Контроллеры и Scope

|  |  |  |  |
| --- | --- | --- | --- |
| № урока: | 4 | Курс: | AngularJS |
|  | | |  |
| Средства обучения: | | | Visual Studio 2013 Web Express или коммерческая версия |

# Обзор, цель и назначение урока

Получить четкое понимание о работе контроллеров и их предназначении. Понимать такие ключевые моменты как : контроллеры монолиты, наследование контроллеров, коммуникации между контроллерами, применение множественных контроллеров и проблемы которые могут возникать при таком подходе.

**Изучив материал данного занятия, учащийся сможет:**

* Создавать контроллеры в AngularJS.
* Использовать технику наследования контроллеров.
* Применять множество контроллеров.
* Использовать rootScope для коммуникации между контроллерами.
* Использовать другие JavaScript фреймворки вместе с AngularJS.

# Содержание урока

1. Создание контроллеров.
2. Контроллеры «монолиты».
3. Повторное использование контроллеров.
4. Коммуникации между контроллерами.
5. Наследование данных и поведения от других контроллеров.
6. Создание контроллеров без контекста (scope less).

# Резюме

* Контроллер – центральный элемент паттерна MVC, а также ключевой компонент AngularJS без которого невозможно построить полноценное приложение. Контроллер передает данные во View и предоставляет методы которые можно вызывать из View.
* Scope – это контекст который есть у каждого экземпляра контроллера созданного фабрикой. С помощью scope можно передавать данные из контроллера в модель. Так же существует иерархия scope где базовым элементом является rootScope. rootScope используется для коммуникации между scope из разных контроллеров.
* Контроллер монолит – это один контроллер который применяется ко всему документу. Существует несколько преимуществ такого подхода:

вам не нужно беспокоится о взаимодействии между контроллерами,

методы контроллера доступны по всей разметке. Если вы используете подход контроллера монолита, то у вас будет создана одна view.

* Когда вы применяете один и тот же контроллер несколько раз во View то это может привести к неожиданным последствиям, так как если вы применяете контроллер 2 раза, то фабричная функция, которая создает контроллеры создаст 2 независимых экземпляра контроллеров у каждого из которых будет свой scope. И это может превратится в проблему.
* Если у вас есть несколько контроллеров которые работают со своими view и нужно чтобы эти контроллеры могли общаться между собой следует применять $rootScope. Все scope организованы в иерархию которая начинается с $rootScope и каждый scope является дочерним элементом $rootScope. Так же $rootScope позволяет распространять event между scope что позволяет им взаимодействовать между собой.
* Методы для работы с событиями rootScope:
* $broadcast (name, args) отсылает event из текущего scope всем дочерним

Аргументы: 1) имя события; 2) объект, предоставляющий дополнительные данные.

* $emit (name, args) отсылает event из текущего scope к rootScope.
* $on (name, handler) регистрирует обработчик который будет вызван, когда scope инициирует необходимый event.

Аргументы: 1) название события для которого сработает обработчик; 2) функция обработчик.

* Для того чтобы избежать дублирования кода в AngularJS существует техника наследования контроллеров. Для использования этой техники следует к родительскому элементу разметки применить базовый контроллер, а к дочерним элементам контроллеры наследники, которые могут использовать функционал базового контроллера. Каждый из контроллеров обладает своим scope, но в этом scope так же присутствуют данные и поведение родительского scope.
* Для взаимодействия с другими фреймворками в AngularJS предусмотрены несколько методов которые позволяют добавлять данные в scope из других фреймворком, а также создавать функции обработчики для ответа на изменения в scope:
  + $apply (expression) применяет изменения к scope;
  + $watch (expression, handler) регистрирует обработчик (handler) который будет срабатывать при изменении выражения (expression);
  + $watchCollection (object, handler) регистрирует обработчик который будет реагировать, когда хоть какое-то свойство из коллекции свойств изменится.

# Закрепление материала

* Объясните работу контроллера монолита. Когда его следует применять?
* Какое поведение будет присуще scope из разных контроллеров?
* Как можно коммуницировать между scope из разных контроллеров?
* Какая особенность наблюдается в наследовании контроллеров, при работе с данными?
* Перечислите методы с помощью которых можно взаимодействовать с другими JavaScript фреймворками.

# Дополнительное задание

Задание

Создайте приложение в котором существует два поля для ввода и две кнопки (copy и paste).

Когда пользователь ввел данные в первое поле он может нажать на кнопку copy и данные скопируются, далее он может нажать на кнопку paste данные вставятся во второй input.

# Самостоятельная деятельность учащегося

Задание 1

Расширьте предыдущий пример используя два независимых контроллера.

Задание 2

Используя технику наследования контроллеров создайте калькулятор. Базовый контроллер содержит один метод, например, суммирования чисел. Создайте контроллеры наследники, которые бы расширяли поведение базового контроллера. При использовании последнего контроллера из цепочки наследования вам должны быть доступны все 4 операции (суммирование, вычитание, умножение и деление).

# Рекомендуемые ресурсы

Ng-controller:

<https://docs.angularjs.org/api/ng/directive/ngController>

$rootScope:

<https://docs.angularjs.org/api/ng/service/$rootScope>

Scope:

<https://docs.angularjs.org/api/ng/type/$rootScope.Scope>