Technologia i rozwiązania

Android Studio
Podstawy

Najlepsze IDE dla programistów platformy Android!

Belén Cruz Zapata

Helion

[PACKT] open source
PUBLISHING
community experience distilled

Tytuł oryginału: Android Studio Essentials

Tłumaczenie: Piotr Rajca

ISBN: 978-83-283-1466-5

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/andrsp
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

4

ROZDZIAŁ

Korzystanie z edytora kodu

Skoro już utworzyliśmy nasz pierwszy projekt i umiemy poruszać się po różnych katalogach, podkatalogach i plikach, nadszedł czas, by zabrać się do programowania. Czy kiedykolwiek chcieliśmy programować efektywniej? W jaki sposób możemy przyspieszyć proces powstawania kodu? Czy przy okazji nie chcielibyśmy poznać użytecznych skrótów klawiszowych? Na przykład w jaki sposób można umieścić w komentarzu kilka wierszy kodu jednocześnie, jak odszukać i zastąpić łańcuch znaków, przechodzić szybciej pomiędzy poszczególnymi parametrami w wywołaniu metody?

W tym rozdziale zostaną przedstawione możliwości dostosowywania sposobu działania edytora kodu, dzięki którym będziemy mogli pracować bardziej komfortowo. Warto także poznać podstawowe możliwości edytora, by poprawić wydajność pracy. Poznamy automatyczne uzupełnianie oraz generowanie kodu. Na samym końcu przedstawione zostaną najbardziej użyteczne skróty klawiszowe, które pozwolą jeszcze bardziej przyspieszyć i usprawnić pracę.

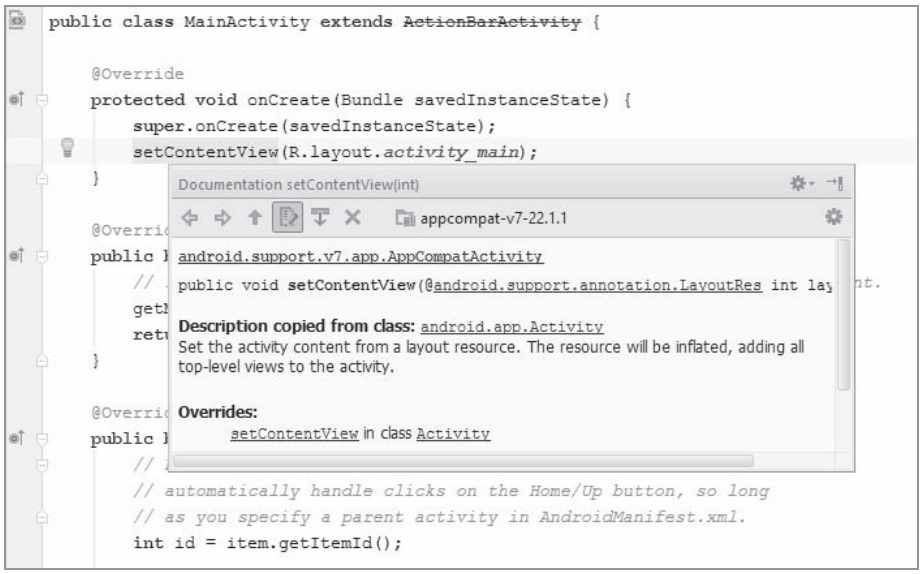
W tym rozdziale zostaną opisane następujące zagadnienia:

- Dostosowywanie działania edytora kodu
- Uzupełnianie kodu
- Generacja kodu
- Odnajdowanie powiązanych treści
- Przydatne skróty klawiaturowe

Dostosowywanie ustawień edytora

Aby wyświetlić ustawienia edytora, należy wybrać z menu głównego opcję *File/Settings*, a następnie w oknie dialogowym *Settings* rozwinąć opcję *Editor* dostępną na hierarchicznej liście z lewej strony okna i kliknąć opcję *General*. W rezultacie w panelu z prawej strony okna dialogowego wyświetlą się ogólne opcje edytora. Zachęcam do zaznaczenia dwóch z nich, które domyślnie nie są wybrane:

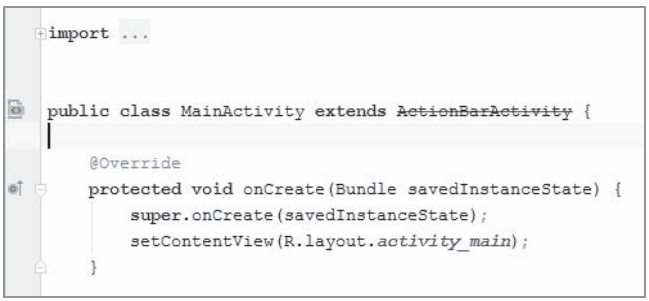
- *Change font size (Zoom) with Ctrl+Mouse Wheel* — zaznaczenie tej opcji pozwala zmieniać wielkość czcionki używanej w edytorze przy użyciu kółka myszy, podobnie jak można to robić w innych programach, takich jak przeglądarki WWW.
- *Show quick doc on mouse move* — zaznaczenie tej opcji pozwala na wyświetlanie krótkich informacji o kodzie w niewielkim oknie dialogowym, jeśli wskaźnik myszy zostanie umieszczony na wybranym fragmencie kodu na dłużej niż 500 milisekund. Po ponownym przesunięciu wskaźnika myszy okienko zostanie automatycznie ukryte, jeśli jednak wskaźnik zostanie przesunięty na okienko, to będzie można dokładnie przejrzeć całą zawartość prezentowanego w nim dokumentu. To, że można przeczytać informacje o przeznaczeniu metody oraz o jej parametrach bez odnajdowania jej w dokumentacji, jest niezwykle użyteczne. Przykładowa postać takich informacji została przedstawiona na rysunku 4.1.



Rysunek 4.1. Okienko dialogowe z informacjami o wybranej metodzie

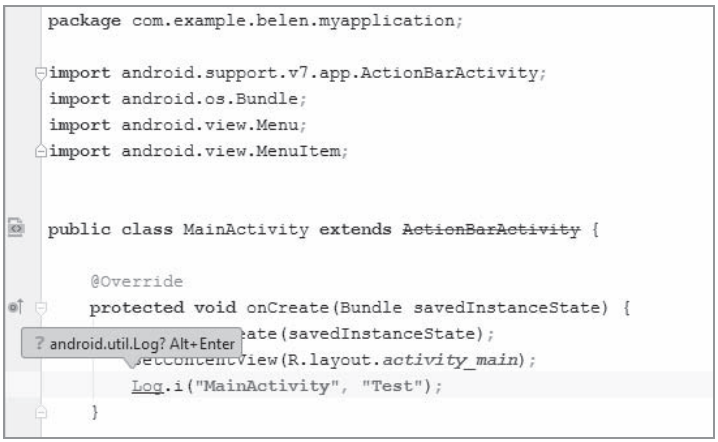
Okno dialogowe *Settings* udostępnia także wiele innych ustawień określających wygląd i sposób działania edytora. Są one dostępne za pośrednictwem kilkunastu opcji dostępnych na liście z lewej strony, w gałęzi opcji *Editor*. Poniżej zamieszczone zostały krótkie informacje na temat kilku wybranych spośród nich:

- *Smart Keys* — ta kategoria pozwala skonfigurować akcje, które będą wykonywane automatycznie podczas wpisywania kodu, takie jak: automatyczne dodawanie nawiasów zamykających, znaczników zamykających bądź automatyczne dodawanie wcięcia po naciśnięciu klawisza *Enter*.
- *Appearance* — ta kategoria zawiera opcje określające wygląd edytora. Można tu zażądać wyświetlania numeracji wierszy, różnego rodzaju ikon przekazujących dodatkowe informacje o kodzie, odstępów itd. Sugerowałbym zmianę dwóch ustawień, które domyślnie nie są zaznaczone:
  - *Show line numbers* — ta opcja pozwala wyświetlać numerację wierszy z lewej strony edytora. Numery wierszy są bardzo przydatne podczas debugowania programów oraz weryfikacji informacji zapisywanych w dzienniku aplikacji.
  - *Show method separators* — ta opcja pozwala na wizualne oddzielenie metod klasy.
- *Colors & Fonts* — ta grupa opcji konfiguracyjnych pozwala na określanie czcionek i kolorów. Jest ona bardzo obszerna i udostępnia możliwości określania postaci wielu różnych elementów (słów kluczowych, liczb, ostrzeżeń, błędów, komentarzy, łańcuchów znaków itd., a wszystko to z podziałem na różne języki programowania obsługiwane przez edytor). Wszystkie te ustawienia można zapisywać w formie schematów.
- *Editor Tabs* — ta kategoria pozwala konfigurować karty edytora. Sugeruję zaznaczenie opcji *Mark modified tabs with asterisk*, dzięki której z łatwością będzie można określić, które ze zmodyfikowanych plików jeszcze nie zostały zapisane.
- *Code Folding* — w tej kategorii są dostępne opcje związane z mechanizmem zwijania i rozwijania kodu. Zwijanie kodu (ang. *code folding*) pozwala na ukrywanie bloków kodu, które nie są aktualnie edytowane, dzięki czemu prezentowany kod staje się prostszy i czytelniejszy. Fragmenty kodu można zwijać, a następnie rozwijać, korzystając z ikon wyświetlanych w edytorze (patrz rysunek 4.2) lub przy użyciu opcji dostępnych w menu *Code/Folding*.



Rysunek 4.2. Ikony pozwalające na zwijanie i rozwijanie bloków kodu

- *Code Completion* — opcje dostępne w tej kategorii pozwalają na konfigurację mechanizmu uzupełniania kodu, który zostanie bardziej szczegółowo opisany w następnym podrozdziale.
- *Auto Import* — opcje dostępne w tej kategorii pozwalają skonfigurować działanie edytora w przypadku wklejania kodu korzystającego z klas, które nie zostały jeszcze zaimportowane w bieżącym pliku. Domyślnie w takim przypadku wyświetlane jest okienko dialogowe sugerujące dodanie odpowiednich instrukcji `import` (patrz rysunek 4.3). Jeśli jednak zostanie zaznaczona opcja *Add unambiguous imports on the fly*[1], to instrukcje `import` będą dodawane automatycznie, bez zmuszania nas do interakcji z IDE.



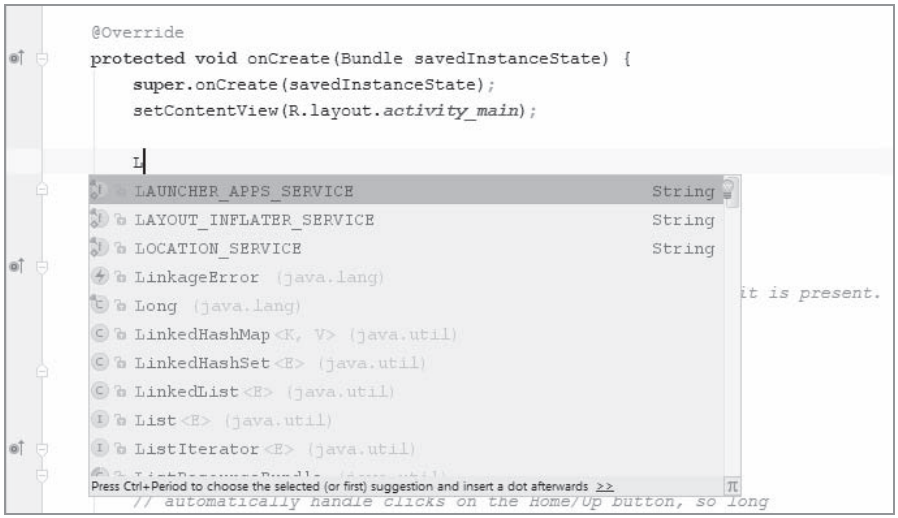
Rysunek 4.3. Okienko sugerujące dodanie odpowiedniej instrukcji import

Uzupełnianie kodu

Mechanizm automatycznego uzupełniania kodu pomaga zwiększyć szybkość pisania kodu. Jego działanie polega na wyświetlaniu listy sugestii, której zawartość zależy od tego, co w danej chwili piszemy.

Przykład takiej podstawowej listy sugestii wyświetlanej przez mechanizm automatycznego uzupełniania podczas wpisywania kodu przedstawia rysunek 4.4. Jeśli lista nie została wyświetlona, to można to zrobić, używając kombinacji klawiszy *Ctrl*+spacja.

[1] Na bieżąco dodawaj jednoznaczne instrukcje `import` — *przyp. tłum.*



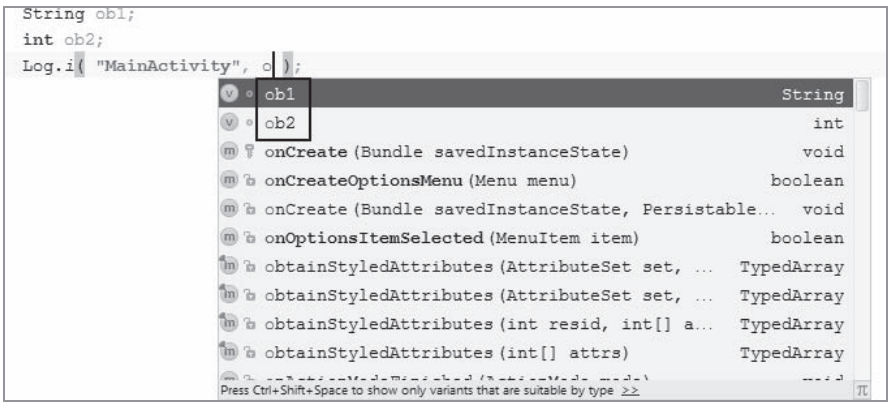
Rysunek 4.4. Lista sugestii mechanizmu automatycznego uzupełniania kodu

Teraz można kontynuować wpisywanie, a następnie wybrać jedną z opcji z listy i nacisnąć klawisz *Enter* lub dwukrotnie kliknąć tę opcję, by dodać ją do kodu. Jeśli wpisywanym fragmentem kodu jest wyrażenie, a my chcemy wstawić je w zanegowanej formie, wystarczy je wybrać z listy sugestii, a następnie zamiast naciskać klawisz *Enter* lub dwukrotnie klikać, należy nacisnąć klawisz ze znakiem wykrzyknika (*!*). Spowoduje to dodanie zanegowanego wyrażenia.

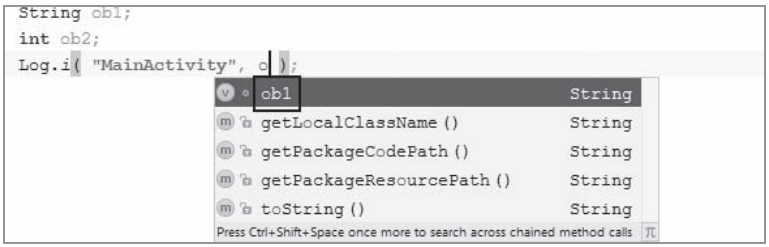
Kolejnym typem uzupełniania kodu jest **inteligentne wpisywanie** (ang. *smart type*). Jeśli wpisujemy wywołanie metody, która ma parametr typu `String`, to sugerując dostępne argumenty, edytor będzie prezentował wyłącznie obiekty typu `String`. Tego typu inteligentne wpisywanie jest wykonywane w kilku przypadkach: z prawej strony operatora przypisania, w wywołaniach metod, w instrukcji `return` oraz w inicjalizatorach zmiennych. By wyświetlić listę inteligentnych sugestii, należy nacisnąć klawisze *Ctrl+Shift* oraz klawisz spacji.

W prosty sposób można samemu przekonać się, czym się różnią te oba typy list sugestii. W tym celu należy zdefiniować w kodzie dwie zmienne, typu `String` oraz `int`, a następnie wywołać metodę wymagającą przekazania argumentu typu `String`, na przykład metodę `i` klasy `Log`. Podczas wpisywania argumentu typu `String` należy zwrócić uwagę na różnicę pomiędzy zwyczajną listą sugestii (*Ctrl+*spacja) przedstawioną na rysunku 4.5 a listą inteligentnych sugestii (*Ctrl+Shift+*spacja) przedstawioną na rysunku 4.6.

W przypadku pierwszej listy, przedstawionej na rysunku 4.5, sugerowane są obie zmienne, choć typ drugiej z nich nie odpowiada typowi parametru. W przypadku drugiej listy zasugerowana została wyłącznie zmienna typu `String`.



Rysunek 4.5. Zwyczajna lista sugestii automatycznego uzupełniania kodu



Rysunek 4.6. Lista inteligentnych sugestii

Ostatnim zastosowaniem mechanizmu automatycznego uzupełniania kodu jest **uzupełnianie instrukcji**. Wystarczy wpisać instrukcję, nacisnąć klawisze *Ctrl+Shift+Enter*, a wszystkie pozostałe elementy instrukcji zostaną automatycznie dodane do kodu. Jeśli na przykład użyjemy tej kombinacji klawiszy po wpisaniu słowa kluczowego `if`, to edytor automatycznie doda do kodu parę nawiasów oraz parę nawiasów klamrowych. Tej samej kombinacji klawiszy można używać do uzupełniania deklaracji metod. Wystarczy zacząć wpisywać metodę, a następnie, po wpisaniu nawiasu otwierającego, nacisnąć klawisze *Ctrl+Shift+Enter*. W efekcie edytor doda nawias zamykający oraz parę nawiasów klamrowych.

Generowanie kodu

W celu wygenerowania bloków kodu podczas tworzenia klas można skorzystać z opcji *Code/Generate*, dostępnej w menu głównym, bądź też z kombinacji klawiszy *Alt+Insert*. W ten sposób można generować konstruktory, akcesory `get` i `set`, metody `equals` oraz `toString`, jak również przesłaniać i delegować metody.

Inną formą generowania kodu jest umieszczanie wybranego bloku w jakiejś instrukcji (`if`, `if/else`, `while`, `for`, `try/catch` itd.). Można to zrobić, zaznaczając wiersz kodu i wybierając z menu głównego opcję *Code/Surround With* lub naciskając klawisze *Ctrl+Alt+T*.

Trzecią formą generowania kodu jest wstawianie gotowych szablonów. Aby to zrobić, należy wybrać z menu głównego opcję *Code/Insert Live Templates* lub nacisnąć klawisze *Ctrl+J*. Wykonanie każdej z tych czynności powoduje wyświetlenie listy dostępnych szablonów. Pozwalają one na wstawianie gotowych fragmentów kodu przeglądającego w pętli kolekcje, tablice, listy itd., kodu do wyświetlania odpowiednio sformatowanych łańcuchów znaków, a także kodu do zgłaszania wyjątków lub deklaracji zmiennych statycznych i sfinalizowanych. Z lewej strony listy wyświetlany jest prefiks danego szablonu — wystarczy wpisać ten prefiks w edytorze i nacisnąć klawisz *Tab*, aby szablon został automatycznie wstawiony do kodu.

W ramach testu można wpisać w metodzie `onCreate` aktywności głównej naszego przykładowego projektu łańcuch `inn` i nacisnąć klawisz *Tab*, a następnie *Enter*. W rezultacie w kodzie pojawi się blok instrukcji warunkowej, a kursor zostanie umieszczony wewnątrz pary nawiasów klamrowych. Jeśli teraz wpiszemy `soutv` i ponownie naciśniemy klawisze *Tab* i *Enter*, to uzyskamy wynik przedstawiony w poniższym przykładzie:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    if (savedInstanceState != null) {
        System.out.println("savedInstanceState = " + savedInstanceState);
    }

}
```

Poruszanie się po kodzie

Najprostszym sposobem przejścia do deklaracji lub deklaracji typów jest naciśnięcie klawisza *Ctrl* i kliknięcie nazwy metody, kiedy zostanie ona wyświetlona w sposób charakterystyczny dla odnośników. Ten sam efekt można uzyskać, wybierając z menu głównego opcję *Navigate/Declaration*.

Korzystając z ikon wyświetlanych w kolumnie z lewej strony okna edytora, można poruszać się po hierarchii metod. Obok deklaracji metody, która należy do hierarchii, jest wyświetlana ikona informująca, czy dana metoda stanowi implementacje metody interfejsu, klasy abstrakcyjnej, czy też przesłania metodę klasy bazowej albo czy jest zaimplementowana lub przesłonięta przez metody klas pochodnych. Aby przejść do następnej metody w hierarchii, wystarczy kliknąć tę ikonę. To samo można zrobić, wybierając z menu głównego opcję *Navigate/Super Method* lub *Navigate/Implementations(s)*. Można to w każdej chwili sprawdzić w aktywności głównej naszego przykładowego projektu (pliku *MainActivity.java*), co pokazano na poniższym rysunku 4.7.

Rysunek 4.7. Ikona pozwalająca przejść do deklaracji następnej metody w hierarchii

Kolejną przydatną możliwością związaną z poruszaniem się po kodzie źródłowym są tak zwane **regiony niestandardowe** (ang. *custom regions*). Taki region to fragment kodu, który można potraktować jako pewną całość i nadać mu nazwę. Jeśli na przykład tworzona klasa składa się z wielu metod, to można utworzyć kilka takich regionów, a następnie dodawać do nich metody. Takie regiony mają nazwy lub opisy i można je zwijać lub rozwijać przy wykorzystaniu mechanizmu zwijania kodu.

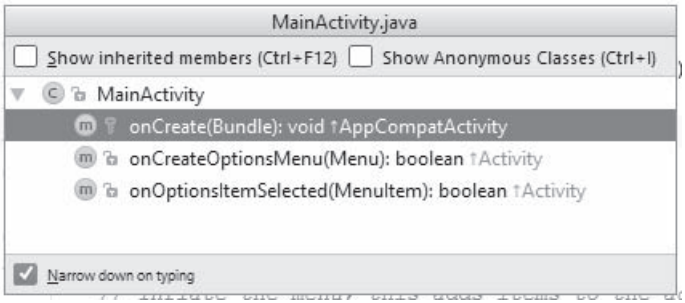
Regiony niestandardowe można tworzyć, korzystając z mechanizmu generowania kodu — wystarczy zaznaczyć fragmentu kodu, wybrać z menu głównego opcję *Code/Surround With*, a następnie jedną z dwóch poniższych możliwości:

- *<editor-fold...> Comments*,
- *region...endregion Comments*.

Wybór każdej z tych opcji spowoduje utworzenie regionu, lecz utworzone regiony będą prezentowane w różny sposób.

W przypadku stosowania regionów niestandardowych, możemy wygodnie poruszać się pomiędzy nimi, korzystając z opcji *Custom Region* dostępnej w menu *Navigate*. W tym menu dostępne są także pozostałe opcje związane z poruszaniem się po kodzie. Poniższa lista przedstawia najważniejsze z nich:

- *Class/File/Symbol* — ta opcja pozwala odnajdować klasy, pliki lub symbole na podstawie nazwy.
- *Line* — ta opcja pozwala przejść do wiersza o podanym numerze.
- *Last Edit Location* — wybranie tej opcji spowoduje przejście do miejsca, w którym zostały wprowadzone ostatnie modyfikacje.
- *Test* — wybór tej opcji powoduje przejście do klasy służącej do testowania aktualnie edytowanej klasy.
- *File Structure* — ta opcja powoduje wyświetlenie okienka dialogowego prezentującego strukturę pliku. Warto wyświetlić to okno dialogowe dla aktywności głównej naszego przykładowego projektu, by przekonać się, w jaki sposób jest prezentowana jego struktura. Będzie to lista nazw metod wraz z ikonami określającymi ich typ lub poziom dostępu (przykładową postać takiego okna przedstawia rysunek 4.8).



Rysunek 4.8. Przykładowe okno struktury pliku

- *File Path* — wybranie tej opcji powoduje wyświetlenie okna dialogowego przedstawiającego pełną ścieżkę dostępu do pliku aktualnie otworzonego w edytorze.
- *Type Hierarchy* — ta opcja powoduje wyświetlenie okna dialogowego pokazującego hierarchię typów dla aktualnie wybranego obiektu.
- *Method Hierarchy* — wybranie tej opcji powoduje wyświetlenie okna dialogowego przedstawiającego hierarchię metod dla aktualnie wybranej metody.
- *Call Hierarchy* — wybranie tej opcji powoduje wyświetlenie okna dialogowego przedstawiającego hierarchię wywołań dla aktualnie wybranej metody.
- *Next Highlighted Error* — ta opcja pozwala przejść do następnego błędu.
- *Previous Highlighted Error* — ta opcja pozwala przejść do poprzedniego błędu.
- *Next Method* — ta opcja pozwala przejść do następnej metody.
- *Previous Method* — ta opcja pozwala przejść do poprzedniej metody.

Przydatne skróty klawiszowe

Lista wszystkich skrótów klawiszowych stosowanych w Android Studio jest dostępna w oknie dialogowym *Settings*, w opcji *Keymap*. Można tam także zmieniać zdefiniowane kombinacje klawiszy. Tabela 4.1 przedstawia wybrane skróty klawiszowe stosowane w Android Studio w systemie Windows.

Podsumowanie

Rozdział ten przybliżył nam kilka użytecznych sztuczek, dzięki którym będziemy w stanie w pełni wykorzystać możliwości edytora kodu Android Studio. Wiemy już, jak korzystać z automatycznego uzupełniania kodu, jak go generować, oraz znamy użyteczne skróty klawiszowe, pozwalające na błyskawiczne wykonywanie przeróżnych operacji. Potrafimy już także modyfikować sposób działania edytora i jesteśmy gotowi, by rozpocząć kodowanie.

Tabela 4.1. Wybrane skróty klawiszowe stosowane w Android Studio

Kombinacja klawiszy	Opis
Ctrl+W	Zaznacza wyrażenie. Kolejne naciśnięcia tej kombinacji klawiszy będą rozszerzać zaznaczony obszar kodu. Odwrotny efekt daje naciskanie klawiszy *Ctrl+Shift+W*.
Ctrl+/	Powoduje umieszczenie na początku wiersza symbolu komentarza. Aby dodać komentarz blokowy, należy skorzystać z kombinacji klawiszy *Ctrl+Shift+/*.
Ctrl+Alt+I	Powoduje wcięcie zaznaczonego fragmentu kodu. Ta kombinacja jest bardzo użyteczna do porządkowania gotowego bloku kodu lub metody.
Ctrl+Alt+O	Powoduje optymalizację instrukcji `import` — usunięcie niepotrzebnych i uporządkowanie pozostałych.
Ctrl+Shift+strzałka	Powoduje przeniesienie zaznaczonego bloku kodu o jeden wiersz niżej lub wyżej.
Alt+strzałka	Pozwala na przełączanie się pomiędzy otwartymi kartami edytora.
Ctrl+F	Pozwala wyszukiwać tekst w aktualnie wyświetlonej karcie edytora.
Ctrl+R	Pozwala wyszukiwać i zastępować tekst w aktualnie wyświetlonej karcie edytora.
Ctrl+A	Zaznacza cały kod w aktualnie edytowanym pliku.
Ctrl+D	Kopiuje zaznaczony fragment kodu i wkleja go poniżej. Jeśli w chwili naciśnięcia tych klawiszy nie jest zaznaczony żaden fragment kodu, to powielony zostanie bieżący wiersz.
Ctrl+Y	Usuwa cały wiersz kodu bez pozostawiania pustego wiersza.
Ctrl+Shift+U	Zmienia wielkość liter.
Tab	Przechodzi do następnego parametru.

W następnym rozdziale zajmiemy się tworzeniem pierwszych układów interfejsu użytkownika aplikacji. Zostanie w nim opisany sposób tworzenia układów przy wykorzystaniu kreatora graficznego, jak również w drodze ręcznej edycji plików XML. Stworzymy także naszą pierwszą aplikację — klasyczny program typu *Witaj, świecie!*, w którym wykorzystamy kontrolkę `TextView`. Będą przedstawione informacje o tym, jak przygotować aplikację do wyświetlania jej na ekranach o różnych wielkościach oraz jak przygotować ją do zmiany orientacji ekranu. Na samym końcu zamieszczone zostały także informacje o tematach interfejsu użytkownika oraz o sposobach obsługi zdarzeń.

Skorowidz

D

E

F

G

H

I

J

K

L

Ł

M

N

O

P

R

S

T

U

V

W

X

Z

Ż

Android Studio
Podstawy

Android to obecnie najpopularniejszy system operacyjny, używany przede wszystkim w urządzeniach mobilnych. Jeżeli chcesz wydajnie tworzyć aplikacje dla tej platformy, potrzebujesz wyjątkowego narzędzia, które pozwoli Ci zrealizować Twoje pomysły. Firma JetBrains stworzyła środowisko, którego szukasz. Android Studio jest oparte na słynnym IDE IntelliJ IDEA, które zmieni Twój sposób pracy z systemem Android.

Jeżeli chcesz w pełni wykorzystać potencjał tego narzędzia, potrzebujesz wyjątkowej książki, która zaprezentuje Ci najlepsze techniki pracy. Właśnie masz ją przed sobą! Przekonaj się, jak zainstalować Android Studio i dostosować je do własnych potrzeb oraz jak efektywnie korzystać z możliwości edytora kodu. W kolejnych rozdziałach poznasz zaawansowane narzędzia wspomagające projektowanie interfejsu użytkownika, debugowanie aplikacji oraz tworzenie paczek APK gotowych do dystrybucji. Książka ta jest doskonałą lekturą dla programistów chcących wykorzystać potencjał najlepszego IDE na świecie!

Zwiększ swoją efektywność dzięki Android Studio!

Dzięki tej książce:

- zainstalujesz środowisko Android Studio i dostosujesz je do swoich potrzeb
- poznasz efektywne techniki pracy z edytorem kodu
- zaprojektujesz i wykonasz atrakcyjny, funkcjonalny interfejs użytkownika
- przygotujesz Twoją aplikację do dystrybucji

Belén Cruz Zapata — zafascynowana mobilnym światem doktorantka Uniwersytetu w Murcji (Hiszpania). Zawodowo zajmuje się badaniem technologii mobilnych. Pracuje dla firmy Groupon.

[PACKT] open source PUBLISHING community experience distilled

Helion

37120 numer katalogowy

księgarnia internetowa
http://helion.pl

zamówienia telefoniczne
0 801 339900
0 601 339900

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-283-1466-5

cena: 32,90 zł