Subject: Direct generate implement file from .lay file
Posted by ktj9 on Mon, 19 Apr 2010 15:46:43 GMT
View Forum Message <> Reply to Message

The designer is neat. However, when I tried to use code navigation, such as context go to, to understand the codes, the .lay file becomes a pain of understanding, since it requires constant mind set switching. The preprocessor uses lay.h to generate implementation, which wraps macros and templates, two scary things together. Maybe, it is just because I am not familiar with .lay concept, or it is just because I am not comfortable switching reading .lay and c++ sources. Should pure c++ implementation from .lay be helpful? Works for me...
So I implement the generator (see below). Wish it will be helpful for some others feel the similar way. And thanks to the neat design of UPP, theIDE and lay file structure, they make these all simple...

```
void LayDes::GenCppRepresentation()
{
 if (!IsNull(fileerror))
  return;
 String cppfilename(filename);
 cppfilename.Cat(".h");
 String ret;
 for (int i = 0; i< layout.GetCount(); ++i) {
  layout[i].SetCharset(charset);
  ret << layout[i].GenCppRepresentation() << "\r\n";
 }

 if (!SaveChangedFileFinish(cppfilename, ret))
  return;
}

String LayoutData::GenCppRepresentation()
{
 String out;
 out << "struct " << name << "__layid{};\r\n";
 out << "template<class T>\r\n"
   "struct With" << name << " : public T, public " << name << "__layid {\r\n"
   "\tstatic UPP::Size GetLayoutSize() {\r\n"
   "\treturn UPP::Ctrl::LayoutZoom(" << size.cx << ", " << size.cy << ");\r\n"
   "\t}\r\n";
 for (int i=0; i< item.GetCount(); ++i) {
  if (!item[i].isUntyped())
   out << "\t" << item[i].classname() << " " << item[i].var(i) << ";\r\n";
 }
 out << "};\r\n";
 out << "template <class L, class D>\r\n"
   "void InitLayout(UPP::Ctrl& parent, L& layout, D& uts, " << name << "__layid&) {\r\n"
   "\tparent.LayoutId(\"" << name << "\");\r\n";
```

```
 for (int i=0; i< item.GetCount(); ++i) {
  if (item[i].isUntyped()) {
   out << "\tuts." << item[i].var(i) << "." << item[i].param() << ";";
   out << "\tuts." << item[i].var(i) << ".LayoutId(\"" << item[i].var(i) << "\");";
   out << "\tparent.Add(uts." << item[i].var(i) << ");\r\n";
  } else {
   out << "\tlayout." << item[i].var(i) << "." << item[i].param() << ";";
   out << "\tlayout." << item[i].var(i) << ".LayoutId(\"" << item[i].var(i) << "\");";
   out << "\tparent.Add(layout." << item[i].var(i) << ");\r\n";
  }
 }
 out << "};\r\n";
 return out;
}

bool LayoutItem::isUntyped() const
{
 return type.IsEmpty();
}

String LayoutItem::var(int i) const
{
 String s = variable.IsEmpty() ? Format("dv___%d", i) : variable;
 return s;
}

String LayoutItem::param() const
{
 return SaveProperties();
}
```

To hook them into IDE, add following in LayDes::EditBar

```
 bar.Add(islayout, AK_GenCPPRep, LayImg::GenCPP(), THISBACK(GenCppRepresentation));
```

where LayImg::GenCPP is a home-made image in LayImg.
Add following to laydes.key file

```
KEY(GenCPPRep, "Generate CPP Representation", K_ALT_G)
```