Jakarta Struts

Tomasz Mularczyk
Kwiecień 2005

Jakarta Struts - wprowadzenie

- Servlety Javy głównie dzięki swojej szybkości od początku zdobyły wielką popularność
- Servlety były dodatkowo bardziej przenośne oraz skalowalne niż zwykłe skrypty sieciowe
- Ich stosowanie posiadało w zasadzie tylko jedną wadę... Wygenerowanie przy ich pomocy sensownej wielkości strony wymagało użycia setek linii kodu opierającego się na wypisywaniu zawartości tworzonej strony (zwykle przy użyciu instrukcji println)

Jakarta Struts - wprowadzenie

- Systemem tworzenia stron www pozbawionym tej wady szybko stał się JSP (Java Server Pages)
- Dzięki zastosowaniu znaczników kod tworzący stronę nie musiał być wielokrotnie powtarzany
- Zmiana wyglądu strony stała się łatwiejsza (wystarczyło zmienić kilka znaczników)
- Zalety tej technologii spowodowały, że zdobyła wielką popularność
- Niestety, JSP nie jest pozbawione wad...

Jakarta Struts - wprowadzenie

- Czyste Java Server Pages nie pozwalają między innymi na kontrolę przepływu, co często jest wymagane przy poważnych zastosowaniach
- Powstaje pomysł, aby połączyć Servlety oraz JSP
- Może uda się połączyć kontrolę, którą posiada się tworząc strony www przy pomocy Servletów oraz łatwość pisania wynikowego kodu html przy pomocy JSP

Jakarta Struts - wprowadzenie

- Wykorzystanie Java Server Pages wraz z Servletami Javy nazwano Modelem 2
- Domyślnie nazwano tym samym wykorzystanie samych JSP Modelem 1
- Połączenie JSP z Servletami realizuje się poprzez realizację wzorca MVC (Model View Controller)

Architektura MVC – założenia

Program kiedyś:

Wejście -> Przetwarzanie -> Wyjście

Input->Processing ->Output

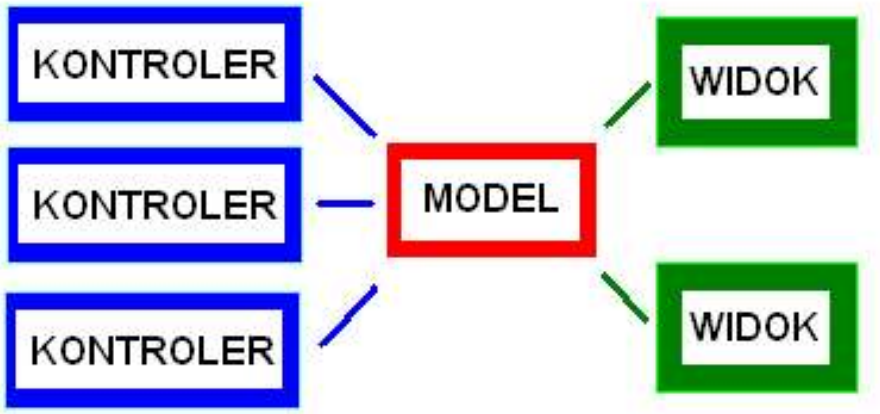
Program dziś:

Kontroler -> Model -> Widok

Controller->Model ->View

Architektura MVC - opis

- Podejście to pojawiło się w SmallTalk 80
- Kontrolery, model i widoki traktowane jako oddzielne byty
- Zmiany modelu natychmiast odzwierciedlane w widokach



Architektura MVC - opis

- Wszystkie komponenty w systemie muszą dać się określić jako kontroler, model, widok bądź część któregoś z nich
- Połączenie model <-> widok jest realizowane dynamicznie – w czasie działania programu, a nie w czasie kompilacji
- Jeżeli zaprojektujemy w MVC każdy komponent, to łatwe będzie ich łączenie w spójny projekt

Architektura MVC - opis

- Usuwanie bądź przebudowa komponentu nie pociąga za sobą potrzeby ingerencji w cały system
- Model nie wie, jakie są jego kontrolery, nie zna także korzystających z niego widoków – sam system zajmuje się ich spójnym połączeniem
- Kontrolery i Widoki znają skojarzony z nimi model

Architektura MVC - opis

- Kontrolery i Widoki mają tylko po jednym przypisanym im modelu
- Model oczywiście może posiadać wiele kontrolerów i widoków
- Zmiany w modelu są zwykle zapoczątkowywane przez jeden z kontrolerów tego modelu
- Zmiany te są natychmiast odwzorowywane we wszystkich skojarzonych z modelem widokach

Architektura MVC - przykład

Załóżmy, że projektujemy wieloosobową, trójwymiarową grę, w której gracze dowodzą okrętami:

- Model – cały świat 3d wraz opisami statków graczy, lecących pocisków, własności morza
- Kontrolery – akcje podejmowane przez różnych graczy, np. rozkaz wystrzelenia pocisku
- Widoki – np. widok z mostku, rzut pionowy (mapa akwenu), czy patrzenie przez lornetkę

Architektura MVC w Javie

Java oferuje wsparcie dla MVC poprzez dwie klasy:

- Obserwator (Observer) – obiekt, który chce być powiadamiany o zmianach w innym obiekcie
- Obserwowany (Observable) – każdy obiekt, którego zmiana stanu może zainteresować inny obiekt

Architektura MVC w Javie

- MODEL jest klasy „Obserwowany”
- WIDOK jest klasy „Obserwator”
- WIDOK i KONTROLER posiadają referencję do MODELU i potrafią się z nim komunikować poprzez wywoływanie jego metod

MVC a Jakarta Struts

- W aplikacji stworzonej w architekturze MVC przepływowi kontroli pośredniczy centralny kontroler
- W przypadku tworzenia stron www kontroler ten przydziela zewnętrzne żądania do konkretnych obiektów zajmujących się nimi
- Każdy z tych obiektów to adapter pomiędzy modelem, a przydzielonym mu żądaniem

MVC a Jakarta Struts

- Model reprezentuje całą logikę biznesową bądź stan aplikacji
- W tym momencie kontrola zwykle jest kierowana do określonego widoku
- Wybór widoku, do którego kontrola zostanie skierowana, polega na skorzystaniu z mapowania zadanego w pliku konfiguracyjnym bądź bazie danych aplikacji

Model w Jakarta Struts

- Model możemy logicznie podzielić na:
- sekcję opisującą aktualny, wewnętrzny stan systemu
- sekcję zawierającą możliwe do wykonania akcje, które mogą zmienić stan systemu

- Naturalne podejście gramatyczne – akcje to czasowniki, zaś stany to rzeczowniki

Model w Jakarta Struts

- Stany systemu zwykle realizuje się poprzez Java Beans
- W zależności od wielkości systemu, Java Beans mogą potrafić same przechowywać (i dbać o ich zachowanie) wiadomości o swoim stanie bądź są tylko fasadami do rozbudowanych komponentów, z których ten stan pobierają

Model w Jakarta Struts

- Komponenty, które zwykle są używane jako źródła danych dla Java Beans:
- Bazy danych
- Wyszukiwarki
- Inne Java Beans

- Nic nie stoi na przeszkodzie, aby wykorzystać dowolny inny komponent

Model w Jakarta Struts

- W rozwiązaniach dużej wielkości możliwe do wykonania na modelu operacje są zwykle reprezentowane poprzez zestaw metod należących do Java Beans
- W ten sposób można na przykład powiązać jeden Java Bean z każdym użytkownikiem, gdy zależy nam na przykład na zapamiętaniu, które towary zostały przez niego wybrane w sklepie internetowym w czasie trwania jego sesji

Model w Jakarta Struts

- Taki Java Bean miałby na pewno metody dodajZakup, podajListęZakupów, przejdźDoKasy

- Rozwiązaniem alternatywnym jest stworzenie zestawu Java Beans, w którym za poszczególne operacje odpowiadałyby różne obiekty – np. jako Session Enterprise Java Beans

Model w Jakarta Struts

- W niewielkich systemach zamiast budowania oddzielnych Java Beans, akcje, które mają być wykonane można zdefiniować bezpośrednio z klas Akcji będących częścią składową powłoki Kontroli Jakarta Struts

- Rozwiązanie to jest przydatne, gdy logika biznesowa jest bardzo prosta

Widok w Jakarta Struts

- Widok w Jakarta Struts jest zwykle częścią opartą na Java Server Pages
- JSP jest na tyle elastyczną technologią, że pozwala zarówno na wykorzystanie statycznego tekstu bądź wyglądu strony wraz z dodatkiem dynamicznie tworzonej zawartości – w tym wykorzystanie Java Beans

Widok w Jakarta Struts

- Jakarta Struts posiadają wiele zestawów własnych znaczników JSP
- Znaczniki JSP dostępne w Jakarta Struts pozwalają na tworzenie bardzo uniwersalnych i wielojęzycznych stron www

Kontroler w Jakarta Struts

- Kontroler odpowiada w Struts za pobieranie żądań od klientów (zwykle użytkowników przeglądarek), ustalanie, która część logiki biznesowej powinna je spełnić oraz przekazywanie ich dalej do wyznaczonych widoków, które są odpowiedzialne za stworzenie interfejsu dla użytkownika

Kontroler w Jakarta Struts

- Pierwotnym komponentem kontrolera jest ActionServlet, którego konfiguracja polega na podaniu mapowania akcji

- Wszystkie mapowania akcji są podawane jako ścieżki, które następnie porównuje się z URI żadań w celu określenia nazwy akcji, którą należy podjąć

Kontroler w Jakarta Struts

- Wszystkie akcje są podklasami org.apache.struts.action.Action
- Akcje zawierają odwołania do logiki biznesowej, interpretują wyniki i ostatecznie przekazują kontrolę odpowiednim składnikom widoku

Kontroler w Jakarta Struts

- Jakarta Struts pozwalają także na wykorzystanie mapowania akcji, które zachowują się inaczej niż przytoczone powyżej

- Pozwala to na korzystanie z Jakarta Struts nawet, jeśli nasza aplikacja nie do końca działa według schematu przyjętego w całej reszcie systemu Struts

Jakarta Struts – jak to działa

- Na początku wczytuje się pliki konfiguracyjne (m.in. struts-config.xml), które są wykorzystywane do stworzenia innych obiektów powłoki kontroli
- Wszystkie powstałe obiekty tworzą konfigurację, która zawiera między innymi wspomniane już mapowania akcji

Jakarta Struts – jak to działa

- Dla każdego żądania z zewnątrz serwlet kontrolera sprawdza w mapowaniach akcji, co powinien zrobić z konkretnym żądaniem (zwykle wywołać Akcję bądź przekazać żądanie do JSP)
- Czasem wygodnie jest najpierw przeprowadzić pewną Akcję w systemie, a dopiero potem przekazać żądanie do JSP

Jakarta Struts – jak to działa

- Najlepiej jest tak zaprojektować Akcję, aby polegała ona tylko na wywołaniu metody zawartej w odpowiednim Java Bean

- W ten sposób akcje nie definiują nam tylko rodzaj czynności, którą należy zrobić, a to „jak” zostanie to zrobione zależy od konkretnego Java Bean (obiektowe podejście)

Jakarta Struts – jak to działa

- Jakarta Struts pozwala na zastosowanie Formularzy Akcji
- Obiekty te służą przechowywaniu i walidacji danych wprowadzanych przez użytkowników
- Istnieje do nich także dostęp z obiektów Akcji oraz z JSP

Jakarta Struts – jak to działa

- Dobry przykład ich zastosowania pokazuje poniższy schemat:
- Najpierw Formularz Akcji jest używany przez JSP do pobrania danych od użytkownika
- Następnie przez obiekty Akcji, aby sprawdzić spójność tych danych i wykorzystać je
- Wreszcie znów korzysta z niego JSP aby na nowo zapełnić okna dialogowe u użytkownika

Jakarta Struts – jak to działa

- Co ciekawe, Jakarta Struts zawierają biblioteki znaczników, które umożliwiają automatyczne wypełniania pól formularzy korzystając z informacji zawartych w Formularzach Akcji
- Istnieją także znaczniki pozwalające na powiadamianie użytkownika o wykonanych akcjach

Jakarta Struts – ciekawostki

- Twórcy Struts na każdym kroku podkreślają, jak ważne było dla nich umieszczenie w systemie wsparcia dla wielu języków. W tym celu zaprojektowano system od dołu w górę tak, aby każdy najdrobniejszy nawet komponent pozwalał na korzystanie z najdogodniejszych dla użytkownika wersji zasobów (np. wyświetlanie nagłówków w tabelach stron www czy też wiadomości w wybranym przez niego języku)

Zasoby Jakarta Struts

- Zdecydowanie najpełniejsza dokumentacja znajduje się na stronie projektu:

  http://struts.apache.org/

- Mniej dokładne schematy działania:

  http://www-106.ibm.com/developerworks/library/j-struts/?dwzone=java

Zasoby Jakarta Struts

- Strona zawierająca wiele samouczków i publikacji o Jakarta Struts:

http://www.strutskickstart.com/

- Opis działania wraz z przykładami:

http://www.onjava.com/pub/a/onjava/2001/09/11/jsp_servlets.html

Zasoby Jakarta Struts

- Podstawy architektury MVC:

http://ootips.org/mvc-pattern.html

http://c2.com/cgi/wiki?ModelViewControllerHistory

http://wact.sourceforge.net/index.php/ModelViewController

http://java.sun.com/blueprints/patterns/MVC.html