ПWS

ПОИТ-4,

лабораторная 8

6 часов

**Разработка и исследование простейших JSON-RPC-сервиса**

1. Разработайте JSON-RPC-сервис с именем **JRService** на основе Web API MVC 5+.
2. **JRService** обеспечивает работу следующих методов

|  |  |  |
| --- | --- | --- |
| Имя метода и его назначение | Параметры метода | Возвращаемый тип |
| **SetM**  Сохраняет (перезаписывает) значение ***x*** во внутренней памяти сервиса (ВПС) c ключом ***k***. Возвращает записанное в память значение | string k  int x | int |
| **GetM**  Возвращает значение, записанное в ВПС с помощью **SetM** с ключом ***k*** | string k | int |
| **AddM**  Увеличить значение, записанное в ВПС с помощью ***SetM*** с ключом ***k***, на значение ***x***. Возвращает сумму записанную в ВПС. | string k  int x | int |
| **SubM**  *Уменьшить значение, з*аписанное в ВПС с помощью ***SetM*** с ключом ***k***, на значение ***x***. Возвращает разницу записанную в ВПС. | string k  int x | int |
| **MulM**  *Умножить значение, з*аписанное в ВПС с помощью ***SetM*** с ключом ***k***, на значение ***x***. Возвращает произведение, записанное в ВПС. | string k  int x | int |
| **DivM**  *Разделить значение, з*аписанное в ВПС с помощью ***SetM*** с ключом ***k***, на значение ***x***. Возвращает произведение, записанное в ВПС. | string k  int x | int |
| **ErrorExit**  Если предыдущий запрос завершился, то оставшиеся вызовы игнорируются, а все данные в ВПС очищаются. |  |  |

1. **JRService** возвращает сообщение обо всех ошибках и в соответствии с спецификацией JSON-RPC
2. **JRService** хранит состояние в ***рамках сессии***.
3. **JRService** может выполнять ***одиночные запросы*** и ***пакеты запросов***.
4. Продемонстрировать работу одиночных сервисов с помощью web-приложения, использующего AJAX-запросы (***обратите внимание в JSON-RPC применяется POST***).
5. Работу JSON-RPC-пакета продемонстрируйте с помощью POSTMAN.

### Теория, необходимая для выполнения лабораторной работы №8

#### 1. ****JSON-RPC: Определение и Принципы****

JSON-RPC — это простой, легковесный протокол удалённого вызова процедур (RPC), использующий JSON для обмена сообщениями.  
Основные особенности JSON-RPC:

* **Формат**: Запрос и ответ формируются в формате JSON.
* **Методы**: Клиент отправляет запрос с указанием имени метода, параметров и идентификатора (id), чтобы отличить ответы.
* **Протокол**: JSON-RPC поддерживает вызов одиночных методов и пакеты запросов.
* **Ошибки**: Предусмотрена стандартная структура ошибок, позволяющая однозначно идентифицировать проблемы.

Пример запроса:

json

Копировать код

{

"jsonrpc": "2.0",

"method": "method\_name",

"params": {"key": "value"},

"id": 1

}

Пример ответа:

json

Копировать код

{

"jsonrpc": "2.0",

"result": "response\_data",

"id": 1

}

#### 2. ****Основные элементы JSON-RPC****

* **Версия протокола**: "jsonrpc": "2.0".
* **Идентификатор запроса (id)**: Целое число, строка или null, используется для сопоставления запросов и ответов.
* **Метод (method)**: Имя удалённой процедуры.
* **Параметры (params)**: Дополнительные данные, передаваемые методу. Поддерживаются массивы или объекты.
* **Результат (result)**: Данные, возвращаемые методом при успешном выполнении.
* **Ошибка (error)**: В случае сбоя возвращается объект с описанием ошибки.

#### 3. ****Особенности реализации JRService****

Ваш JSON-RPC-сервис включает шесть методов:

1. **SetM**: Сохраняет или обновляет значение с указанным ключом.
2. **GetM**: Возвращает значение по ключу.
3. **AddM**: Увеличивает значение по ключу на указанное.
4. **SubM**: Уменьшает значение по ключу.
5. **MulM**: Умножает значение по ключу.
6. **DivM**: Делит значение по ключу (с проверкой деления на ноль).
7. **ErrorExit**: Очищает данные и завершает выполнение.

Примеры использования:

* Запрос для SetM:

json

Копировать код

{

"jsonrpc": "2.0",

"method": "setm",

"params": {"k": "a", "x": 10},

"id": 1

}

* Ответ:

json

Копировать код

{

"jsonrpc": "2.0",

"result": 10,

"id": 1

}

* Ошибка в запросе:

json

Копировать код

{

"jsonrpc": "2.0",

"error": {

"code": -32602,

"message": "Invalid params",

"data": "Division on 0 is incorrect"

},

"id": 1

}

#### 4. ****Сессии и состояние сервиса****

Ваш сервис должен сохранять данные (ключи и значения) в течение сессии пользователя. Для этого используются:

* SessionMiddleware (включено в ASP.NET Core): Позволяет сохранять данные между запросами.
* JSON-сериализация: Данные внутренней памяти конвертируются в JSON и сохраняются в сессии.

#### 5. ****Обработка одиночных и пакетных запросов****

* **Одиночный запрос**: Один JSON-объект в теле запроса.
* **Пакет запросов**: Массив JSON-объектов, обрабатываемых последовательно. Например:

json

Копировать код

[

{

"jsonrpc": "2.0",

"method": "setm",

"params": {"k": "a", "x": 10},

"id": 1

},

{

"jsonrpc": "2.0",

"method": "addm",

"params": {"k": "a", "x": 5},

"id": 2

}

]

Ответ:

json

Копировать код

[

{

"jsonrpc": "2.0",

"result": 10,

"id": 1

},

{

"jsonrpc": "2.0",

"result": 15,

"id": 2

}

]

#### 6. ****Ошибки JSON-RPC****

Ошибки описываются объектами вида:

json

Копировать код

{

"code": -32700,

"message": "Parse error",

"data": "Invalid JSON received"

}

Коды ошибок:

* -32700: Ошибка парсинга JSON.
* -32600: Неверный запрос.
* -32601: Метод не найден.
* -32602: Неверные параметры.
* -32603: Внутренняя ошибка.

#### 7. ****AJAX и Postman для тестирования****

* **AJAX**: Используйте JavaScript для отправки запросов к сервису.

javascript

Копировать код

$.ajax({

url: '/map',

type: 'POST',

contentType: 'application/json',

data: JSON.stringify({

jsonrpc: "2.0",

method: "setm",

params: { k: "test", x: 100 },

id: 1

}),

success: function(response) {

console.log(response);

},

error: function(error) {

console.error(error);

}

});

* **Postman**: Для отправки одиночных и пакетных запросов, задавая POST-запрос с телом JSON.

#### 8. ****Рекомендации****

* Используйте middleware для обработки запросов и управления состоянием.
* Следите за правильной обработкой ошибок и соответствием JSON-RPC спецификации.
* Для тестирования состояния сервиса (сессии) попробуйте передать разные значения в рамках одного сеанса пользователя.

### Дополнительные материалы:

* JSON-RPC Specification
* [ASP.NET Core Session Middleware](https://learn.microsoft.com/en-us/aspnet/core/fundamentals/app-state?view=aspnetcore-7.0)