Pimpl

BoostCon 2011 LiaW

Pimpl != Pimpl Pointer

- Pimpl Pointer: by Asger Mangaard
  - Rejected in 2006
- Pimpl: by Vladimir Batov
  - In review queue, no manager

Pointer Semantics, Manual

```
class Book
{
public:
    Book(string const& title,
         string const& author);
    string title() const;
    string author() const;

    bool operator==(Book const& that) const
      { return impl_ == that.impl_; }
    bool operator!=(Book const& that) const
      { return !operator==(that); }
    operator bool() const { return impl_; }

private:
    struct Implementation;
    std::shared_ptr<Implementation> impl_;
};
```

Pointer Semantics, Pimpl

```
class Book
    : public pimpl<Book>::pointer_semantics
{
public:
    Book(string const& title,
         string const& author);
    string title() const;
    string author() const;
};
```

Value Semantics, Manual

```
class Book
{
public:
    Book(string const& title,
         string const& author);
    string title() const;
    string author() const;

    Book(const Book& o);
    Book& operator =(const Book& o);

    bool operator==(Book const& that) const;
    bool operator!=(Book const& that) const
        { return !(*this == that); }

private:
    struct Implementation;
    std::unique_ptr<Implementation> impl_;
};
```

Value Semantics, Pimpl

```
class Book
    : public pimpl<Book>::value_semantics
{
public:
    Book(string const& title,
         string const& author);
    string title() const;
    string author() const;
    bool operator==(Book const&) const;
};
```

Implementation, Manual

```
struct Book::Implementation {
    Implementation(string const& title,
                   string const& author);
    string title;
    string author;
};

Book::Book(string const& title,
           string const& author)
  : impl_(new Implementation(title, author))
{}

string Book::title() { return impl_->title; }
string Book::author() { return impl_->author; }

// Value semantics only
Book::Book(const Book& o)
  : impl_(new Implementation(o.title(),
                             o.author()))
{}
// Similar for operator =.
bool Book::operator ==(const Book& o)
    { return title() == o.title() &&
             author() == o.author(); }
```

Implementation, Pimpl

```
template <>
struct pimpl<Book>::implementation {
    implementation(string const& title,
                   string const& author);
    string title;
    string author;
};

Book::Book(string const& title,
           string const& author)
  : base(title, author)
{}

string Book::title() { return (*this)->title; }
string Book::author() { return (*this)->author; }

// Value semantics only
bool Book::operator ==(const Book& o)
    { return title() == o.title() &&
             author() == o.author(); }
```

Additional Features

- Pointer Semantics supports null, hierarchies
- Boost.Serialization support
- ~600 LoC, mostly comments, forwarding