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.

**JSON-RPC –** это stateless-протокол для создания API в стиле RPC.

На сервере есть един. endpoint, кот. принимает запросы с телом

JSON-RPC – это **стандарт**! - запрос и ответ в теле

сам json стандартизирован (rfc-4627)

**JSON-RPC**

Пусть у нас есть сервер, к-му мы м отпр rpc-запросы и получать rpc-ответы. Запросы и ответы при этом будут иметь формат json. Как будет транспортироваться объект json – спецификация не рассматривает.

{"jsonrpc": "2.0", "method": "post.like", "params": {"post": "12345"}, "id": 1}  
**Request object:** json-объект:  
\* *jsonrpc* (версия)  
\* *method* (как мы сами назвали на стороне С процедуру)  
\* *params* (параметры, там м нах массив или объект)  
\* *id* (уник. С выполн эти процедуру и отвеч за результаты, ссылаясь на Id – надо обесп с-му, кот обесп уникальность – GUID, напр (генер 128-бит посл-сти)) – необяз., т.к. бывают вызовы процедур не требующие ответа (уведомления)

Такой объект мб отправлен на сторону С

{"jsonrpc": "2.0", "result": {"likes": 123}, "id": 1} **Response object:**  
\* *jsonrpc* (версия протокола)  
\* *result* (сод. объект, понятный клиенту)  
\* *error* (взаимозаменяем с result: сод сообщ об ошибке)  
\* *id* (ссылается на запрос, на кот. отвеч ошибкой)

{"jsonrpc": "2.0", "error": {"code": 666, "message": "not found"}, "id": "1"} **Объект ответы (ошибка):**  
\* *code*  
\* *message*  
\* *data* (нек д-е в формате понятном клиенту, кот позв более точно идентифиц ошибку)

**Параметры:**  
\* позиционные (в params перед массив – “params”: [42,23] – это 1й, 2й парам))  
\* поименованные (“params” : {“subs” : 23, “mid” : 42} – м перед в любом порядке)

**Уведомления** (запрос без ответа): в запросе не указ id – С интерпретирует как уведомл, ответа не получ

**RPC с пакетом запросов**: [ [запрос 1], [запрос 2], [запрос 3] ] пакет м сделать с сохранением состояния в рамках ответа м. написать неск вызовов, где каждый след. использует резу-т предыдущего пакет вып в рамках одного запрос

пакет позв в один запрос собирать набор процедур м. сохранять сост в рамках пакет

Маршрутизатор вызывает action и передает туда параметры на уровне Reflection – т.е. идет сопоставление имен параметров action с именами параметров в запросе. В запросе все парам стринговые и они стараются преобразоваться к типу параметров в action. М задать в action дефолтные параметры (если параметры request не смогут преобразоваться)

Иногда неудобно, если пришел параметр не int, вместо него будет 0 (сопоставления не произошло), мы можем не понять пришел 0 или что-то левое?  
=> поэтому существует **‘int?’** (будут null-типы если ошибка преобразования)

любое приложение связ. с обработкой запроса. Эти запросы м получить с пом объекта Request.

Models.cs

Index.cshtml

JRServiceController.js