Wprowadzenie

- Dla kogo przeznaczona jest ta książka? (22)
- Wiemy, co sobie myślisz (23)
- Metapoznanie - myślenie o myśleniu (25)
- Oto, co możesz zrobić, by skłonić swój mózg do posłuszeństwa (27)
- Ważne informacje (28)
- Zespół korektorów merytorycznych (30)
- Podziękowania (31)

1. Początki

- Aplikacja musi robić wiele rzeczy (35)
- Co jest potrzebne aplikacji? (36)
- Rails służy do tworzenia aplikacji bazodanowych, takich jak system sprzedaży biletów (38)
- Nową aplikację tworzy się za pomocą polecenia rails (39)
- Teraz do domyślnej aplikacji trzeba dodać własny kod (41)
- Rusztowanie to kod GENEROWANY (42)
- W bazie danych nie ma jeszcze tabel! (46)
- Tabelę tworzy się dzięki wykonaniu migracji (47)
- Pięknie! Uratowałeś pracę kumpla! (51)
- By zmodyfikować aplikację, musisz przyjrzeć się jej architekturze (52)
- Trzy części Twojej aplikacji: model, widok i kontroler (53)
- Cała prawda o Rails (54)
- Trzy typy kodu przechowywane są w OSOBNYCH folderach (57)
- Trzeba zmodyfikować pliki WIDOKU (58)
- Edycja kodu HTML w widoku (59)
- Aplikacja musi teraz przechować większą liczbę informacji (63)
- Migracja to po prostu skrypt w języku Ruby (64)
- Rails może generować migracje (65)
- Nadaj swojej migracji odpowiednią nazwę, a Rails napisze za Ciebie kod (66)
- Migrację należy wykonać za pomocą rake (67)
- Sama zmiana bazy danych nie wystarczy (68)
- Dlaczego Rails mówi do mnie po angielsku? (75)
- Uczymy Rails języków obcych (76)

2. Poza rusztowaniem

- Rusztowanie robi O WIELE za dużo (85)
- Zaczynamy od wygenerowania modelu MeBay... (86)
- ...a następnie utworzymy tabelę za pomocą polecenia rake (87)
- Ale co z kontrolerem? (88)
- Widok tworzony jest przez szablon strony (90)
- Szablon strony zawiera kod HTML (91)
- Trasa mówi Rails, gdzie znajduje się strona (93)
- Widok nie ma danych do wyświetlenia (100)
- Co zatem powinna pokazywać strona? (101)
- Kontroler przesyła ogłoszenie do widoku (102)
- Rails zmienia rekord w obiekt (104)

- Dane znajdują się w pamięci, a strona internetowa je widzi (105)
- Jest problem - ludzie nie potrafią znaleźć żądanych stron (109)
- Trasy wykonywane są w kolejności (112)
- By przesłać dane do widoku, będziesz potrzebował kodu kontrolera (114)
- Strona indeksująca potrzebuje danych ze WSZYSTKICH rekordów (115)
- Metoda Ad.find(:all) wczytuje całą tabelę naraz (116)
- Dane zwracane są jako obiekt zwany tablicą (117)
- Tablica to ponumerowana sekwencja obiektów (118)
- Wczytanie wszystkich ogłoszeń za pomocą pętli for (122)
- Potrzebny nam kod HTML dla każdego elementu tablicy (123)
- Rails konwertuje szablony stron na kod języka Ruby (124)
- Pętle można dodawać do szablonów stron za pomocą scriptletów (125)
- Z każdym przejściem pętli strona generuje jeden odnośnik (126)
- Jak wygląda wygenerowany kod HTML? (127)
- Ale my mamy dwa szablony stron... czy powinniśmy zmieniać kod każdego z nich? (130)
- A co z nową treścią statyczną wysłaną przez MeBay? (133)

3. Wstawianie, uaktualnianie i usuwanie

- Ludzie chcą sami publikować ogłoszenia w Internecie (140)
- Wiesz już, jak budować aplikację publikującą dane z bazy (141)
- Zapisywanie danych działa dokładnie ODWROTNIE do ich odczytywania (142)
- Potrzebny nam formularz służący do dodawania danych oraz metoda akcji zapisująca te dane (143)
- Czy formularze i obiekty są ze sobą powiązane? (145)
- Rails może tworzyć formularze powiązane z obiektami modelu (146)
- Obiekt formularza @ad nie został utworzony (150)
- Obiekt formularza musi zostać utworzony przed wyświetleniem formularza (151)
- Obiekt ogłoszenia formularza zostanie utworzony w akcji new kontrolera (152)
- Każdy szablon strony ma teraz odpowiadającą mu metodę kontrolera (153)
- Formularz nie odsyła obiektu, odsyła DANE (155)
- Rails musi przekształcić dane na obiekt przed ich zapisaniem (156)
- Metoda create kontrolera krok po kroku (157)
- Kontroler musi zapisać rekord (158)
- Nie twórz nowej strony, użyj istniejącej (164)
- Jak jednak akcja kontrolera może wyświetlać stronę INNEJ akcji? (165)
- Przekierowania pozwalają kontrolerowi określić, który widok zostanie wyświetlony (166)
- Ale co się dzieje, kiedy ogłoszenie należy po opublikowaniu poprawić? (169)
- Uaktualnienie ogłoszenia przypomina utworzenie go... tylko jest trochę inne (170)
- Zamiast tworzyć ogłoszenie, musimy je odnaleźć; zamiast je zapisać, musimy je uaktualnić (171)
- Ograniczanie dostępu do funkcji (178)
- ...teraz jednak stare ogłoszenia trzeba usunąć (181)
- Wykonanie tego samodzielnie dało Ci możliwość zrobienia więcej, niż potrafi rusztowanie (187)

4. Wyszukiwanie w bazie danych

- Dbaj o siebie z Rubyville Health Club (190)
- Aplikacja w zasadzie wygląda dość podobnie... (193)
- Poprawimy rusztowanie (194)
- Zaprojektowanie opcji wyszukiwania (195)
- Zacznijmy od utworzenia formularza (196)
- Dodanie wyszukiwania do interfejsu (199)
- Jak możemy znaleźć rekordy klientów? (207)
- Potrzebne nam jedynie te rekordy, gdzie client_name = łańcuch wyszukiwania (208)
- Dla każdego atrybutu istnieje metoda wyszukująca (209)
- Musimy dopasować albo nazwisko klienta, albo trenera (214)
- Metody wyszukujące piszą zapytania do bazy danych (215)
- Musimy być w stanie zmodyfikować warunki wykorzystane w zapytaniu SQL (216)

5. Sprawdzanie poprawności danych

- Uwaga - pojawiły się niepoprawne dane (224)
- Kod sprawdzający poprawność danych przynależy do MODELU (226)
- Na potrzeby prostego sprawdzania poprawności danych Rails wykorzystuje walidatory (227)
- Jak działają walidatory? (228)
- Sprawdźmy, czy coś jest liczbą (230)
- Użytkownicy pomijają niektóre pola formularzy (232)
- Jak sprawdzamy obowiązkowe pola? (233)
- Walidatory są proste i działają dobrze (236)
- W MeBay wydarzyło się coś dziwnego (239)
- Walidatory sprawdzają, jednak nie wyświetlają błędów (240)
- Jeśli tworzysz własne strony, musisz także pisać własny kod komunikatów o błędach (243)
- Kontroler musi wiedzieć, czy wystąpił błąd (244)
- Nadal musimy wyświetlić komunikaty o błędach! (248)
- System MeBay wygląda przepięknie (250)

6. Tworzenie połączeń

- Linie Coconut Airways potrzebują nowego systemu rezerwacji (256)
- Chcemy widzieć loty i rezerwacje miejsc razem (258)
- Zobaczmy, co daje nam rusztowanie dla miejsc (259)
- Na stronie lotu musi się znaleźć formularz rezerwacji oraz lista miejsc (260)
- Jak możemy podzielić zawartość strony na odrębne pliki? (261)
- ERb SKŁADA nasze strony (265)
- Jak można utworzyć szablon częściowy formularza rezerwacji? (266)
- Teraz musimy dołączyć szablon częściowy do szablonu strony (267)
- Musimy przekazać szablonowi częściowemu miejsce! (270)
- Zmienne lokalne można przekazywać do szablonu częściowego (271)
- Niezbędny jest nam szablon częściowy dla listy miejsc (278)
- Ludzie trafiają na niewłaściwe loty (280)
- Powiązanie łączy ze sobą modele (281)
- Jak jednak definiujemy powiązanie? (283)
- Niektóre osoby mają jednak za duży bagaż (285)
- Musimy napisać WŁASNY walidator (286)

- Potrzebne nam jest ODWROTNE powiązanie (289)
- System wystartował w Coconut Airways (296)

7. Ajax

- Linie Coconut Airways mają nową ofertę (300)
- Które części strony najbardziej się zmieniają? (301)
- Czy przeglądarka nie uaktualnia zawsze całej strony? (306)
- Co INNEGO może wykonać żądanie? (307)
- Najpierw musimy dołączyć biblioteki Ajaksa... (308)
- ...a następnie dodać odnośnik "Odśwież" oparty na Ajaksie (309)
- Przeglądarka musi prosić o uaktualnienie (314)
- Czy jednak POWINNIŚMY nakazywać przeglądarce nieustanne proszenie? (315)
- Licznik obsługuje się podobnie jak przycisk czy odnośnik (316)
- Cała prawda o Ajaksie (320)
- Ktoś ma kłopot ze swoim wieczorem kawalerskim (321)
- Formularz musi wykonać żądanie oparte na Ajaksie (322)
- Formularz musi pozostawać pod KONTROLĄ JavaScriptu (323)
- Musimy zastąpić metodę create (325)
- Jaki efekt ma ten kod? (326)
- Teraz pojawił się problem z rezerwacjami lotów (331)
- Potrafimy uaktualnić jedną część strony naraz (332)
- Kontroler musi zamiast HTML zwracać kod w JavaScripcie (333)
- Co generuje Rails? (337)
- Jeśli nie powiesz, gdzie umieścić odpowiedź, zostanie ona wykonana (338)

8. XML i różne reprezentacje

- Zdobywanie szczytów świata (344)
- Użytkownicy nienawidzą interfejsu aplikacji! (345)
- Dane muszą się znaleźć na mapie (346)
- Musimy utworzyć nową akcję (347)
- Nowa akcja wydaje się działać... (348)
- Nowa strona potrzebuje mapy... w tym właśnie rzecz! (349)
- Jakiego typu kod jest nam potrzebny? (350)
- Kod ten działa jedynie dla serwera lokalnego (351)
- Teraz potrzebne nam dane mapy (352)
- Co zatem powinniśmy wygenerować? (354)
- Wygenerujemy kod XML z modelu (355)
- Obiekt modelu może generować kod XML (356)
- Jak powinien wyglądać taki kod kontrolera? (357)
- Tymczasem na wysokości kilku tysięcy metrów... (362)
- Musimy generować XML oraz HTML (363)
- XML i HTML to po prostu reprezentacje (365)
- W jaki sposób powinniśmy decydować, z którego formatu skorzystać? (366)
- Jak działa strona z mapą? (370)
- Kod jest gotowy do opublikowania (372)
- Kanały RSS to po prostu kod XML (380)
- Utworzymy akcję o nazwie news (381)
- Musimy zmienić strukturę kodu XML (384)

- Użyjemy nowego typu szablonu - XML Builder (385)
- Teraz dodajmy kanały RSS do stron (389)
- Zdobyłeś szczyt! (391)

9. Architektura REST i Ajax

- Zdarzeń jest zbyt dużo! (394)
- Mapa mogłaby pokazywać więcej szczegółów (395)
- Możemy rozszerzyć funkcjonalność mapy za pomocą Ajaksa (396)
- Jak jednak możemy przekształcić stronę indeksującą? (397)
- Co będzie musiała wygenerować akcja show? (398)
- Nowa funkcjonalność mapy jest pełnym sukcesem! (403)
- Musimy utworzyć żądania wykorzystujące Ajaksa (404)
- Szablon częściowy mapy pozwala nam wybrać akcję new (406)
- Jak możemy UDOWODNIĆ, że zdarzenie zostało zapisane? (411)
- Formularz musi uaktualnić zawartość elementu wyskakującego okna (412)
- Lawina! (417)
- Jak działa to teraz... (418)
- Możemy umieścić odnośnik "Edit" w oknie wyskakującym (419)
- Zaczniemy od zmodyfikowania akcji edit (420)
- Na stronie show potrzebny nam jest także nowy odnośnik (422)
- Jak stosuje się metodę pomocniczą link_to? (423)
- Na pomoc spieszy odnośnik oparty na Ajaksie (427)
- Używamy niewłaściwej trasy! (429)
- Na wybór trasy ma wpływ metoda HTTP (430)
- Czym jest zatem metoda HTTP? (431)
- Witryna Head First Climbers Cię potrzebuje! (434)

10. Prawdziwe aplikacje

- Patrz! Eksperymenty z językiem Ruby! (441)
- Aplikacje internetowe muszą być testowane (442)
- Jakie rodzaje testów są dostępne? (443)
- Udostępnienie aplikacji użytkownikom (444)
- Jak zmienia się bazę danych? (445)
- Czym jest architektura REST? (446)
- Aplikacje internetowe pobłądziły (447)
- Życie na krawędzi (448)
- Uzyskanie dodatkowych informacji (449)
- Nieco dodatkowej lektury... (450)
- Książki Head First o podobnej tematyce (451)
- Koniec wycieczki... (453)

Skorowidz (455)