Subject: Re: GridCtrl performance
Posted by crydev on Fri, 26 Apr 2013 11:33:14 GMT
View Forum Message <> Reply to Message

Thanks Sender Ghost. I got it to work now and it is at least 10x faster. As others may be able to get the same problem as I have, I'll post my solution here.

My problem was mainly the use of templated structs that could have 7 different types, that also needed a different cast.

```
template <String (GetData) (SearchResult& instance, const int index)>
struct MemoryBlockValueConvert : public Convert
{
 virtual Value Format(const Value& q) const
 {
  return GetData(mMemoryScanner->GetSearchResult(), int(q));
 }
};

String GetAddress(SearchResult& instance, const int index)
{
 return FormatIntHexUpper(instance.mFirstMillionResults[index]->Address, 0);
}

String GetValue(SearchResult& instance, const int index)
{
 switch (mMemoryScanner->GetSearchResult().GetResultType())
 {
  case VALUETYPE_BYTE:
   return IntStr(((MemoryBlock<Byte>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_2BYTE:
   return IntStr(((MemoryBlock<short>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_4BYTE:
   return IntStr(((MemoryBlock<int>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_8BYTE:
   return IntStr64(((MemoryBlock<__int64>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_FLOAT:
   return DblStr(((MemoryBlock<float>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_DOUBLE:
   return DblStr(((MemoryBlock<double>*)instance.mFirstMillionResults[index])->Buffer);
   break;
  case VALUETYPE_STRING:
```

```
   return ((MemoryBlock<String>*)instance.mFirstMillionResults[index])->Buffer;
   break;
  default:
   return STRING_EMPTY;
   break;
 }
}
```

The rest is the same as Sender Ghost posted. Just call:
SetVirtualCount(mFirstMillionResults.GetCount()) to add all the items to the ArrayCtrl.

Thanks again for helping me out!
