文字化けが。。。

どうすれば解消できるか調べてもらってよいですか？

おそらく、リポジトリがutf-8なのに対して、ソースがSJISであることが原因かと思われます。

> 日本語のままなので、英語に修正しました。

関数の引数についてですが、

変更がされない変数は const参照渡し

変更がされる変数は 参照渡し

としてください。

→ほかの関数も全て見直しをお願いします。

＞　了解です。すべての引数を修正しました。

メンバ変数の命名規則は統一しましょう。

頭にm\_をつけるようにしてください。

→ほかのクラスも対応お願い致します。

＞　了解です。すべての変数を修正しました。

名前をキーとして、様々な情報をmapで持つのは面白い発想です。

しかし、DBとフィーチャクラスとフィールド名はそれぞれ

それぞれ関連付いているものなので、関連付きがわかるような持ち方のほうがベターです。

例えば、POSTALPOINT側のフィールド名とPOI側のフィールド名が万が一同じで有った場合、処理が正常に動かないかと思います。

＞フィーチャクラスを呼ぶとき、関連のDBキーワードを入れることも必要です。フィーチャクラスは混ざることがありません

＞そして、本ツールはただオプションからフィールド名を保存するだけなのでフィールド同じあっても問題ありません。

doEditがfalseになることはあるのでしょうか？

＞あります。POILocationImproverにPOSTALPOINTに対してm\_dataManager->setNewDB(POSTALPOINT\_DB, PPServerName, isPP\_FGDB, !doEdit);をしておいています。POI\_INFOに対してdoEdit必要で、doEditにしています。

編集しないときにRONLYにするのは良いのですが、

ハードコーディングにしてしまうのは、良くありません。

外から指定できるようにしましょう。

そもそも、RONLYでの接続は必要なのでしょうか？

＞以上の理由で!doEditの場合はRONLYします。本ツールのDBは指定されるのでツール以内でRONLY指定しても損はありません。

CStringの変数にserverName等を入れていますが、serverName等をstd::wstringで定義する必要があるのでしょうか？最初からCStringを使用したほうが良いかと思います。

std::stringは文字列がマルチバイトである必要がある場合、std::wstringは文字列がユニコードである必要がある場合のみ使用するようにしてください。

あと、ビルド時の文字コードの設定がマルチバイトの場合は、CStringはcharの文字列になり、文字コードの設定がユニコードの場合は、wcharの文字列となります。

よって、std::wstringをCStringに入れるということは、良くないです。

＞BoostのOptionライブラリはCStringを受け入れられないため、wstringを使うはめになりました。dataManagerの関数はwstringを受けてツールの標準変数CStringに変換する経緯です。しかし、よく考えると最初からwstringをCStringにしたほうがいいので、引数を修正します

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/OptionManager.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100623)(Diff revision 1) | | | |
|  |  | **68** | \_IOCtrl->print\_no\_timestamp\_run(\_T("POILocationImprover.exe FILEVERSION:1.0.0.0 PRODUCTVERSION:1.0.0.0")); |

バイナリ名やバージョン名を直接書くのはやめましょう。

展開したWinLibを使えば、リソースファイルから自動で取得できます。

やり方は、先日送ったサンプルプログラムを参考にしてください。(CVersionで検索すれば分かるかと思います。)

オプションをenumにしたことで、オプション名を取ってくるのに処理が必要となってしまい、少し冗長なコードとなってしまっています。

＞オプションに関係がある変数などはOptionManager.cppにまとめると、冗長に引き換え管理安くなっています。

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **120** | \_bstr\_t columnFilter = m\_PPFIELD + "," + sindy::schema::global::postal\_point::kShape; |

ArcObjectsのメソッドに渡す引数は、\_bstr\_tでなく

CComBSTRを使用しましょう。

理由としては、こちらから渡しているライブラリではCComBSTRを使っているため、統一したほうが良いからです。

あと，\_bstr\_tよりCComBSTRのほうが実装が軽いそうです。

ほかの場所も\_bstr\_tのところはCComBSTRに統一してもらえますでしょうか？

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **167** | std::pair<std::map<\_bstr\_t, std::pair<double, double> >::iterator, bool> isSuccess; |
|  |  | **168** | isSuccess = PPCoordinates->insert(std::make\_pair((\_bstr\_t)postalCode.bstrVal,std::make\_pair(orgX,orgY))); |
|  |  | **169** | // if postal code is already stored |
|  |  | **170** | if (isSuccess.second == false) { |
|  |  | **171** | CString errorMsg = (CString)"Postal Code : " + postalCode.bstrVal + " has more than 1 record in POSTALPOINT,"; |
|  |  | **172** | m\_IOManager->print\_error(ErrorManager::ECode::E\_FAILED\_TO\_GET\_DATA, errorMsg); |
|  |  | **173** | return ErrorManager::RCode::R\_FAILED\_FATAL\_ERROR; |
|  |  | **174** | } |

以下のように書くことで、コードを減らせます。

if (! PPCoordinates->insert(・・・).second) {

CString errMsg = ・・・;

:

return ErrorMaanger・・・;

}

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **256** | // get objectID index |
|  |  | **257** | long objectIDIndex; |
|  |  | **258** | if (S\_OK != m\_ipPOICursor->Fields->FindField((\_bstr\_t)sindy::schema::global::poi\_info::kObjectID, &objectIDIndex)) { |
|  |  | **259** | CString errorMsg = "ObjectID field index"; |
|  |  | **260** | m\_IOManager->print\_error(ErrorManager::ECode::E\_FAILED\_TO\_GET\_DATA, errorMsg); |
|  |  | **261** | return endProgramWithError("get ObjectID field index"); |
|  |  | **262** | } |

OBJECTIDはインデックスを調べなくても取得できます。

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **269** | if (S\_OK != ipPOIFeature->get\_Value(objectIDIndex, &objectID)) { |

if (S\_OK != ipPOIFeature->get\_OID(&objectID))

でOBJECTIDは取得できます。

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **317** | if (S\_OK != m\_POIFeatureClass->Search(POIipQueryFilter, VARIANT\_FALSE, &m\_ipPOICursor)) { |

IFeatureCursorPtrをメンバ変数にすると、

使う時に既にSearch済みかどうか判断する必要があります。

IFeatureCursorPtrはメンバ変数にせず、必要なタイミングでSearchをするほうが、分かりやすいコードになります。

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **333** | // get objectID index |
|  |  | **334** | long objectIDIndex; |
|  |  | **335** | if (S\_OK != m\_ipPOICursor->Fields->FindField((\_bstr\_t)sindy::schema::global::poi\_info::kObjectID, &objectIDIndex)) { |
|  |  | **336** | CString errorMsg = "ObjectID field index"; |
|  |  | **337** | m\_IOManager->print\_error(ErrorManager::ECode::E\_FAILED\_TO\_GET\_DATA, errorMsg); |
|  |  | **338** | return endProgramWithError("get ObjectID field index"); |
|  |  | **339** | } |
|  |  | **340** | IFeaturePtr ipPOIFeature; |
|  |  | **341** | while (m\_ipPOICursor->NextFeature(&ipPOIFeature) == S\_OK && ipPOIFeature) { |
|  |  | **342** | // get objectID |
|  |  | **343** | \_variant\_t objectID; |
|  |  | **344** | if (S\_OK != ipPOIFeature->get\_Value(objectIDIndex, &objectID)) { |
|  |  | **345** | CString errorMsg = (CString)"POI\_INFO OBJECTID"; |
|  |  | **346** | m\_IOManager->print\_error(ErrorManager::ECode::E\_FAILED\_TO\_GET\_DATA, errorMsg); |
|  |  | **347** | continue; |
|  |  | **348** | } |

ここもget\_OIDでobjectidを取得するようにしましょう。

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **373** | int returnStatus = copyPostalPointCoordinates(objID, convertedPostalCode, ipPOIGeom, PPCoordinates); |

copyPostalPointCoordinatesがこの関数の前のtestのためにgetTargetPOIRecordでも呼ばれていますが、必要あるでしょうか？

→この関数だけで呼んで、失敗した場合はstoreしなければ良いだけではないでしょうか？

　→storeしなくても、座標が変わっている可能性はあるため、ツール実行後の目視確認は必須となります。

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **432** | return getTargetPOIRecord(&PPCoordinates, POIipQueryFilter); |

return時にgetTargetPOIRecord()を呼び出し、

更にgetTargetPOIRecord()のreturn時に、

updateToDatabase()を呼び出していることで、

コードが少し分かりづらくなっています。

以下のように書いたほうが、ほかの人にとって分かりやすくなります。

if ( is\_success != getRagetPOIRecord(・・・)) {

エラー処理

}

if ( is\_success != updateToDatabase(・・・)) {

　エラー処理

}

return ErrorManager::RCode::R\_SUCCESS

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **439** | ErrorManager \_errorManager; |
|  |  | **440** | IOManager \_IOManager; |
|  |  | **441** | OptionManager \_OptionManager; |
|  |  | **442** | DataManager \_dataManager; |
|  |  | **443** | setErrorManager(&\_errorManager); |
|  |  | **444** | setIOManager(&\_IOManager); |
|  |  | **445** | setOptionManager(&\_OptionManager); |
|  |  | **446** | setDataManager(&\_dataManager); |
|  |  | **447** |  |
|  |  | **448** | m\_errorManager->initErrorMessage(); |
|  |  | **449** | m\_IOManager->setErrorManager(m\_errorManager); |
|  |  | **450** | m\_OptionManager->setLogger(m\_IOManager); |
|  |  | **451** | m\_dataManager->setLogger(m\_IOManager); |

これらの処理は、インスタンスを生成しているだけなので、コンストラクタとかで実施したほうが、コードがすっきりします。

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **448** | m\_errorManager->initErrorMessage(); |

m\_errorManagerをなぜPOILocationImproverのメンバ変数にしているのでしょうか？

初期化して、m\_IOManagerに渡しているので、m\_IOManagerで定義すれば良いのではないでしょうか？

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **450** | m\_OptionManager->setLogger(m\_IOManager); |

複数のクラスにm\_IOManagerを渡していますが、

こういう場合は、singletonというやり方が効果的ですので、今後のために、勉強してみてください。

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/POILocationImprover.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100625)(Diff revision 1) | | | |
|  |  | **452** | //Init option |
|  |  | **453** | if (is\_success != m\_OptionManager->initOption()) { |
|  |  | **454** | return endProgramWithError("init options"); |
|  |  | **455** | } |
|  |  | **456** | //Get option list from console |
|  |  | **457** | if (is\_success != m\_OptionManager->getOption(argc, argv)) { |
|  |  | **458** | return endProgramWithError("getting options"); |
|  |  | **459** | } |
|  |  | **460** | //Parse all option into data manager |
|  |  | **461** | if (is\_success != setNewOption()) { |
|  |  | **462** | return endProgramWithError("setting options"); |
|  |  | **463** | } |
|  |  | **464** | //Create files specified in option |
|  |  | **465** | if (is\_success != m\_dataManager->createFiles()) { |
|  |  | **466** | return endProgramWithError("create files"); |
|  |  | **467** | } |
|  |  | **468** | //Print specified option into run log |
|  |  | **469** | m\_OptionManager->printDescription(); |
|  |  | **470** | //Initialize connection to server |
|  |  | **471** |  |
|  |  | **472** | AccessSV \_accessSV; |
|  |  | **473** | setSVManager(&\_accessSV); |
|  |  | **474** | m\_dataManager->setAccSV(m\_accessSV); |
|  |  | **475** | //Initialize DBs |
|  |  | **476** | if (is\_success != m\_dataManager->initDBs()) { |
|  |  | **477** | return endProgramWithError("initialize DB"); |
|  |  | **478** | } |
|  |  | **479** |  |
|  |  | **480** | //Initialize featureclasses |
|  |  | **481** | if (is\_success != m\_dataManager->initFeatureClasses()) { |
|  |  | **482** | return endProgramWithError("initialize featureclass"); |
|  |  | **483** | } |

これらの処理は、

メインの処理の前の準備だと思うので、

init()関数みたいなものを作成し、別で行ったほうがすっきりします。

|  |  |  |  |
| --- | --- | --- | --- |
| [**http://preon/svn/source/trunk/training/p\_sereeyotin/POILocationImprover/main.cpp**](http://preon.mr.ipc.pioneer.co.jp/reviewboard/r/6495/diff/1/?file=100626)(Diff revision 1) | | | |
|  |  | **10** | return cPOILocationImprover.preStartTool(argc,argv); |

関数名にpreがあると、事前処理のように見えてしまいます。

名前を変えましょう。