json0

C++ library to read/write json

@MikeNicolella

June 5, 2011

Installation

For convenience, json0 is offered as a header-only library. Add the root 'json' directory to your compiler's include paths, and you are ready to go.

Reading

To read json:

```
#include <json/json_reader.h>
```

Create a json::Value object and set it's type to Object:

```
json::Value root;
root.SetObject();
```

Pass a std::istream and your root json::Value object to json::read()

```
json::read( some_istream, root );
```

This function will either return or throw a json::JsonException. Call what() on the caught exception to get the error message.

Writing

To write json:

```
#include <json/json_writer.h>
```

Pass a std::ostream and the root json::Value object to serialize to json::write()

```
json::write( some_ostream, root );
```

The root json::Value object must be an Object value.

Example 1

The following code generates this json:

```
{
   "prop0" : "hello world"
  ,"prop1" : "hello world2"
  ,"arr0" : [ 1, true, 0.220000, null ]
  ,"one" :
   {
      "name" : "nested0"
     ,"hello" : null
     ,"probably" : true
     ,"probably not" : false
   }
}
```

```
void SampleJsonWriter()
{
  // create the root object
  json::Value root;

  // set its type to Object
  root.SetObject();

  // Object types support setting key/value members
  root.SetStringMember("prop0", "hello world");
  root.SetStringMember("prop1", "hello world2");

  // create an array
  json::Value array;
  array.SetArray();

  // add some values to the array
  array.AddInt(1);
  array.AddBool(true);
  array.AddFloat(.22f);
  array.AddNull();

  // set this array as a member of the root object. Note that doing this
  // makes a copy of the array. modifying 'array' after passing it as
  // a member will not affect the 'arr0' array of the root object.
  root.SetArrayMember("arr0", array);


  // create another object, set some members, and add it as a member of the root object
  json::Value nest;
  nest.SetObject();
  nest.SetStringMember("name", "nested0");
  nest.SetNullMember("hello");
  nest.SetBoolMember("probably", true);
  nest.SetBoolMember("probably not", false);


  // note that this copies the 'nest' object in the same way the array was copied above
  root.SetObjectMember("one", nest);

  // here we serialize the root object to std::cout
  json::write( std::cout, root );
}
```

Example 2

The following code generates this json:

```
{
  "menus" :
   [
      {
        "name" : "File"
       ,"items" :
        [
           {
             "label" : "New"
            ,"action" : "NewFile()"
           }
         , {
             "label" : "Open"
            ,"action" : "OpenFile()"
           }
        ]
      }
    , {
        "name" : "Edit"
       ,"items" :
        [
           {
             "label" : "Copy"
            ,"action" : "Copy()"
           }
         , {
             "label" : "Paste"
            ,"action" : "Paste()"
           }
        ]
      }
   ]
}
```

```
struct MenuItem
{
  std::string label;
  std::string action;

  MenuItem() {}
  MenuItem( std::string const& label, std::string const& action )
    :label(label)
    ,action(action)
  {
  }
};

struct Menu
{
  std::string name;
  std::vector<MenuItem> items;

  Menu(){}
  Menu( std::string const& name )
    :name(name)
  {
  }
};


void SampleMenuWriter()
{
  std::vector<Menu> menus;

  Menu file("File");
  file.items.push_back( MenuItem("New", "NewFile()") );
  file.items.push_back( MenuItem("Open", "OpenFile()") );

  Menu edit("Edit");
  edit.items.push_back( MenuItem("Copy", "Copy()") );
  edit.items.push_back( MenuItem("Paste", "Paste()") );

  menus.push_back(file);
  menus.push_back(edit);

  json::Value root;
  root.SetObject();

  json::Value jMenus;
  jMenus.SetArray();

  for( std::vector<Menu>::iterator menuIter = menus.begin(); menuIter != menus.end(); +
+menuIter )
  {
    json::Value jMenu;
    jMenu.SetObject();
    jMenu.SetStringMember("name", menuIter->name);

    json::Value jMenuItems;
    jMenuItems.SetArray();

    for( std::vector<MenuItem>::iterator itemIter = menuIter->items.begin(); itemIter !=
menuIter->items.end(); ++itemIter )
    {
      json::Value jItem;
      jItem.SetObject();

      jItem.SetStringMember("label", itemIter->label);
      jItem.SetStringMember("action", itemIter->action);

      jMenuItems.AddObject(jItem);
    }

    jMenu.SetArrayMember("items", jMenuItems);

    jMenus.AddObject(jMenu);
  }

  root.SetArrayMember("menus", jMenus);

  json::write( std::cout, root );
}
```

Example 3

This code reads the json written from Example 2 and rebuilds the menu data

```
void SampleMenuReader(std::istream& in)
{
  std::vector<Menu> menus;

  json::Value root;
  json::read(in, root);

  json::Value const& jMenus = root.GetArrayMember("menus");

  std::size_t numMenus = jMenus.GetNumElements();
  for( std::size_t menuIdx = 0; menuIdx < numMenus; ++menuIdx )
  {
    json::Value const& jMenu = jMenus.GetElement(menuIdx);

    Menu menu;
    menu.name = jMenu.GetStringMember("name");

    json::Value const& jMenuItems = jMenu.GetArrayMember("items");
    std::size_t numItems = jMenuItems.GetNumElements();

    for( std::size_t itemIdx = 0; itemIdx < numItems; ++itemIdx )
    {
      json::Value const& jItem = jMenuItems.GetElement(itemIdx);

      MenuItem item;
      item.label = jItem.GetStringMember("label");
      item.action = jItem.GetStringMember("action");

      menu.items.push_back(item);
    }

    menus.push_back(menu);
  }

}
```

Contact

If you have any comments / suggestions / issues / bugs, you can reach me on Twitter @MikeNicolella