Subject: Re: ConvertInt > templatable Convert<T>
Posted by mrjt on Fri, 07 Aug 2009 16:03:08 GMT
View Forum Message <> Reply to Message

I'm not really sure what problem you're trying to solve with this, though I think I see what you're trying to do.

I think there are simpler ways of achieving it though, for instance:

```
template <class DataType>
class ConvertT : public Convert
{
private:
 DataType minval;
 DataType maxval;
 bool notnull;

public:
 ConvertT() : minval(Null), maxval(Null), notnull(false) {}

 virtual Value Scan(const Value& text) const {
  if (IsNotNull() && IsNull(text)) return NotNullError();
  Value v;
  if(  (typeid(DataType) == typeid(int))
    || (typeid(DataType) == typeid(unsigned int))
    || (typeid(DataType) == typeid(short))
    || (typeid(DataType) == typeid(unsigned short))
    || (typeid(DataType) == typeid(char))
    || (typeid(DataType) == typeid(unsigned char))
   )
  {
   v = StdConvertInt().Scan(text);
  }
  else if( (typeid(DataType) == typeid(double))
    || (typeid(DataType) == typeid(float))
   )
  {
   v = StdConvertDouble().Scan(text);
  }
  else if( (typeid(DataType) == typeid(uint64))
    || (typeid(DataType) == typeid(int64))
   )
  {
   v = ConvertInt64().Scan(text); // No StdConvert64
  }
  else
   v = StdConvert().Scan(text);
```

```
  if (!IsNull(v) && !IsNull(minval) && !IsNull(maxval)) {
   DataType m = DataType(v);
   if(m >= minval && m <= maxval)
    return v;
   return ErrorValue(UPP::Format(t_("Number must be between %s and %s."), Format(minval),
Format(maxval)));
  }
  return v;
 }
 virtual Value Format(const Value& q) const {
                // You may want to check the type of q here to ensure it matches DataType
  return StdConvert().Format(q);
 }

 ConvertT<DataType>& MinMax(DataType _min, DataType _max) { minval = _min; maxval =
_max; return *this; }
 ConvertT<DataType>& Min(DataType _min)              { minval = _min; return *this; }
 ConvertT<DataType>& Max(DataType _max)              { maxval = _max; return *this; }
 ConvertT<DataType>& NotNull(bool b = true)          { notnull = b; return *this; }
 ConvertT<DataType>& NoNotNull()                     { return NotNull(false); }
 DataType         GetMin() const                 { return minval; }
 DataType         GetMax() const                 { return maxval; }
 bool         IsNotNull() const              { return notnull; }
};
```

And then use templated edit ctrls like so:

```
template <class T>
struct EditMinMaxT : public EditMinMax<T, ConvertT<T> >
{ };
```