Trasowanie

Trasowanie (z ang. Routing) jest mechanizmem mapującym adresy URL do odpowiadającym im akcjom kontrolera. Kiedy użytkownik klika w link, przechodzi do strony poprzez bezpośrednie wpisanie adresu lub następuje automatyczne przekierowanie rozpoczyna się nowe połączenie z serwerem. W założeniach protokołu http adres URL zawierał w sobie ścieżkę reprezentującą fizyczne położenie pliku na dysku.

TODO: Obrazek z katalogami i adresem

Takie rozwiązanie, choć w pełni dające możliwość zbudowania kompletnej aplikacji, posiada szereg wad, m.in.:

* Eksponowanie klientowi struktury i hierarchii plików oraz katalogów na serwerze. O ile prawdą jest, że dla poprawnie zabezpieczonej aplikacji powinno być obojętne czy struktura plików jest jawna czy nie, to jednak jej znajomość przez hakera jeszcze przed atakiem pozwala na jego sprawniejsze i szybsze wykonanie.
* Trudność zaimplementowania dispachowania {TODO: język} akcji kontrolera.
* Bezpośredni dostęp poprzez ścieżkę narzuca problem ograniczania dostępu do konkretnych plików, które nie powinny być dostępne dla wszystkich użytkowników, bądź tylko dla wybranej grupy.
* Nieatrakcyjny i nieprzyjazny wygląd adresów URL dla użytkownika.

Mechanizm trasowania umożliwia obejście tego problemu. Żądanie do serwera jest przechwytywane, a następnie kierowanie do warstwy trasowania. Tam tzw. Trasownik (z ang. Router) jest odpowiedzialny za przeanalizowanie żądania, nie tylko pod kątem podanego adresu, ale również użytej metody http.

TODO: przykład przechwytywania z użyciem .httpaccess

Czym jest trasa?

Trasa (ang. Route) (z ang. „szlak” lub „trakt”) definiuje jeden lub grupę adresów URL. Określa również liczne parametry jakie charakteryzują dany adres. Mechanizm trasowania po otrzymaniu do przetworzenia wysłanego żądania do serwera przegląda kolejno skonfigurowane trasy. Poszukuje tak długo, aż uda mu się dopasować parametry żądania do któregoś ze skonfigurowanych tras. Jeśli nie uda się dopasować

Definiowanie routów

Wszystkie adresy URL, których dostarcza do użytku aplikacja MVC powinny być zdefiniowane. Istnieje wiele sposobów na definiowanie tras. Niektóre frameworki umożliwiają tworzenie zasad trasowania na kilka sposobów, inne tylko jeden.

Do najpowszechniejszych sposobów definiowania tras należą:

* Definiowanie poprzez plik konfiguracyjny
* Definiowanie bezpośrednio poprzez kod programu
* Definiowanie poprzez adnotacje

Każdy sposób ma swoje wady i zalety, jednak przed porównaniem należy się głębiej z nimi zapoznać. Do porównania użyjemy przykładowej trasy, który zostanie zdefiniowana różnymi sposobami.

Cechy trasa:

* będzie odpowiedzialny do kierowania na stronę z listą samochodów
* samochody będą podzielone na kategorie
* zawiera paginację

Szablon adresu, którego chcemy użyć:

TODO: samochody/{category}/{page}

Powyższy szablon składa się z trzech sekcji:

* samochody – wymagamy, aby ta część adresu była niezmienna
* category – ta sekcja jest parametrem - dowolnym ciągiem znaków reprezentującym kategorię
* page – ta sekcja również jest parametrem - numerem strony, jest liczbą całkowitą, parametrem opcjonalnym (zakładamy, że jeśli nie zostanie podana to kierujemy na pierwszą stronę)

Przykładowe adresy spełniające powyższe wymogi:

samochody/terenowe/5

samochody/osobowe

samochody/ciężarowe/17

Chcemy, aby ta trasa kierowała do określonej akcji kontrolera:

* kontroler: Cars
* akcja: List

Posiadając powyższe wymagania jakie ma spełnić trasa możemy przejść do porównania różnych sposobów jej definicji.

Definiowanie poprzez plik konfiguracyjny

Ten sposób zakłada umieszczenie definicji tras w odrębnym pliku. W zależności od możliwości frameworku może być więcej plików niż jeden. Jego zawartość reprezentowana jest w określonym formacie np. XML, yaml.

TODO: przykład pliku konfiguracyjnego

Definiowanie bezpośrednio poprzez kod programu

W tym rozwiązaniu wykorzystuje się dostarczone klasy i ich metody do konfigurowania tras. Kod lub odwołanie do niego zazwyczaj należy umieścić w odpowiednim miejscu aplikacji (np. w bootstrapie).

TODO: przykład definiowania przez kod

Definiowanie poprzez adnotacje

Adnotacje (ang. annotations) są to metadane dołączone do kodu programu. Są atrybutami nadającymi specjalne cechy klasom, ich metodom i polom. Niektóre frameworki umożliwiają ich użycie do definiowania tras. Specjalna adnotacja wraz ze wszystkimi parametrami opisującymi trasę dołączana jest wprost do akcji kontrolera.

TODO: constrainty

Porównanie rozwiązań

Każdy z tych sposobów ma swoje wady i zalety. Najlepszym sposobem ich porównania, będzie wybranie obszarów procesu tworzenia programu, na które mają wpływ i rozpatrzenie jak się w nich sprawują.

1. Przeglądanie i wyszukiwanie tras

Podczas tworzenia aplikacji może dojść do sytuacji, w której programista przejdzie do jakiejś podstrony serwisu. Jeśli ma za zadanie zmodyfikowanie zachowania tejże strony zapewne będzie chciał przejść akcji kontrolera, która została wywołana, aby wyświetlić stronę. Jeśli projekt jest mały, to szukanie może nie być problematyczne, ale w sytuacji gdy w dużym serwisie istnieją setki tras to odnalezienie odpowiedniej może być kłopotliwe. W przypadku dostępnych narzędzi do debugowania problem może być łatwo rozwiązany. Przykładem są dodatki, które dołączają do zawartości strony pasek, który zawiera pomocne informacje przy tworzeniu strony (patrz rys. TODO\_RYS\_001).



Rys. Dodatek ZFDebug do Zend Framework 2 dodaje pasek na dole strony, na którym widać jaka akcja jakiego kontrolera jest aktualnie wykorzystywana.

Niestety czasami może nastać sytuacja, w której taki dodatek nie będzie pokazany. Przykładem może być zapytanie AJAX do serwera, które zwraca jedynie informacje w formacie JSON zamiast kodu HTML, który jest wymagany do działania narzędzia. Inna sytuacja występuje, gdy strona nie działa, a błąd powoduje niezaładowanie widoku. W takich przypadkach jeśli programista nie pamięta lub nie wie, do której akcji doprowadza adres URL, wtedy musi odnaleźć ją własnoręcznie. Jeśli zna strukturę projektu wtedy może się domyślić gdzie leży kontroler, jeśli nie, wtedy musi ręcznie przejrzeć projekt. Jeśli adres jest stały tzn. nie posiada żadnych parametrów wtedy wystarczy zwykłe wyszukiwanie. Jednak gdy adres posiada parametry wtedy odnalezienie odpowiedniej trasy może trudne.

* Definiowanie poprzez plik konfiguracyjny

Zaletą tego podejścia jest to, że dokładnie wiadomo, od czego rozpocząć poszukiwanie. Jeśli plik jest jeden to od razu wiemy, że w nim znajduje się odpowiedź. Jednak, niektóre umożliwiają i zachęcają do dzielenia aplikacji na moduły. Przykładem jest Zend Framework 2, w którym zaleca się dużą granulację modułów. Biorąc pod uwagę fakt, że w dużym serwisie takich modułów mogą być dziesiątki, a każdy moduł może posiadać konfigurację własnych tras to przeszukiwanie może potrwać długo. Jedyną pomocą jest znajomość wyrażeń regularnych i możliwość ich użycia do przeszukiwania projektu.

* Definiowanie bezpośrednio poprzez kod programu

Definiowanie poprzez kod programu pozwala na dowolność organizowania definicji tras w programie. W zależności od skomplikowania przyjętych metod odnalezienie odpowiedniej trasy może być łatwe jak w sposobie definiowania poprzez plik. Jednak, zastosowane konstrukcje programistyczne, choć pomocne w innych sytuacjach, mogą znacząco utrudnić poszukiwania.

* Definiowanie poprzez adnotacje

Choć adnotacje wydają się eleganckim rozwiązaniem, które umieszcza definicje tras przy kontrolerach, których dotyczą to powodują jeden problem. Wyszukiwanie trasy pasującej do danego adresu URL jest uciążliwe ponieważ należy przeszukać wszystkie kontrolery, których zapewne będzie jeszcze więcej niż modułów aplikacji (zazwyczaj moduł posiada więcej niż jeden kontroler). Dodatkowym utrudnieniem jest to, że adnotacje w kontrolerze przeplatają się z metodami i z innymi adnotacjami. Powoduje to wyszukiwanie konkretnej treści w szumie informacji, co jest bardzo uciążliwe.

Podsumowując, rozwiązania poprzez plik konfiguracyjny i kod programu w sytuacji, gdy nie zastosowano skomplikowanych konstrukcji, są wygodnymi i łatwymi sposobami, pozwalającymi w miarę sprawnie przeszukiwać trasy. Adnotacje z powodu dowiązania do akcji są porozrzucane po kontrolerach i ich odnajdowanie może być kłopotliwe.

1. Sprawdzenie użyć akcji

TODO: przykład przez adnotację

TODO: constrainty

Kontrolery

Model

TODO: opisać jak model w ujęciu ORM bywa mylony z modelem MVC

<http://michalorman.pl/blog/2010/03/model-widok-kontroler/>

Widok

Słownik:

Hacker

MVC

Model

Widok

Kontroler

Framework

Żądanie – Request

Paginacja