9. JSON-RPC: определение JSON-RPC-сервиса, форматы запросов и ответов, обработка ошибок, пакеты запросов, реализация JSON-RPC на платформе Web API

1. Определение JSON-RPC-сервиса

JSON-RPC – это протокол, который позволяет одной программе (клиенту) вызывать функции в другой программе (сервере) через интернет или локальную сеть. Он использует простой текстовый формат JSON для отправки запросов и получения ответов. Например, если вы хотите выполнить действие на сервере (например, добавить два числа), JSON-RPC отправляет соответствующий запрос и ждет ответ.

2. Форматы запросов и ответов

JSON-RPC использует фиксированный формат для запросов и ответов. Вот как они выглядят:

- Запрос:

```json

{

"jsonrpc": "2.0", // Версия протокола

"method": "add", // Имя функции, которую нужно вызвать

"params": [5, 7], // Аргументы для функции

"id": 1 // Уникальный идентификатор запроса

}

```

- Ответ на успешный запрос:

```json

{

"jsonrpc": "2.0", // Версия протокола

"result": 12, // Результат выполнения функции

"id": 1 // Уникальный идентификатор, совпадает с запросом

}

```

- Ответ с ошибкой:

```json

{

"jsonrpc": "2.0", // Версия протокола

"error": { // Объект ошибки

"code": -32601, // Код ошибки

"message": "Method not found" // Сообщение об ошибке

},

"id": 1 // Уникальный идентификатор

}

```

3. Обработка ошибок

Если сервер не может обработать запрос, он отправляет сообщение об ошибке. В сообщении указываются:

- Код ошибки (например, `-32600` – неверный запрос, `-32601` – метод не найден).

- Текст сообщения, объясняющий ошибку.

- Поле `id` для связи с запросом.

Пример ошибки, когда запрашиваемый метод не существует:

```json

{

"jsonrpc": "2.0",

"error": {

"code": -32601,

"message": "Method not found"

},

"id": 1

}

```

4. Пакеты запросов

JSON-RPC позволяет отправлять несколько запросов одновременно (пакет). Это полезно, если нужно сделать много действий за один раз.

- Пример пакета запросов:

```json

[

{"jsonrpc": "2.0", "method": "add", "params": [1, 2], "id": 1},

{"jsonrpc": "2.0", "method": "subtract", "params": [10, 4], "id": 2}

]

```

- Ответ сервера:

```json

[

{"jsonrpc": "2.0", "result": 3, "id": 1},

{"jsonrpc": "2.0", "result": 6, "id": 2}

]

```

Если один из запросов вызовет ошибку, сервер включит ошибку в массив ответов.

5. Реализация JSON-RPC на платформе Web API

Для реализации JSON-RPC на веб-платформе можно использовать REST API в сочетании с HTTP-запросами. Вот основные шаги:

1. Серверная часть:

Сервер получает JSON-запросы, обрабатывает их и возвращает JSON-ответы.

На Node.js можно использовать фреймворки, например, Express.

Пример реализации сервера:

```javascript

const express = require('express');

const app = express();

app.use(express.json());

app.post('/rpc', (req, res) => {

const { jsonrpc, method, params, id } = req.body;

if (method === 'add') {

const result = params[0] + params[1];

res.json({ jsonrpc: '2.0', result, id });

} else {

res.json({

jsonrpc: '2.0',

error: { code: -32601, message: 'Method not found' },

id

});

}

});

app.listen(3000, () => console.log('JSON-RPC Server is running on port 3000'));

```

2. Клиентская часть:

Клиент отправляет запросы на сервер с помощью HTTP-запросов.

Пример клиента:

```javascript

const fetch = require('node-fetch');

const request = {

jsonrpc: '2.0',

method: 'add',

params: [5, 7],

id: 1

};

fetch('http://localhost:3000/rpc', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(request)

})

.then(res => res.json())

.then(data => console.log('Response:', data));

```

JSON-RPC прост в использовании, но мощен благодаря возможности обработки ошибок и работы с пакетами запросов.