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.

Moduł administracyjny: Agenda

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

Django admin

Podstawy

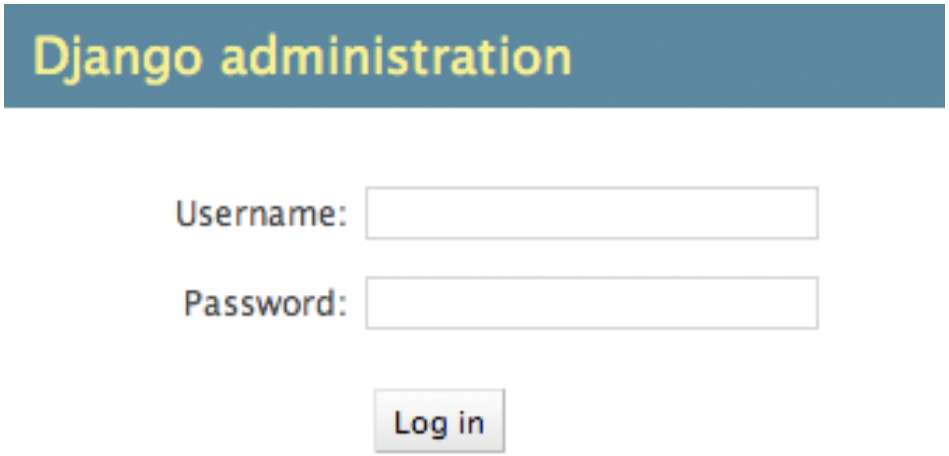
Aktywacja strony admina

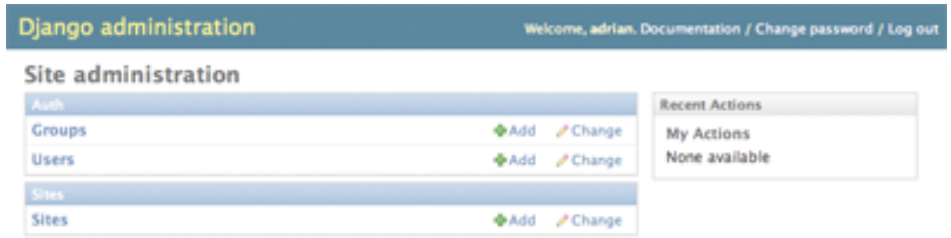
- Moduł admin jest włączony w domyślnym szablonie projektu używanym przez `startproject`.
- Aplikacja `django.contrib.admin` w `INSTALLED_APPS` w `settings.py`
- Wpis w `urls.py`

```
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 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!





Rejestracja modeli i dostosowanie formularzy admina

- Plik `admin.py` w katalogu aplikacji:

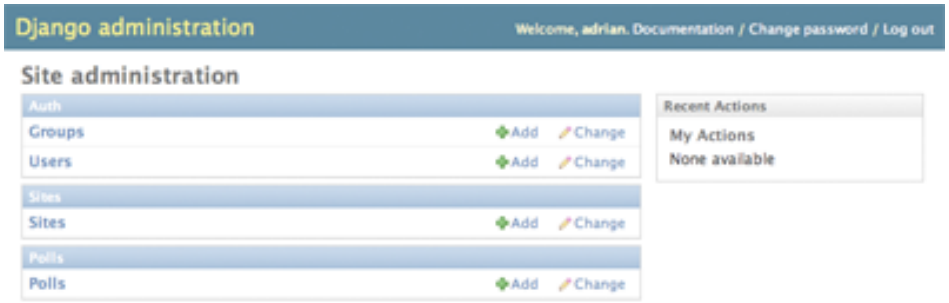
```
1  from django.contrib import admin
2  from myproject.myapp.models import Author
3
4  admin.site.register(Author)
```

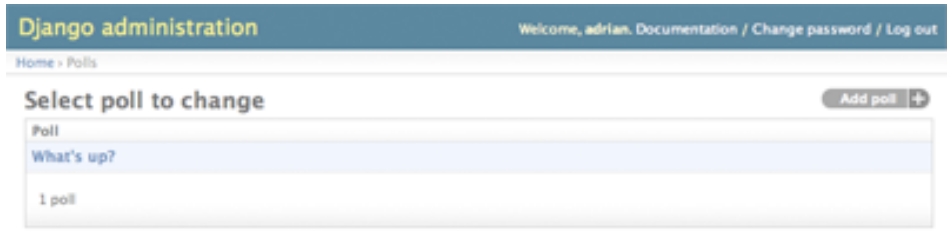
- Dostosowanie formularza admina:

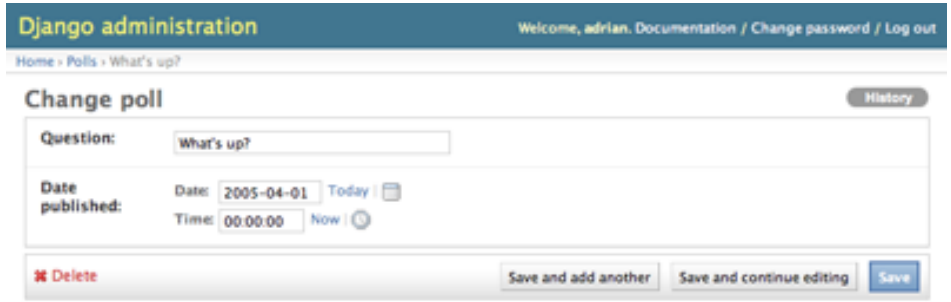
```
1  from django.contrib import admin
2  from myproject.myapp.models import Author
3
4  class AuthorAdmin(admin.ModelAdmin):
5      pass
6  admin.site.register(Author, AuthorAdmin)
7
8  @admin.register(Author)
9  class AuthorAdmin(admin.ModelAdmin):
10     pass
```

Rejestracja modeli i dostosowanie formularzy admina

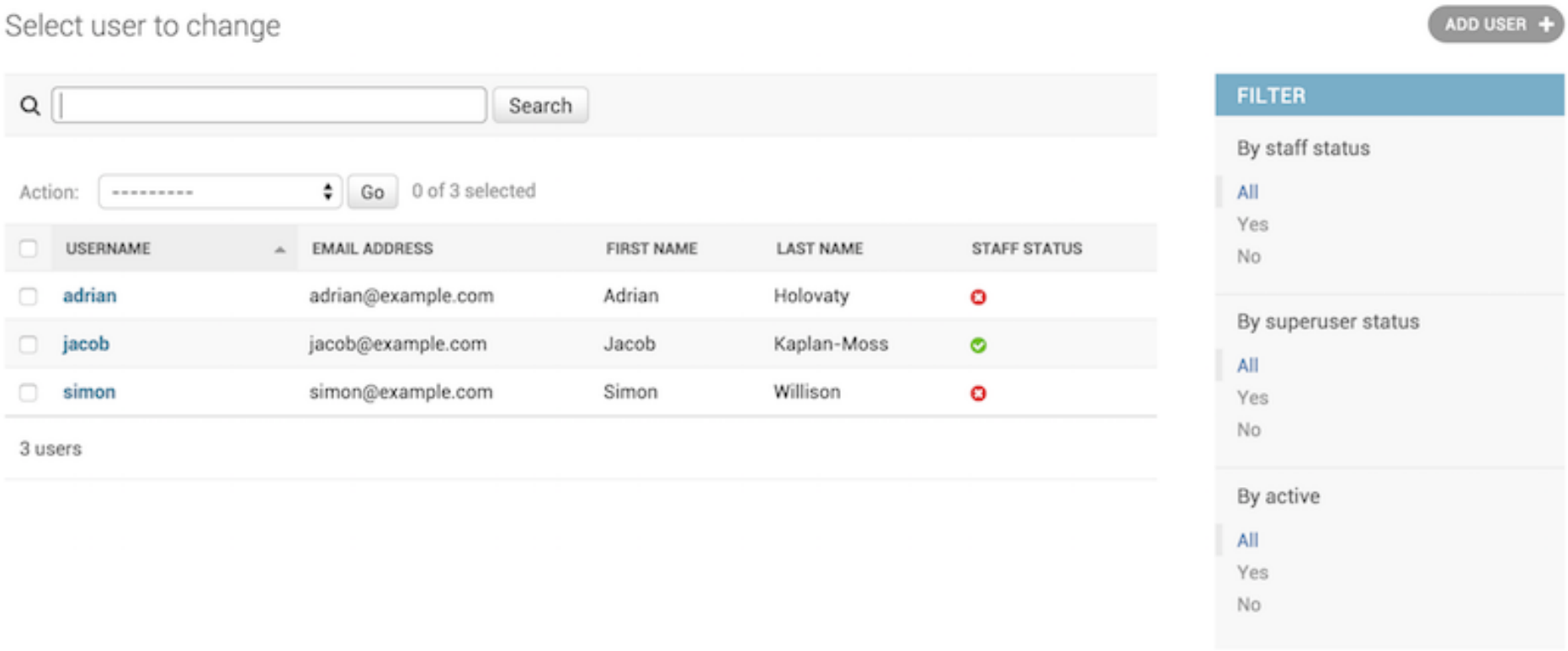
Browse/edit your models!







Wygląd interfejsu admin



Wygląd interfejsu admin

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

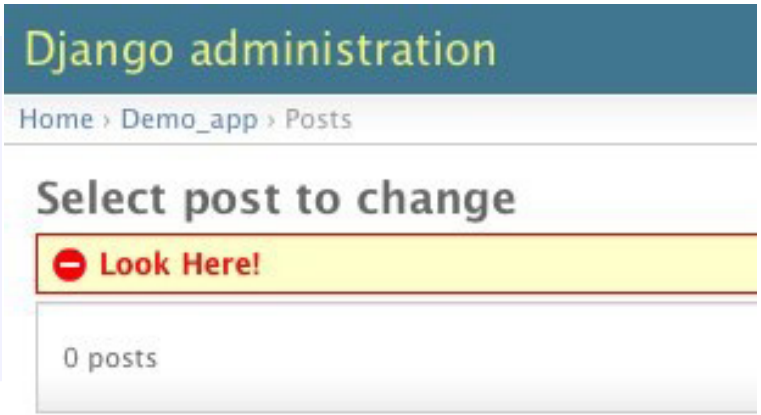
Templatki mogą być nadpisane:

- 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 %}
```

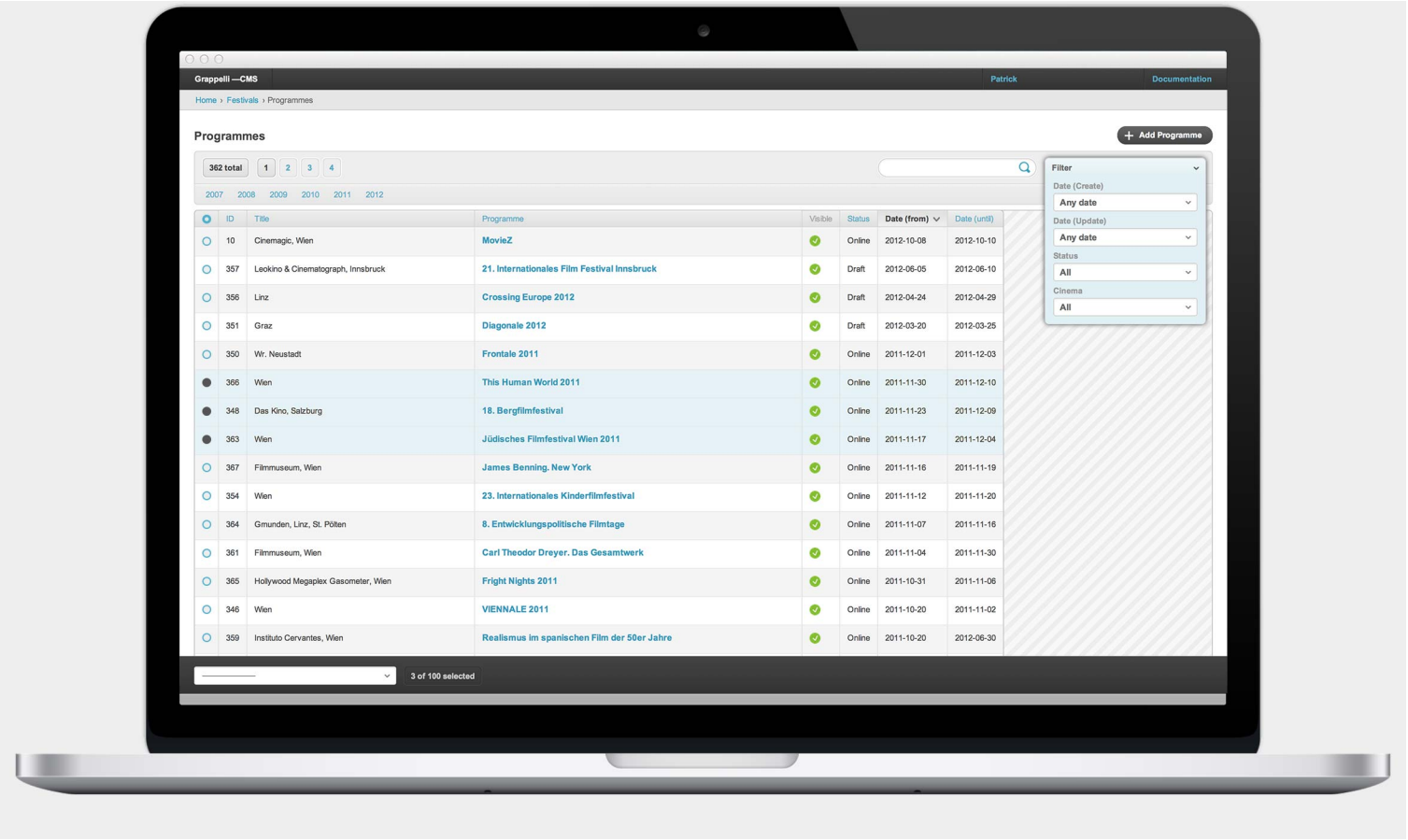


Porady do niestandardowych tamplatek

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

Przykłady niestandardowych templatek

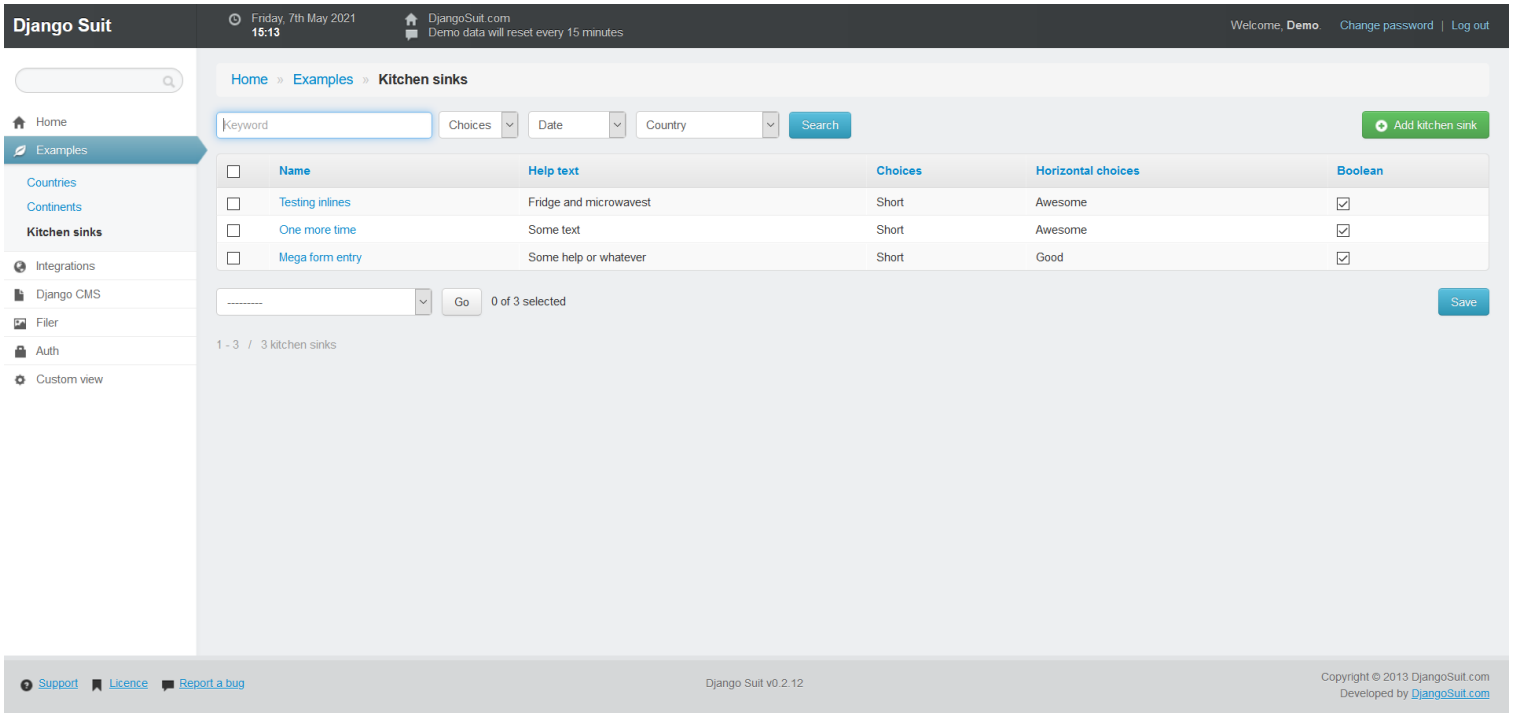
django-grappelli



https://grappelliproject.com/

Przykłady niestandardowych templatek

django-suit



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 kontrolek (`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_vi
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, "sometemplate.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> &rsaquo;
6    <a href="../../">{{ app_label|capfirst|escape }}</a> &rsaquo;
7    {% if has_change_permission %}<a
8        href="../">{{ opts.verbose_name_plural|capfirst }}</a>
9    {% else %}
10       {{ opts.verbose_name_plural|capfirst }}
11   {% endif %} &rsaquo; 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.

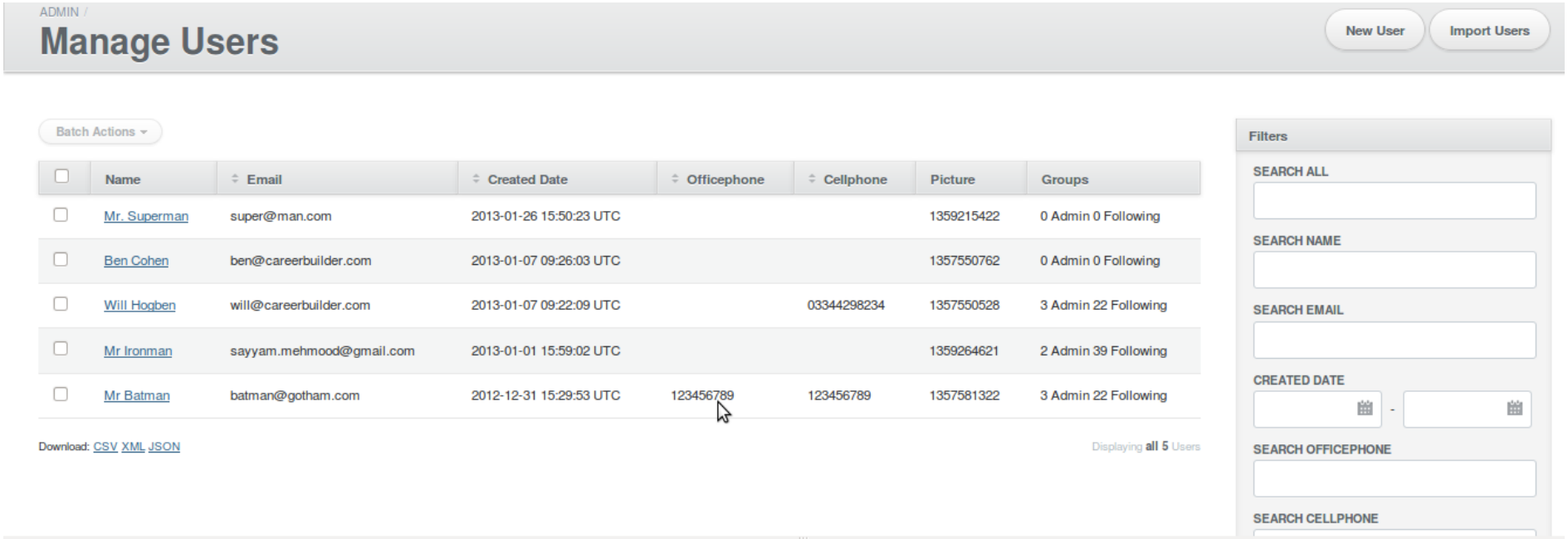
Rails admin

Wprowadzenie

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

Rails admin - przykłady

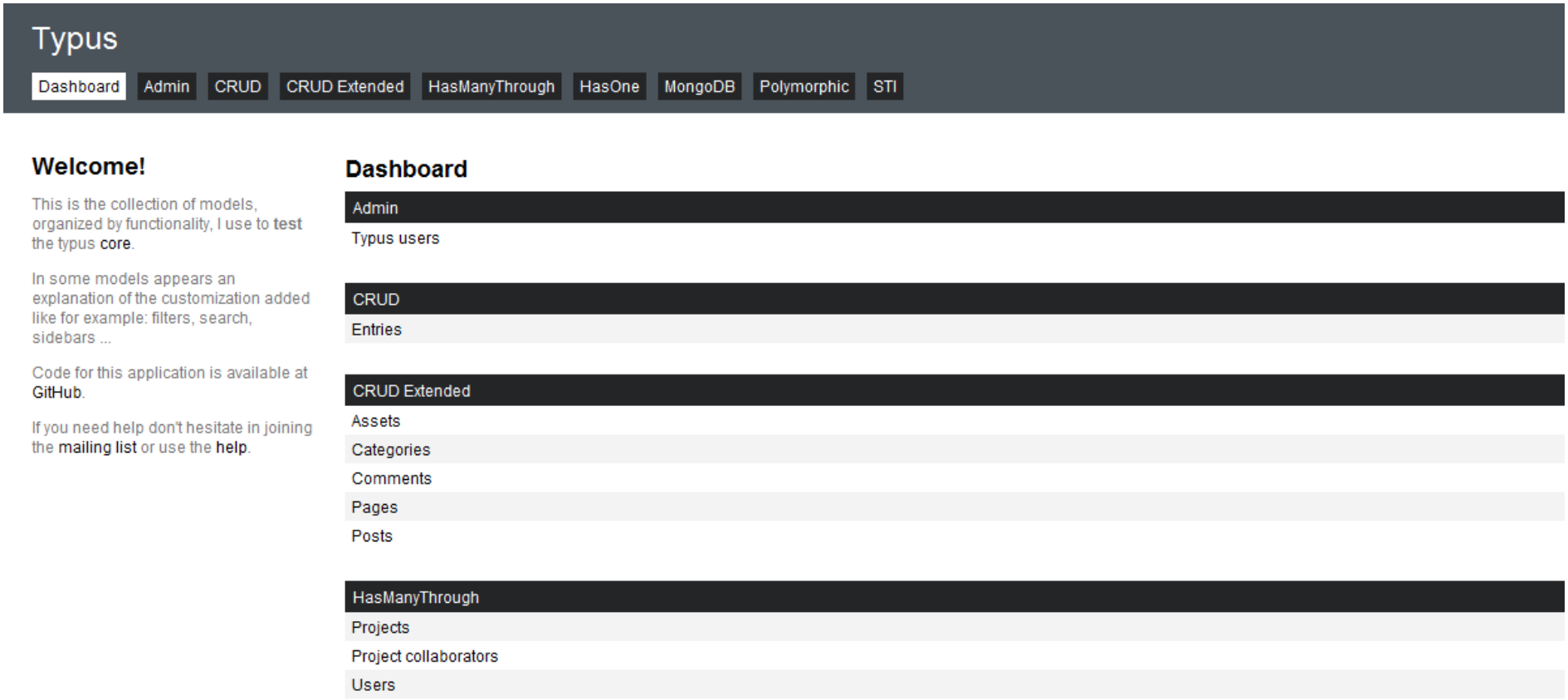
ActiveAdmin



http://activeadmin.info/

Rails admin - przykłady

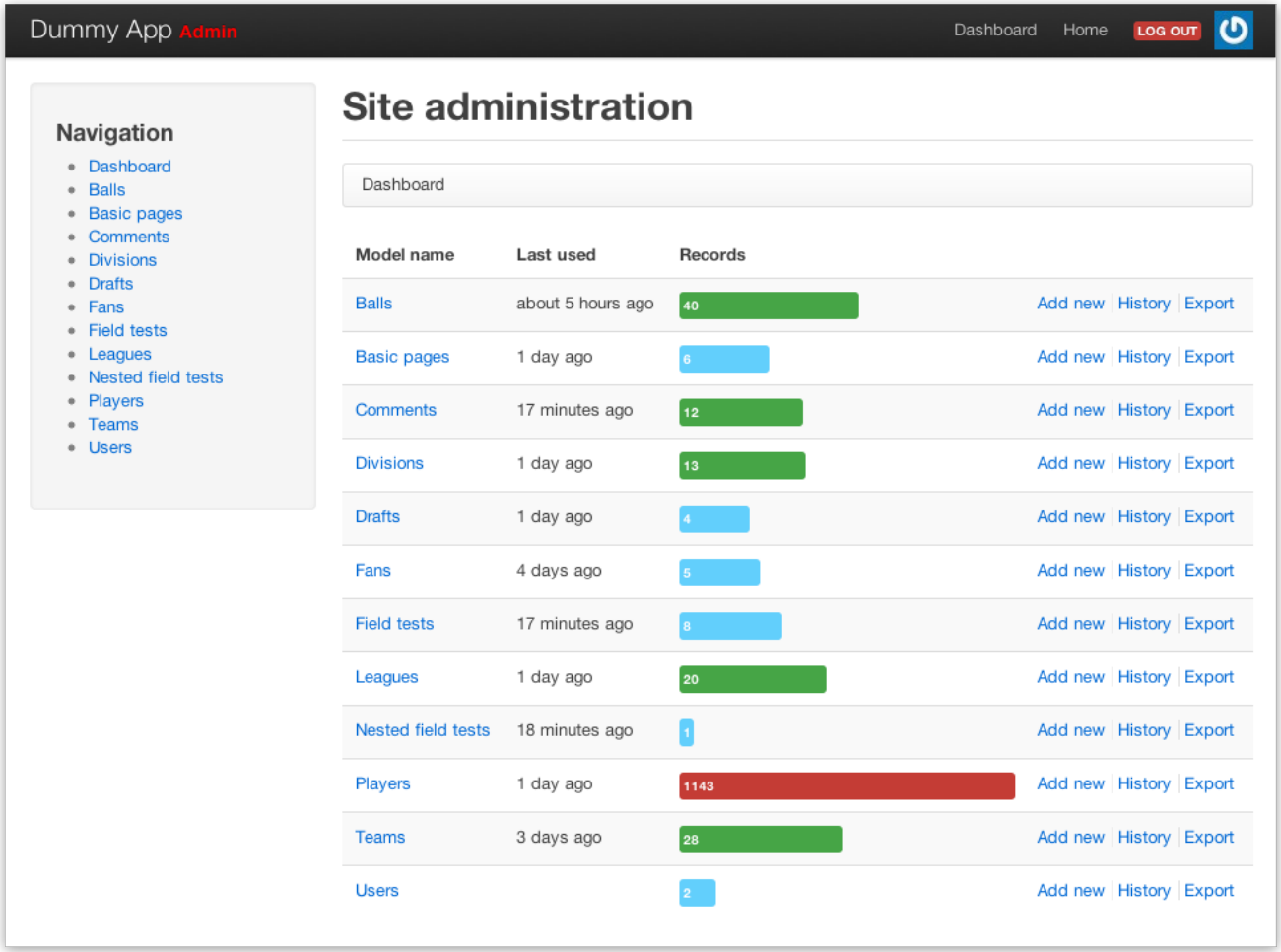
Typus



http://docs.typuscmf.com

Rails admin - przykłady

RailsAdmin



https://github.com/sferik/rails_admin