cpprest/json

Manejo de JSON con C++ REST SDK (<cpprest/json.h>)

El módulo <cpprest/json.h> de cpprestsdk permite manejar datos JSON en C++ de manera sencilla. Se usa la clase web::json::value para crear, leer, modificar y manipular estructuras JSON.

1. Crear un JSON en C++

```
#include <cpprest/json.h>
#include <iostream>

using namespace web;

int main() {
    json::value obj;
    obj[U("nombre")] = json::value::string(U("Juan Pérez"));
    obj[U("edad")] = json::value::number(30);
    obj[U("activo")] = json::value::boolean(true);
    obj[U("hobbies")] = json::value::array({ json::value::string(U("Fútbol")), json::value::string(U
("Ajedrez")) });

    std::wcout << obj.serialize() << std::endl; // Imprimir JSON como string
    return 0;
}
```

Salida:

```
{"nombre":"Juan Pérez","edad":30,"activo":true,"hobbies":["Fútbol","Ajedrez"]}
```

2. Leer un JSON desde un std::wstring

```
#include <cpprest/json.h>
#include <iostream>

using namespace web;

int main() {
    std::wstring jsonStr = L"{\"nombre\":\"Ana\",\"edad\":25,\"activo\":false}";
    json::value obj = json::value::parse(jsonStr);

    std::wcout << U("Nombre: ") << obj[U("nombre")].as_string() << std::endl;
    std::wcout << U("Edad: ") << obj[U("edad")].as_integer() << std::endl;
    std::wcout << U("Activo: ") << obj[U("activo")].as_bool() << std::endl;
```

```
    return 0;
}
```

Salida:

```
Nombre: Ana
Edad: 25
Activo: 0
```

3. Modificar un JSON

```
#include <cpprest/json.h>
#include <iostream>

using namespace web;

int main() {
    json::value obj;
    obj[U("nombre")] = json::value::string(U("Carlos"));
    obj[U("edad")] = json::value::number(40);

    // Modificar valores
    obj[U("nombre")] = json::value::string(U("Carlos López"));
    obj[U("edad")] = obj[U("edad")].as_integer() + 1;

    std::wcout << obj.serialize() << std::endl;
    return 0;
}
```

Salida:

```
{"nombre":"Carlos López","edad":41}
```

4. Recorrer un JSON con Iteradores

```
#include <cpprest/json.h>
#include <iostream>

using namespace web;

int main() {
    json::value obj;
    obj[U("id")] = json::value::number(1);
    obj[U("nombre")] = json::value::string(U("Producto A"));
    obj[U("precio")] = json::value::number(9.99);

    for (const auto& item : obj.as_object()) {
```

```
        std::wcout << item.first << U(": ") << item.second.serialize() << std::endl;
    }

    return 0;
}
```

Salida:

```
id: 1
nombre: "Producto A"
precio: 9.99
```

5. Manejar JSON Anidados

```
#include <cpprest/json.h>
#include <iostream>

using namespace web;

int main() {
    json::value obj;
    obj[U("usuario")][U("nombre")] = json::value::string(U("Laura"));
    obj[U("usuario")][U("edad")] = json::value::number(28);
    obj[U("usuario")][U("direccion")][U("ciudad")] = json::value::string(U("Madrid"));

    std::wcout << obj.serialize() << std::endl;
    return 0;
}
```

Salida:

```
{"usuario":{"nombre":"Laura","edad":28,"direccion":{"ciudad":"Madrid"}}}
```

Métodos Útiles de `json::value`

Método	Descripción
`json::value::string(U("texto"))`	Crea un valor JSON tipo string.
`json::value::number(10)`	Crea un valor JSON tipo número.
`json::value::boolean(true)`	Crea un valor JSON tipo booleano.
`json::value::array({ json::value(1), json::value(2) })`	Crea un array JSON.
`.serialize()`	Convierte un `json::value` en string JSON.
`.as_string()`	Obtiene el valor como `std::wstring`.
`.as_integer()`	Obtiene el valor como `int`.
`.as_double()`	Obtiene el valor como `double`.
`.as_bool()`	Obtiene el valor como `bool`.

`.as_array()`	Obtiene un array como `std::vector<json::value>`.
`.as_object()`	Obtiene un objeto como `std::map<utility::string_t, json::value>`.

Para más detalles, consulta la documentación oficial: cpprestsdk en GitHub.