Dydaktyka Informatyki 11(2016)
ISSN 2083-3156
DOI: 10.15584/di.2016.11.19
http://www.di.univ.rzeszow.pl

Wydział
Matematyczno-Przyrodniczy UR
Laboratorium Zagadnień
Społeczeństwa Informacyjnego

Jacek WOŁOSZYN

Dr inż., Uniwersytet Technologiczno-Humanistyczny w Radomiu, Wydział Informatyki i Matematyki, Katedra Informatyki, ul. Malczewskiego 29, 26-600 Radom; jacek@delta.pl

UKRYWANIE PÓL W FORMULARZU W DJANGO Z WYKORZYSTANIEM JQUERY

HIDING FIELDS IN THE FORM IN DJANGO USING JQUERY

Słowa kluczowe: Django, jQuery, formularz, framework.
Keywords: Django, jQuery, form, framework.

Streszczenie

Budowa aplikacji z wykorzystaniem frameworku jest efektywnym sposobem na szybkie stworzenie aplikacji bazodanowej. Często się jednak zdarza, że rozwiązania proponowane przez zaimplementowane procedury generujące kod wynikowy nie spełniają założeń, które wymaga aplikacja. Przykład opisuje odmienny sposób podejścia do rozwiązania problemu.

Summary

Construction applications using the framework is an effective way to quickly create a database application. It often happens that the solutions proposed by the implemented procedures for generating the object code does not meet the assumptions that requires the application. The example describes a different approach to solve the problem.

Wstęp

Frameworki są doskonałymi narzędziami, za pomocą których można sprawnie napisać aplikację. Wykorzystywanie ich w codziennej pracy skutkuje oszczędnością czasu, który należałoby poświęcić na pisanie funkcji czy procedur związanych z typowymi ścieżkami budowy aplikacji. Często jednak zdarza się, że ogólnie wygenerowany kod nie sprawdza się w rozwiązaniu problemu, nad którym pracujemy. Frameworki są cały czas rozbudowywane o nowe funkcjonalności i zgłaszane na bieżąco problemy, ale nowe wersje pojawiają się zazwyczaj dużo później niż oczekiwana data powstania naszej aplikacji. Dlatego

w artykule tym pokazano inny sposób podejścia do rozwiązania problemu. Do jego uzyskania zastosowano bibliotekę jQuery, która pozwala uzyskać oczekiwany rezultat bez ingerencji w moduły frameworka.

Opis problemu

Framework Django[1] pozwala na szybką budowę aplikacji bazodanowej. Jest to zaawansowane narzędzie posiadające zaimplementowane wysokowydajne procedury i funkcje pozwalające wykonać w sposób szybki oraz profesjonalny postawione zadanie. Często jednak zdarzają się problemy, których nie jest w prosty sposób rozwiązać bez ingerencji w kod wynikowy frameworka lub zaimportowanego modułu. Jednym z takich przykładów jest zapis do bazy danych z automatycznie wygenerowanego formularza z pominięciem wybranego pola. Przedstawiony przykład polega na usunięciu z formularza automatycznie wygenerowanego przez Django formularza wybranego pola.

Zastosowanie biblioteki jQuery opartej na JavaScript do rozwiązania tego problemu umożliwia brak ingerencji w samym kodzie źródłowym.

Mechanizm działania aplikacji

Aplikacja jest typowym programem bazodanowym opartym na frameworku Django. Działa ona w oparciu o typowy mechanizm M-V-C. W skrócie po wywołaniu odpowiedniego adresu url w przeglądarce, zostaje wywołany wskazujący na niego widok, w którym wykonywane są funkcje. Wyniki tych operacji zapisywane są do zmiennej, która zostaje przekazana do odpowiedniego szablonu html, a on wyświetla zawartość zmiennych. We fragmencie przedstawionego przykładu jeden z widoków wyświetla formularz wygenerowany automatycznie na podstawie modelu, co znakomicie skraca czas pracy nad formularzem.

Fragment listingu modelu przedstawia się następująco:

```
przed= models.ForeignKey(Przedmiot, verbose_name="Przedmiot")
user= models.ForeignKey(User, verbose_name="Student")
data_cw  = models.DateField(verbose_name="Data wykonania ćwiczenia")
data_spr=models.DateTimeField(auto_now_add=True,verbose_name="Data oddania sprawozdania")
tytul_sk=models.CharField(max_length=20,choices=CW,verbose_name='Skrócona nazwa ćwiczenia')
tytul=models.CharField(max_length=150, verbose_name='Tytuł ćwiczenia')
cel = models.CharField(max_length=250, verbose_name='Cel ćwiczenia')
teoria=models.TextField(blank=True,null=True, verbose_name='Teoria')
przebieg=models.TextField(blank=True,null=True, verbose_name='Przebieg ćwiczenia')
rezultat=models.TextField(blank=True,null=True, verbose_name='Uzyskane rezultaty')
```

[1] J. Elman, M. Lavin, *Lightweight Django*, O'Reilly 2015; A. Ravindran, *Django Design Patterns nad Best Practices*, Packt Publishing 2015.

```
wnioski=models.TextField(blank=True,null=True, verbose_name='Wnioski')
uwagi= models.TextField(blank=True,null=True, verbose_name='Uwagi')
ocena= models.IntegerField(blank=True,null=True, verbose_name='Ocena')
```

Wynikiem działania funkcji, na podstawie definicji pól modelu oraz definicji samego formularza w pliku projektu form.py jest wyświetlony jak na rys. 1 formularz.

Rys. 1. Fragment wygenerowanego formularza

```
class SprawozdanieForm(forms.ModelForm):
   class Meta:
      model = Sprawozdanie
      fields = '__all__'
```

Listing 1. Fragment zawartości pliku form.py

Wynik pracy jest jak najbardziej zadowalający, ale powstał pewien problem. Zgodnie z założeniami przyjętymi przez twórców frameworka wyświetlone zostało również pole student.

Pole to nie powinno być wygenerowane, ponieważ zalogowany użytkownik będzie miał możliwość zapisu formularza w imieniu innego dowolnie wybranego użytkownika z listy, na co my absolutnie się nie zgadzamy. Jest to zdarzenie niepożądane.

Rozwiązanie problemu z wykorzystaniem bliblioteki jQuery

Powstaje zatem pytanie, co zmienić w kodzie aplikacji, aby rozwiązać powstałą niedogodność?

Najprościej byłoby w pliku forms.py dodać listę pól wykluczonych z edycji jak pokazano poniżej.

```
exclude = ['user']
```

Jednak takie rozwiązanie nie przynosi oczekiwanych rezultatów, gdyż automatyczny mechanizm walidacji pól formularzy podczas zapisu domaga się uzupełnienia tego pola. W tym przypadku niezbędna byłaby ingerencja w kod źródłowy pozwalająca rozwiązać ten problem lub napisać nowy formularz z odpowiednią obsługą walidacji[2]. Ale skoro używamy frameworka to po to, aby korzystać z jego możliwości generowania kodu w tym przypadku formularza.

Wydawałoby się, że ciekawym i skutecznym rozwiązaniem problemu byłoby wykorzystanie z modułu django-bootstrap3 ustawień bootstrap_field, a konkretnie ustawienia parametru set_disabled na wartość false.

```
{% bootstrap_field user_id ser_disabledl=False %}
```

Jednak po ustawieniu tego parametru w formularzu walidacja zgłasza znowu błąd. Taki sam efekt daje wyłączenie pola formularza jako pole edycyjne, czyli ustawień typu read_only. Pozostaje więc ingerencja w kod albo napisanie formularza ręcznie z odpowiednią walidacją. Ale czy po to stosujemy frameworki, aby pisać dodatkowy kod?

Innym rozwiązaniem, które się nasuwa, jest podejście do rozwiązania problemu ze strony frontendu, stosując odpowiednią bibliotekę. JQuery[3] jest to niewielka biblioteka wykorzystująca JavaScript. Takie rozwiązanie wydaje się być w tym przypadku odpowiednie, ponieważ nie ingeruje ono w żaden inwazyjny sposób w kod aplikacji i pozwala osiągnąć zamierzony efekt polegający na ukryciu pola.

Użycie tej biblioteki wymaga jedynie zarejestrowania jej w kodzie HTML na poziomie nagłówka <head> w sekcji <script> z wykorzystaniem dwóch atrybutów:

```
<script type=”text/javascript” src=”jquery-wersja_bilbl.min.js></script>
```

Dodatkowo należy w podobny sposób zarejestrować plik, w którym będą umieszczone skrypty realizujące zadanie ukrywania pól. Nazwa pliku może być dowolna, natomiast ważne jest, aby kończyła się rozszerzeniem js, odpowiadającej konwencji nazewnictwa przyjętej dla JavaScritpt.

We wnętrzu pliku należy umieścić zapisy jak poniżej. Pierwszy z nich ukrywa label, czyli nazwę pola, a drugi zapis już konkretne pole, w tym przypadku pole o nazwie id_user. Znak $ reprezentuje funkcję jQuery, a document informuje nas, że funkcja ma zastosowanie do całego dokumentu. Zastosowane

[2] A. Downey, *Python for Software Design*, Cambridge University Press 2009; A. Downey, *Think Python*, O'Reilly 2012; D. Hellman, *The Python Standard Library by Example*, Addison-Wesley 2011; Y. Hilpisch, *Derivatives Analytics with Python*, Wiley 2015.

[3] D. McFarland, *JavaScript i iQuery*, O'Reilly 2012; E. Sarrion, *jQuery UI*, O'Reilly 2012.

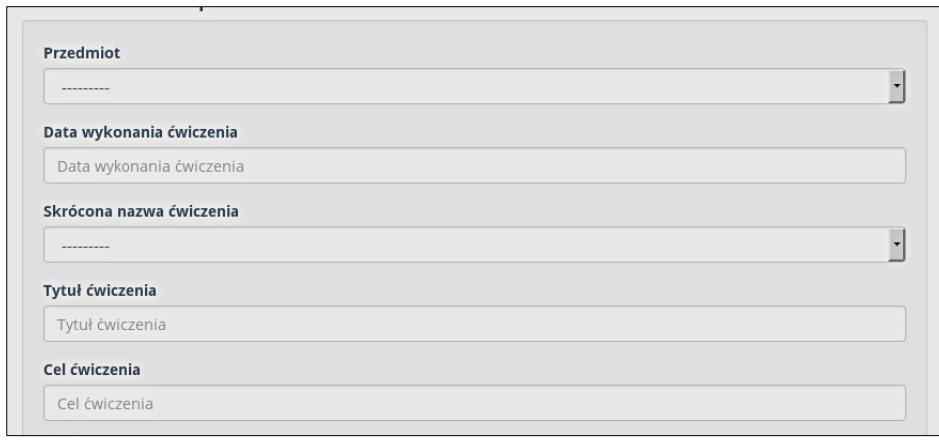
odpowiedniego selektora zapewnia jednoznaczne odwołanie się do pola label odpowiednio for=id_user oraz #id_user unikalnej nazwy pola w formularzu i zamianę attrybutu 'display' na 'none'.

```
$(document).ready(function() {
    $("label[for=id_user]").css("display", "none");
});

$(document).ready(function() {
    $("#id_user").css("display", "none");
});
```

Listing 2. Skrypty jQuery pozwalające ukryć pola w formularzu

Efektem tego działania jest formularz podobny jak na rys. 1, jednak tym razem niezawierający pola student, czyli jest w takiej postaci, jaka była oczekiwana w założeniach programu.



Rys. 2. Fragment wygenerowanego formularza bez pola user

Wnioski

Tempo pracy jest obecnie bardzo szybkie we wszystkich dziedzinach życia. Nie ma obecnie dziedziny przemysłu, która nie jest wspomagana przez systemy informatyczne. Szybka dynamika zmian wymusza stosowanie narzędzi informatycznych[4], które zapewniają wysoką niezawodność, dużą wydajność i elastycz-

[4] N. Gift, J. Jones, *Python for Unix and Linux system Administration*, O'Reilly 2008; M. Goodrich, R. Tamassia, M. Goldwasser, *Data Structures and Algorithms in Python*, Wiley 2013; J. Payne, *Beginning Python*, Wrox 2010; M. Summerfield, *Programming in Python 3*, Addison-Wesley 2010; T. Ziade, *Packt, Expert Python Programming*, Publishing 2008.

ność na wszelkie zmiany. Do tego doskonale nadają się frameworki, za pomocą których można szybko zbudować aplikację, która budowana w sposób tradycyjny byłaby budowana miesiącami, latami przez zespół specjalistów. Aczkolwiek stosując tego rodzaju rozwiązania musimy stosować się do pewnych wzorców i ograniczeń, które narzuca narzędzie. W sytuacjach, kiedy jest to wymagane, niezbędne jest napisanie funkcji realizującej algorytmy czy procedury w sposób tradcyjny. Zawsze należy też pomyśleć, czy problem można rozwiązać innym podejściem bez ingerencji w kod i niewielkim nakładem pracy, jak zaprezentowano w artykule.

Bibliografia

Downey A., *Python for Software Design*, Cambridge University Press 2009.
Downey A., *Think Python*, O'Reilly 2012.
Elman J., Lavin M., *Lightweight Django*, O'Reilly 2015.
Gift N., Jones J., *Python for Unix and Linux sysytem Administration*, O'Reilly 2008.
Goodrich M., Tamassia R., Goldwasser M., *Data Structures and Algorithms in Python*, Wiley 2013.
Hellman D., *The Python Standard Library by Example*, Addison-Wesley 2011.
Hilpisch Y., *Derivatives Analytics with Python*, Wiley 2015.
McFarland D., *JavaScript i iQuery*, O'Reilly 2012.
Payne J., *Beginning Python*, Wrox 2010.
Ravindran A., *Django Design Patterns nad Best Practices*, Packt Publishing 2015.
Sarrion E., *jQuery UI*, O'Reilly 2012.
Summerfield M., *Programming in Python 3*, Addison-Wesley 2010.
Ziade T., *Packt, Expert Python Programming*, Publishing 2008.