AngularJS – framework javascriptowy, zapewniający wierzchnią wartwę ponad jQuery.

Jest oparty na patternie MVC:

* Model – dane, które chcemy wyświetlać użytkownikowi lub te które użytkownik ma wprowadzać do aplikacji. Reprezentowany jest przez obiekty JSON’owe.
* View – interfejs, wyrenderowany HTML. To co użytkownik widzi i z czym wchodzi w interakcję
* Controller – zajmuje się logiką biznesową aplikacji. Decyduje, które części modelu są pokazane, jak obsługiwać walidację, wykonywać operacje biznesowe itd.

Zalety Angular’a:

* Separation of Concerns (MVC),
* Mniej linii kodu potrzebnych do wykonania tych samych operacji, co przy użyciu jQuery,
* Czyste html’owe template’y.
* Łatwa testowalność,
* Łatwa integracja z zewnętrznymi bibliotekami,

1. **Przykład – Hello World**



Co zrobiliśmy:

1. Załadowaliśmy kod źródłowy AngularJS
2. Podpieliśmy AngularJS’a

Podpięcie AngularJS’a odbywa się poprzez dyrektywę **ng-app**. Dyrektywa ta określa, która część HTML’a będzie kontrolowana/zarządzana przez Angulara. W naszym wypadku użyliśmy jej na tagu html, co oznacza, że cały dokument będzie zarządzany przez Angulara. Dyrektywę tą można również dać w określonym elemencie html’woym, wówczas tylko ten element i jego dzieci będą kontrolowane przez Angulara. Wszystko poza tym tagiem będzie nieosiągalne dla Angulara.

1. Tzw. Data binding

Input tag posiada dyrektywę **ng-model**. Mówi to Angularowi, by przechowywał wartość, która użytkownik wprowadzi w zmiennej javascriptowej o nazwie „name”.

Użyliśmy też dyrektywy **ng-bind** i podwójnych klamer **{{ }}**. Obie te „dyrektywy” mają takie samo znaczenie i są używane do pobierania wartości ze zmiennej odniesionej do **ng-model** i wyświetlenia tej wartości w tagu, przypisanym do ng-bind oraz utrzymywania tej wartości „up to date” jeśli ulegnie zmianie. Nazywa się to **one-way-binding**. Bierzemy dane z jednego źródła i automatycznie uaktualniamy DOM.

W tym wypadku za każdym razem, gdy model (dane) ulegnie zmianie, Angularowe wiązanie danych będzie skutkowała automatyczną aktualizacją Data Object Modelu.

Zaimplementowanie takiej samej funkcjonalności z jQuery wymagałoby użycia watcher’ów i listner’ów. Podczas, gdy z Angularem nie zajęło to nawet jednej linii kodu.

Cały nadmiarowy kod jest zarządzany automatycznie przez Angulara, a my możemy się skupić na implementacji logiki biznesowej naszej aplikacji. To jest właśnie moc Angular’a.

1. **Moduły**

Moduł w Angularze może być rozumiany jak javovy pakier. Jest to kontener na różne części aplikacji – controllers, services, filters, directives tec.

* Moduł może definiować własne kontrolery, servicy, filtry, dyrektywy itd., które mogą być dostępne w tym module,
* Moduł może zależeć od innego modułu,
* Moduł może być użyty przez Angulara by zpiąć aplikację. Poprzez podanie nazwy modułu do dyrektywy **ng-app** mówimy Angularowi by załadował ten moduł jako główny moduł wejściowy naszej aplikacji.

Definicja modułu:



Pierwszy argument to nazwa modułu „myapp”, drugi argument to tablica, zawierająca nazwy modułów, od których ten moduł jest zależny. Pusta tabklica oznacza brak zależności.

Poniżej widać definicję moduły xzaelżenego od modułów ui.bootstrap oraz ngResource.



Załadowanie insteniejącego modułu:

Jeśli chcemy załadować istniejący moduł zdefiniowany gdzie indziej robimy to poprzez wywołanie funkcji module z nazwą wywoływanego modułu jako parametrem.



Przykład:



Powyższy przykład definiuje moduł. AngularJS potem spina aplikację wykorzystując ten moduł poprzez dyrektywę ng-app.

**3) Kontrolery**

Kontrolery to najbarddzj użyteczna część Angulara. Są to funkcje/obiekty javascriptowe, zapewniające większość logiki związanej z interfejsem użytkownika. Mogą być rozumiane jako "sterownik" dla zmian modelu i widoku. Są bramką między modelem (danymi), a widokiem.

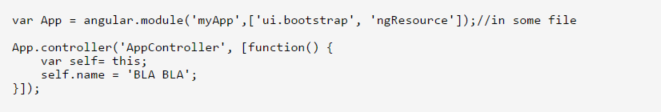
Do kluczowych zadań kontrolerów należą:

* Zapewnienie danych do interfejsu użytkownika. Dane te moga być statyczne lub pobierane z serwera.
* Zarządzanie logiką prezentacji, czyli tym jaki element w danej chwili pokazac/ukryć, czy jaki styl zastosować na konkretnym elemencie lub grupie elementów.
* Obsługa zdarzeń użytkownika, takich jak kliknięcie, walidacja danych itp.
* Przetwarzanie danych wprowadzonych przez użytkownika do wysłania do serwera.

Podstawowa składnia:



Lub jeśli moduł jest zadeklarowany w innym pliku:



Jak widać kontrolery są tworzone jako część modułów przy wykorzystaniu funkcji **controller**, która przyjmuje dwa argumenty: nazwę kontrolera oraz definicję funkcji odpowiadającej z alogikę kontrolera.

Należy zwrócić uwagę na fakt, iż definicja funkcji kontrolera znajduje się wewnątrz tablicy, a dzieje się tak dlatego, że tablica ta przechowuje wszelkie zalezności kontrolera, a ostatni argument tablicy jest własnie funkcją właściwą. W powyższym przykładzie kontroler nie ma zależności zewnętrznych.

Kontroler z zależnościami:

W tym przykładzie kontroler zależy od dwóch wbudowanych usług Angularowych **$log** odpowiedzialnej za logowanie do konsoli oraz **$location** używanej do interakcji z adresem URL w przeglądarce.

W pierwszej kolejności musimy dodać zależności jako stringi wewnątrz tablicy (taki sposób określania zależności nazywany jest "safe style od Dependency Injection"). Następnie wstrzykujemy je jako zmienne (nazwy tu nie mają znaczenia) do wnętrza funkcji, która jest przekazywana jako ostatni argument tablicy.

AngularJS wybierze poszczególne Stringi z tablicy, wyszuka wewnętrznie odpowiadające im usługi i wstrzyknie je do funkcji zgodnie z kolejnością, z która zostały podane w tablicy. Gdy usługi zostaną już wstrzyknięte, mogą być wykorzystywana w naszym kontrolerze.



W tym wypadku kontroler po prostu loguje string do konsoli i ustawia URL na /newpath.

Jak używać kontrolerów w UI?

Przy pomocy dyrektywy **ng-controller**



Dyrektywa ta jest wykorzystywana w cleu połączenia instancji kontrolera z elementem interfejsu użytkownika. W tym wypadku z tagiem body.

controllerAs syntax

Składnia wykorzystana w powyższym przykładzie "AppController as ctrl" pozwala na nadanie nazwy konkretnym instancją kontrolera (tak, może być kilka instancji tego samego kontrolera, przypisanych do różnych elementów UI). W ten sposób mamy możliwość łatwego rozróżenienia poszczególnych instancji.

$scope

Scope jest obiektem javascriptowym zawierającym własności i metody dostępnych za równo dla widoku jak i kontolera (powiązanych dyrektywą ng-controller).



Tutaj obiekt $scope został wstrzyknięty do kontrolera jako zależność. Każda zmienna i metoda do niego przypisana będzie dostępna na odpowiednim elemencie DOM'u.

$rootScope - obiekt globalny dostępny dla całej aplikacji Angularowej

**Dokładnie o $scope - https://github.com/angular/angular.js/wiki/Understanding-Scopes**

controllerAs a $scope

Składnia controllerAs pozwala na ograniczenie lub nawet wyeliminowanie użycia składni $scope i w rpzypadku bardziej rozbudowanych template'ow html'owych jest bardziej czytelna.



Kompletny przykład:



Opis:

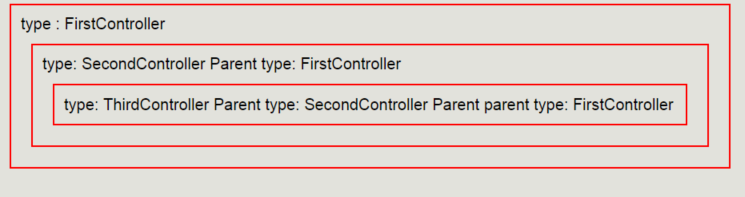
* Zaczęliśmy od definicji modułu 'myApp', użytego do spięcia aplikacji.
* Zdefiniowaliśmy kontroler 'AppController' przy użyciu funkcji **controller** w module.
* Zdefiniowaliśmy dwie zmienne name i message wewnątrz definicji kontrollera.
* Aby móc wykorzystać kontroler w UI spieliśmy go z elementem DOM'u body przy użyciu dyrektywny **ng-controller** (spieliśmy konkretną instancję kontrolera nazwaną ctrl z elementem w tym wypadku body)
* Na tagu input użyliśmy dyrektywny **ng-model**, aby przechowywac wartość wprowadzoną przez użytkownika w zmiennej kontrolera o nazwie 'name'
* Następnie wiążemy 'message' i 'name' z elementami Dom'u
* Mamy również dwa przyciski, których zachowanie zdefiniowaliśmy przy użyciu dyrektywy **ng-click**. W momencie naciśnięcia ich przez użytkownika wywołane zostaną funkcje kontrollera changeMessage() oraz resetMessage()
* Funkcja changeMessage() kontrolera ustawia wartość zmiennej 'message' na "Bye", a funkcja resetMessage() ustawia ją spowrotem na "Hello"

Kontrolery zagnieżdzone

Przy skomplikowanej strukturze, możemy się spotkać z pojęciem zagnieżdzonych kontrolerów. W takim wypadku znacznie życie ułatwia nam składnia controllerAs.



Efekt:



Należy pamiętać, że Angularowy kontroler jest bezpośrednio łączony z widokiem i jeśli zawiera jakąś logikę niewykorzystywaną przez UI, ta logika nie powinna się tam znajdować, a powinna zostać przeniesiona do Services.