AuthorForm
```

Porady odnośnie ModelAdmin/ModelForm

- Im dalej kopiesz, mniej dokumentacji znajdziesz
- Nie bój się przeglądać źródeł:
 - `django.contrib.admin.sites.AdminSite`
 - `django.contrib.admin.options.ModelAdmin`
 - `django.forms.models.ModelForm`
 - `django.contrib.admin.options.InlineModelAdmin`
 - `django.forms.formsets`
- Używaj debuggera (`ipdb.set_trace()`)

Za i przeciw ModelAdmin/ModelForm

- Za
 - Elastyczne
 - Skuteczne
 - Brak dodatkowych prac w kierunku wielokrotnego wykorzystanie aplikacji
- Przeciw
 - Szybko robi się skomplikowane
 - Może wymagać zapoznania się z nieudokumentowanymi wewnętrznymi elementami Django

Niestandardowe widoki

Dodawanie niestandardowych widoków administratora

- Administrator nie został zbudowany, po to aby zrobić tylko kilka gotowych rzeczy
- Jego podstawowe funkcje mogą zostać rozszerzone
- Zbuduj swój własny widok i podłącz go do administratora!

Wystarczy zastąpić metodę `get_urls()` w podklasie `ModelAdmin`:

```
1 class MyModelAdmin(admin.ModelAdmin):
2     def get_urls(self):
3         urls = super().get_urls()
4         my_urls = [
5             path('my_view/', self.admin_site.admin_view(self.my_view))
6         ]
7         return my_urls + urls
```

Niestandardowy widok niestandardowy jest zabezpieczony przed nieuprawnionym dostępem.

Dodawanie niestandardowych widoków administratora

Nie bój się korzystać w swoim kodzie z domyślnych widoków admina!

```
1 from django.contrib import admin
2 from django.template.response import TemplateResponse
3 from django.urls import path
4
5 class MyModelAdmin(admin.ModelAdmin):
6
7     def my_view(self, request):
8         # ...
9         context = dict(
10             # Include common variables for rendering the admin template.
11             self.admin_site.each_context(request),
12             # Anything else you want in the context...
13             key=value,
14         )
15         return TemplateResponse(request, "somemplate.html", context)
```

Templatka niestandardowego widoku

```
1 {% extends "admin/base_site.html" %}
2 {% load i18n %}
3 {% block breadcrumbs %}
4 <div class="breadcrumbs">
5   <a href="../../"/>{% trans "Home" %}</a> &rsquo;
6   <a href="../../">{{ app_label|capfirst|escape }}</a> &rsquo;
7   {% if has_change_permission %}<a
8     href="../">{{ opts.verbose_name_plural|capfirst }}</a>
9   {% else %}
10     {{ opts.verbose_name_plural|capfirst }}
11   {% endif %} &rsquo; My Custom View
12 </div>
13 {% endblock %}
14 {% block content %}
15 <!-- do stuff here -->
16 {% endblock %}
```

Za i przeciw niestandardowym widokom

- Za

- Jeszcze bardziej elastyczne
- Jeszcze bardziej skuteczne
- Brak dodatkowych prac w kierunku wielokrotnego wykorzystanie aplikacji

- Przeciw

- Może być trudne do integracji w workflow
- Na własną rękę, musisz sprawdzać poprawność formularza, budować szablony, itp.

Rails admin

Wprowadzenie

- Brak specjalnego modułu administracyjnego wbudowanego we framework
- Scaffolding pozwala na generowanie prostych stron administracyjnych
- Wiele dodatkowych bibliotek realizujących moduły administracyjne

Rails admin - przykłady

ActiveAdmin

ADMIN

Manage Users

New User

Import Users

Batch Actions ▾

☐	Name	Email	Created Date	Officephone	Cellphone	Picture	Groups
☐	Mr. Superman	super@man.com	2013-01-26 15:50:23 UTC			1359215422	0 Admin 0 Following
☐	Ben Cohen	ben@careerbuilder.com	2013-01-07 09:26:03 UTC			1357550762	0 Admin 0 Following
☐	Will Hooper	will@careerbuilder.com	2013-01-07 09:22:09 UTC		03344298234	1357550528	3 Admin 22 Following
☐	Mr. Ironman	sayyam.mehmood@gmail.com	2013-01-01 15:59:02 UTC			1359264621	2 Admin 39 Following
☐	Mr. Batman	batman@gotham.com	2012-12-31 15:29:53 UTC	123456789	123456789	1357581322	3 Admin 22 Following

Download: [CSV](#) [XML](#) [JSON](#)

Deploying all 5 Users

Filters

SEARCH ALL

SEARCH NAME

SEARCH EMAIL

CREATED DATE
 -

SEARCH OFFICEPHONE

SEARCH CELLPHONE

<http://activeadmin.info/>

Rails admin - przykłady

Typus

Typus

Dashboard Admin CRUD CRUD Extended HasManyThrough HasOne MongoDB Polymorphic STI

Welcome!

This is the collection of models, organized by functionality, I use to test the typus **core**.

In some models appears an explanation of the customization added like for example: filters, search, sidebars ...

Code for this application is available at [GitHub](#).

If you need help don't hesitate in joining the [mailing list](#) or use the [help](#).

Dashboard

Admin

Typus users

CRUD

Entries

CRUD Extended

Assets

Categories

Comments

Pages

Posts

HasManyThrough

Projects

Project collaborators

Users

<http://docs.typuscmf.com>

Rails admin - przykłady

RailsAdmin

Dummy App Admin Dashboard Home LOG OUT

Navigation

- Dashboard
- Balls
- Basic pages
- Comments
- Divisions
- Drafts
- Fans
- Field tests
- Leagues
- Nested field tests
- Players
- Teams
- Users

Site administration

Dashboard

Model name	Last used	Records	
Balls	about 5 hours ago	40	Add new History Export
Basic pages	1 day ago	6	Add new History Export
Comments	17 minutes ago	12	Add new History Export
Divisions	1 day ago	13	Add new History Export
Drafts	1 day ago	4	Add new History Export
Fans	4 days ago	5	Add new History Export
Field tests	17 minutes ago	8	Add new History Export
Leagues	1 day ago	20	Add new History Export
Nested field tests	18 minutes ago	1	Add new History Export
Players	1 day ago	1143	Add new History Export
Teams	3 days ago	28	Add new History Export
Users		2	Add new History Export

https://github.com/sferik/rails_admin