Dynamic C++

Or how to write python code in C++

Mathias Stearn

NYC++ Meetup – April 28, 2010

What is boost::any

- A discriminated, type-safe union of all types
  - read: a better void*
- You can assign anything to it
- It knows what you put put in
- You get out what you put in
  - No conversions between types
  - Not even numeric types
- Value semantics (copies actually copy)

Basic usage

```
boost::any a(1);
a = 1.0;
a = "1";
a = string("1");

any_cast<string>(a).c_str();
any_cast<string&>(a) = "hello";

any_cast<int>(a); //throws bad_any_cast
any_cast<int>(&a); //return null

a.type() == typeid(string);
a.type() != typeid(int);
```

Usage in collection

```
typedef std::vector<boost::any> many

many array;
array.push_back(42);
array.push_back("Mathias");
```

Usage in collection

```
typedef std::vector<boost::any> many

many array;
array.push_back(42);
array.push_back("Mathias");

BOOST_FOREACH(const any& item, many){
    if (int* i = any_cast<int>(&item))
        cout << *i << endl;
    else if (string* s = any_cast<string>(&item))
        cout << *s << endl;
    else
        assert(!"unrecognized type");
}
```

Usage in collection cont.

```
typedef std::vector<boost::any> many

many slice(many array, int skip, int limit){
    many out;
    BOOST_FOREACH(any& item, many){
      if (skip) { skip--; continue }
      if (limit-- == 0) break;

      out.push_back(item);
    }
    return out;
}
```

Virtual Methods

```
class Example{
    // This is illegal!
    template <typename T>
    virutal void doSomething(const T& anything);

    // This is ok
    virutal void doSomething(boost::any anything);
}
```

What is boost::variant

- A discriminated, type-safe union of selected types
  - read: Haskell/ML types in C++
- You declare the supported types
- Stack-based storage
- Uses visitor pattern for access
- Feels “cleaner” than boost::any

Basic usage

```
typedef boost::variant<int, double> number;
number n = 1;
n = 1.0;

cout << n << endl; // built-in

get<double>(n);
get<int>(n); // throws bad_get
```

Visitors

```
struct Square : boost::static_visitor<>{
  void operator()(int& i) { i*= i; }
  void operator()(double& d) { d *= d; }
} square;

number n = 10;
apply_visitor(square, n);
```

Nicer Visitors

```
struct Square : boost::static_visitor<>{
  void operator()(int& i) const { i*= i; }
  void operator()(double& d) const { d *= d; }
  void operator()(number& n) const {
    apply_visitor(Square(), n);
  }
} square;

number n = 10;
square(n);
```

What is BSON

- A binary JSON format used in MongoDB
- Supports nested objects and arrays
- Supports a fixed set of types
- Objects are “frozen” once created
- Nice C++ library

Basic usage

```
BSONObj obj =
    BSON( "name" << BSON( "first" << "Mathias"
                          << "last" << "Stearn")
          << "company" << "10gen"
          << "languages" << BSON_ARRAY( "C++"
                                        << "Python");
          << "minions" << 0
          )

obj["name"].type() == Object;
obj["name"]["first"].type() == String;
obj["minions"].type() == Int;

BSONObj nameobj = obj["name"].embeddedObject();
string fullname = name["first"].String() + " "
                  + name["last"].String();
```

Basic usage

Questions?