Документация по реализации запросов со стороны клиента (браузера) к серверной части приложения.

Используемые языки: Javascript (Фреймворк angularJS), html (5), css

Используемые технологии: ajax, rest

Обмен данными между клиентом и сервером через ajax происходит в формате JSON

В 1 главе описан разработанный нами сервис для реализации запросов и общения с серверной частью.

Во 2 главе описана структура запросов.

В 3 главе описана структура ответов.

Конкретное апи описано в других документах.

# Angularjs cервис (на стороне клиента - браузера)

Скрипт: /services.js

## Общение с сервером

Запросы сервиса используют архитектуру REST и формат передачи данных в формате json

Список запросов реализующих общение с серверной частью:

### Server.QUERY(url, method[, data])

Абстрактный запрос на одновременную передачу и получения данных с сервера.

url - относительный или абсолютный адрес ресурса

method - метод запроса: GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH, TRACE, LINK, UNLINK, CONNECT

data - данные отправляемые на сервер (любой объект, переманная (javascript))

параметр необязателен

Возвращает объект promise

### Server.GET(url)

Абстрактный запрос на получения данных с сервера.

url - относительный или абсолютный адрес ресурса

Возвращает объект promise

### Server.POST(url[, data])

Абстрактный запрос на передачу данных на сервер.

url - относительный или абсолютный адрес ресурса

data - данные отправляемы на сервер (любой объект, переманная (javascript))

параметр необязателен

Возвращает объект promise

### Server.PUT(url[, data])

Абстрактный запрос на передачу данных на сервер.

url - относительный или абсолютный адрес ресурса

data - данные отправляемы на сервер (любой объект, переманная (javascript))

параметр необязателен

Возвращает объект promise

### Server.DELETE(url)

Абстрактный запрос на передачу данных на сервер.

url - относительный или абсолютный адрес ресурса

Возвращает объект promise

## Асинхронный обмен и сохранение данных между контроллерами AngularJS

Централизованное сохранение данных в сервисе на стороне клиента и асинхронный обмен ими между контроллерами AngularJS. В переделах приложения. До ее перезагрузки.

### Server.SetData(index, value)

Сохранение данных в сервисе

index - индекс под которым сохраняются передаваемые в хранилище данные.

value - сохраняемые данные (любой объект, переменная (javascript))

Ничего не возвращает

### Server.GetData(index)

Получение данных из сервиса

index - индекс под которым сохраняются передаваемые в хранилище данные.

Возвращает сохраненные данные по указанному индексу

### Server.RemData(index)

Удаление данных из сервиса

index - индекс под которым сохраняются передаваемые в хранилище данные.

Ничего не возвращает

## Синхронный обмен данными между контроллерами AngularJS

Создание канала и подписка на каналы в сервисе для синхронного обмена данными в случае их изменения.

### Server.SubscribeEvent(chanel[, param …])

Событие канала.

chanel - имя канала (произвольная строка)

param - произвольный набор параметров через запятую. Параметры передаются всем функциям, которые подписаны на указанный канал (методом 1.3.2).

### Server.Subscribe(chanel, function)

Подписка на канал

chanel - имя канала (произвольная строка)

function - функция которая будет вызвана при возникновении события в канале. Ей будут переданы данные посылаемые в этот канал (методом 1.3.1).

# Структура запросов

Описание запросов:

1. Методы - (GET, POST, PUT, DELETE, OPTIONS, ~~HEAD, PATCH, TRACE, LINK, UNLINK, CONNECT~~)
2. Запрос - url адрес ресурса
3. Стандартные параметры запроса в адресной строке

Синтаксис: http://www.domain.ru/api/pagename/7...?param=value

1. Данные запроса

Это простая переменная, объект либо список (массив) объектов в формате json

int | float | string - простая переменная

{“Id” int, “Name” string, “Price” float, “Flag” bool, …} - объект

[{“Id” int, “Name” string, “Price” float, “Flag” bool }] - список объектов

Объекты в свою очередь помимо простых значений могут включать другие объекты либо списки объектов.

# Структура ответов

Описание ответов:

Ответ на все ajax запросы приходит в следующей структуре:

{“Code” int, “Message” string, “MessageDebug” string, “Data” mixed}

Code - код ошибки (200 – ошибок нет, 200 < код ошибки)

Message - сообщение об операции

MessageDebug - сообщение об операции (техническое для разработки и отладки)

Data - Данные любой структуры в формате json.

В конкретных описаниях апи описаны только структуры возвращаемых данных, без их значений (Data).

[] – список каких-либо данных

{} – какой-либо объект

Int, string, float, bool … - простые типы данных