MITSUBISHI

三菱Web地理情報システム構築パッケージ

PreSerV for Web

MapDataManager

Q&A 集

ver.2

はじめに

本ドキュメントは，PreSerV for Web MapDataManager について問い合わせのあった件について Q&A 形式にまとめたものです。

目次

1 技術的な質問

技術的な質問についての Q&A です。

1.1 マルチスレッドに対応しないデータベースについて

Ver.4 Ver.5

Q Microsoft Access の MDB ファイルをデータベースとして使用しています。
データベースとの接続には JDBC-ODBC ブリッジを使用しています。
これらの環境で複数ユーザから mdm サーブレットに処理要求を行うと次のような例外がスローされます。なにか回避策は無いでしょうか？

例外の内容：
関数シーケンスエラーです。

A マルチスレッドに対応していないデータベース及び JDBC ドライバを使用する事はお勧めしません。
止む終えず使用する場合、以下のような回避策があります。

１．mdm サーブレットのシングルスレッド化
以下のように mdm サーブレットを継承したクラスを作成し、mdm サーブレットの代わりとして使用します。

リスト 1　コーディング例（mdm サーブレットのシングルスレッド化）

```
// ファイル名：synServlet.java
import jp.co.melco.preserv.mdm.light.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class synServlet extends mdmServlet {
   protected synchronized void main (HttpServletRequest request,
                                     HttpServletResponse response ) {
      super.main (request, response);
   }
}
```

1.2 セッション毎に情報を保持する方法について

Ver.4 Ver.5

Q mdm サーブレットでセッション毎に情報を保持して使用したいのですが、そういったことは実現可能でしょうか？実現可能な場合、どのように実現したらよいのでしょうか？

A 基本的に mdm サーブレットは１回の処理要求に対して１度の結果を返答して処理を終えるため、セッション毎に情報を保持する機能はありません。
セッション毎に情報を保持する場合、独自にカスタマイズして実現する必要があります。
図１は mdm サーブレット内の情報の格納場所と流れを示したものです。

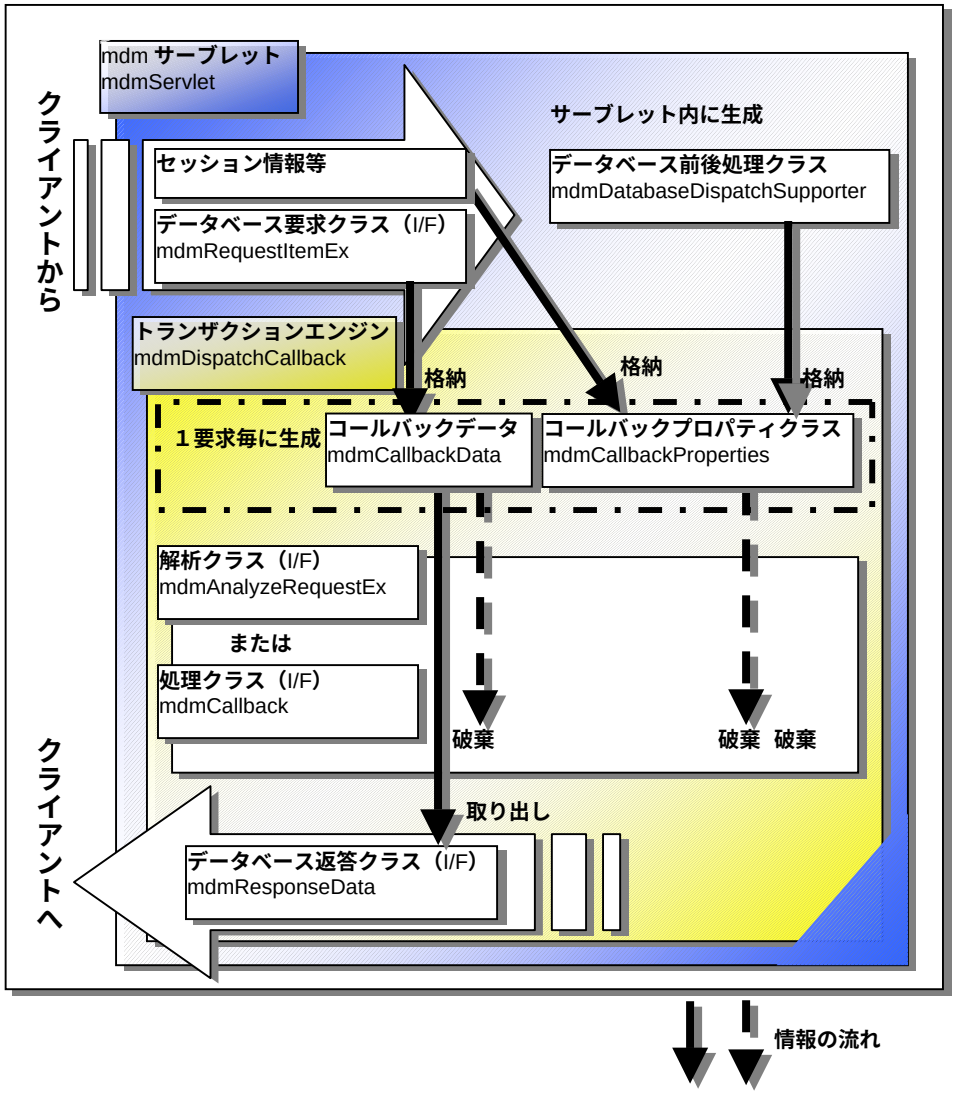

図 1-1 mdm サーブレットの情報の流れ（概略）

図 1-1 からわかる通り、ほとんどの情報はコールバックプロパティとコールバックデータの各オブジェクトに格納され、一連の処理を行った後、解放されます。サーブレット内で生成され、１要求毎に破棄されないオブジェクトは前後処理クラスです。セッション毎に値を保持するには前後処理クラスを継承して独自の情報を保持するようにカスタマイズします。

カスタマイズ後のデータの流れについてのイメージを図 1-2 に示します。

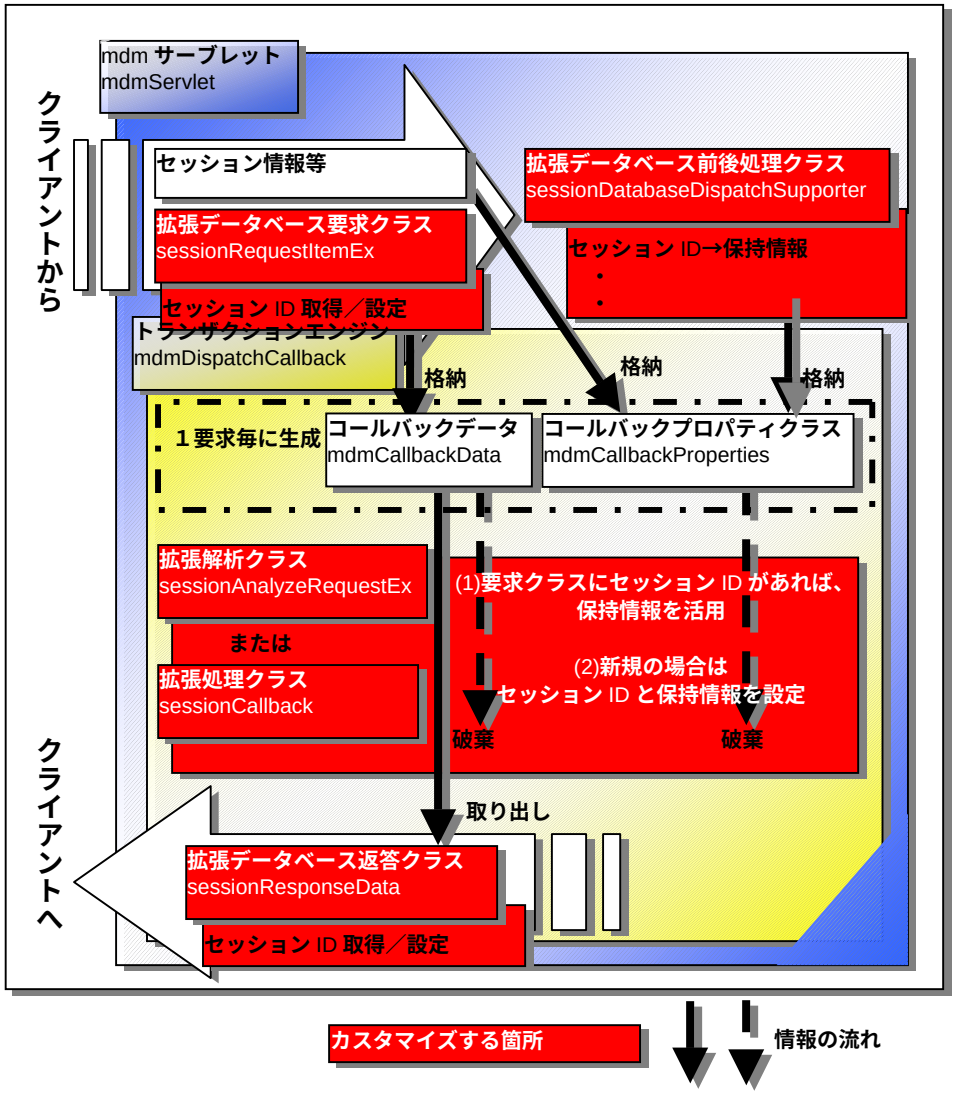

図 1-2　拡張後 mdm サーブレットの情報の流れ（概略）

セッション ID に対応した情報を保持する前後処理クラスの構築例と解析クラスや処理クラスでセッション情報の取得、およびセッション毎の情報の設定例を以下に示します。

リスト 2　コーディング例（セッションに対応した情報保持するクラス）

```
// セッション ID に対応した除法を保持する前後処理クラス例
// ファイル名：sessionSampleDispatchSupporter.java

   ・・・（省略）・・・

public class sessionSampleDispatchSupporter
   extends mdmDatabaseDispatchSupporter {

   // クライアント毎のデータ保持用ハッシュテーブル
   Hashtable _hash = null;

   ・・・（省略）・・・

   // ID に対応した情報を追加します。
   public void add_info (String id, String[] info) {
      Vector vect = (Vector)_hash.get(id);
      if (vect==null) vect = new Vector();
      vect.addElement(info);
      _hash.put (id, vect);
   }

   // ID に対応した情報を取得します。
   public Vector get_info (String id) {
      return (Vector)_hash.get (id);
   }
}
```

リスト 3　コーディング例（セッション情報取得、対応情報設定）

```
// 解析クラスでセッション情報の取得、対応情報の設定例
// ファイル名：sessionSampleAnalyzeRequestEx.java

   ・・・（省略）・・・

public class sessionSampleAnalyzeRequestEx
   implements mdmAnalyzeRequestEx {

   ・・・（省略）・・・

   public Statement[] toStatement (
      mdmCallbackData data, mdmCallbackProperties prop, Connection con)
      throws java.lang.Exception {

   ・・・（省略）・・・

      // セッション情報保持前後処理クラス取得
      sessionSampleDispatchSupporter ssds =
(sessionSampleDispatchSupporter)prop.get(prop.DISPATCH_SUPPORTER);

      // セッション ID 取得
      HttpServletRequest hsRequest =
 (HttpServletRequest)prop.get(prop.HTTP_SERVLET_REQUEST);
      HttpSession hSession = hsRequest.getSession();

      // セッション ID に対応した情報の設定
      ssds.add_info (item.get_id(), [セッション ID に対応した情報]);

   }
}
```

具体的なサンプルは添付の MapDataManager Q&A **サンプル集の**”¥Sample1.2”**ディレクトリ以下を参照して下さい。**

1.3 複数のデータベース接続について

Ver.4 Ver.5

Q MapDataManager プロパティには複数のデータベース接続情報を定義できますが、これらの複数のデータベース接続情報をテーブルグループ毎に切り替えて指定することは出来るのでしょうか？

A 現状テーブルグループ単位でデータベース接続情報を切り替えて指定、使用することは出来ません。

1.4 MDM マップのトランザクションに別トランザクションを含める方法について

Ver.4 Ver.5

Q MDM マップ（psvMdmMapEx）の機能として、複数のトランザクションを登録し、まとめてトランザクションの反映（psvMdmMapEx#reflect_mdm()）を行う機能がありますが、MDM マップで提供しているメソッド（psvMdmMapEx#insert_data()、change_data()、erase_data()）で登録するトランザクション以外のトランザクションについて、同時にトランザクションの反映を行うことは出来ないでしょうか？

A 現在提供しているクラス、メソッドをそのまま利用して別のバッチ的な処理を追加する事は出来ません。しかし、独自に psvMdmMapEx をカスタマイズすることで基本情報テーブル、図形情報テーブル、サブ情報テーブルと関連のないテーブルの更新処理を要求に加える事は可能です。
reflect_mdm 内で使用している send_data(String, Vector)メソッドをカスタマイズして、独自の要求を加えます。

１． MapDataManager と関連のないテーブルへのトランザクションの追加
以下の例は send_data()メソッドをカスタマイズしたクラスとその使用例です。

リスト 4　コーディング例（send_data()メソッドのカスタマイズ）

```
// send_data()メソッドのカスタマイズ
//  ファイル名：Ex_psvMdmMapEx.java

   ・・・（省略）・・・

public class Ex_psvMdmMapEx extends psvMdmMapEx {

   ・・・（省略）・・・

   //  独自要求クラス格納用
   Vector _batch_items = new Vector();

   //  独自要求クラス追加
   public void add_batch_request(mdmRequestItem item, String gk) {
      mdmRequestSet rs = set_requestSet (item, gk);
      _batch_items.addElement(rs);
   }
```

```
   // 独自要求クラス初期化
   public void init_batch_request() {
      _batch_items = new Vector();
   }

   // 独自要求クラスを追加して要求を行います。
   public Vector send_data (String url, Vector senddata) {
      // 独自要求クラスを追加
      for (int i=0; i<_batch_items.size(); i++) {
         mdmUpdateDBRequestData urd =
            new mdmUpdateDBRequestData(
            (mdmRequestSet)_batch_items.elementAt(i));
            senddata.addElement(urd);
      }

     // 本来の send_data()メソッドの処理
     return super.send_data (url, senddata);
  }
}
```

＜使用例＞
※[独自の更新を行う為の要求クラス]については解析クラスも含め正常に動作し、プロパティに登録してあるものとします。

```
psvMdmMapEx mdm_map=MDM マップとします。

mdmRequestItem item = [独自の更新を行う為の要求クラス];
mdm_map.add_batch_request (item, [使用するテーブルグループ識別子]);
try {
   boolean ret = mdm_map.reflect_mdm();
   if (!ret) System.out.print ("reflect failed. ");
   else mdm_map.init_batch_request();
} catch(Exception e){
   e.printStackTrace() ;
}
```

1.5 図形オブジェクトを読み出す際のカスタマイズ(情報の付加)について

Ver.4 Ver.5

Q　データベースから図形オブジェクトを読み出す際、図形オブジェクトのユーザ情報に付加情報を追加することは出来るでしょうか？

A　図形情報検索用解析クラス、図形情報検索後のフィルタクラスをカスタマイズすることで実現できます。

以下のような前提の場合を例として説明します。

[前提]

- サーバ側 JDK は 1.4 とします。
- DB は Oracle として説明します。
- 以下のテーブル構成とします。

表 1-1　テーブル構成

テーブル名	テーブル実名
基本情報テーブル	BASIC01
メッシュ情報テーブル	MESH01
図形データ情報テーブル	SHAPEDATA01
サブ情報テーブル（属性情報）※	SUBDATA01

※　ただし、サブ情報テーブルは以下の構成とする。

表 1-2 サブ情報テーブル構成

カラム	型
SEQUENCE_NO	NUMBER(10)
ZOKUSEIDATA1	CHAR(20)

１．図形オブジェクトのユーザ情報に付加情報を追加する

以下の手順による実装方法を解説いたします。

（１）解析クラスの SQL 作成部分についてのカスタマイズ

（２）フィルタクラスのカスタマイズ

（３）プロパティ定義、作成クラスの配置

（１）解析クラスの SQL 作成部分についてのカスタマイズ

解析クラス（mdmAnalyzeSelectShapeDataEx）を extens して図形情報の検索 SQL にサブ情報を同時に検索するようにします。

リスト 5　SQL イメージ（図形情報検索時にサブ情報を同時に検索）

```
// ※の部分を追加する
select
   ... ,SUBDATA01.ZOKISEIDATA1（※）
from
   ... ,SUBDATA01（※）
where
   SUBDATA01.SEQUENCE_NO=SHAPEDATA01.SEQUENCE_NO      and
（※）...
```

リスト 6　コーディング例（図形情報検索時にサブ情報を同時に検索）

```
// ファイル：addSubAnalyzeSelectShapeDataEx.java

   ・・・（省略）・・・

// 図形情報検索 SQL にサブ情報を付加するサンプル
public class addSubAnalyzeSelectShapeDataEx
   extends mdmAnalyzeSelectShapeDataEx {

   ・・・（省略）・・・

   // SQL 作成メソッド
   public String[] toSQL (mdmRequestItem item) {
      String[] sql = super.toSQL(item);
      String add_select = ",SUBDATA01.ZOKUSEIDATA1 ";
      String add_from     = ",SUBDATA01 ";
      String add_where    =
" SUBDATA01.SEQUENCE_NO=SHAPEDATA01.SEQUENCE_NO and ";

      // for j2sdk1.4
      sql[0] = sql[0].replaceAll("from", add_select+"from");
      sql[0] = sql[0].replaceAll("where", add_from+"where"+add_where);

      return sql;
   }
}
```

（2）フィルタクラスのカスタマイズ

フィルタクラス（mdmFilterSelectShapeDataEx**）を継承して図形のユーザ情報にサブ情報テーブルの情報を付加します。**

リスト 7　コーディング例（ユーザ情報にサブ情報を付加）

```
// ファイル：addSubFilterSelectShapeDataEx.java

  ・・・（省略）・・・
```

```
// 図形情報検索 SQL で付加したサブ情報をユーザ情報に追加する
public class addSubFilterSelectShapeDataEx
   extends mdmFilterSelectShapeDataEx {

  ・・・（省略）・・・

   // ユーザ情報にサブ情報を付加
  public void add_attrib (psvShape shape, Object[] result) {
      super.add_attrib (shape, result);
      shape.set_values ("ZokuseiData1", result[result.length-1]);
   }
}
```

（3）プロパティ定義、作成クラスの配置

作成したクラスをサーバ側の MapDataManager が参照可能な場所に配置します。プロパティファイルを修正し、既存クラス定義を作成クラス定義に修正します。

Callback 定義の以下の個所を修正します。

リスト 8　プロパティファイル修正（カスタマイズクラスの設定）

```
// MapDataManager プロパティファイル：mdmprop.xml

[解析クラス定義修正]
<CallbackClass callbackname="AnalyzeSelectShapeDataEx"
   classify="DATABASE40">
   jp.co.melco.preserv.mdm.light.mdmAnalyzeSelectShapeDataEx
</CallbackClass>
            ↓（修正）
<CallbackClass callbackname="AnalyzeSelectShapeDataEx"
   classify="DATABASE40">
   addSubAnalyzeSelectShapeDataEx
</CallbackClass>

[フィルタクラス定義修正]
<CallbackClass callbackname="FilterSelectShapeDataEx"
   classify="STANDARD">
   jp.co.melco.preserv.mdm.light.mdmFilterSelectShapeDataEx
</CallbackClass>
            ↓（修正）
<CallbackClass callbackname="FilterSelectShapeDataEx"
   classify="STANDARD">addSubFilterSelectShapeDataEx
</CallbackClass>
```

1.6 図形オブジェクトを読み出す際のカスタマイズ(条件の付加)について

Ver.4 Ver.5

Q データベースから図形オブジェクトを読み出す際、何らかの条件を追加することは出来るでしょうか？

A 図形情報検索用解析クラスをカスタマイズすることで実現できます。
1.5 と同様の前提を例として説明します。

１．図形オブジェクトを読み出す際、何らかの条件を追加する

ここでは基本情報テーブルの基本情報拡張２（extend2）カラムの値が-1 の場合は該当する図形オブジェクトを読み込まない、といった処理を追加します。

以下の手順による実装方法を解説いたします。

（１）解析クラスの SQL 作成部分についてのカスタマイズ

（２）プロパティ定義、作成クラスの配置

（１）解析クラスの SQL 作成部分についてのカスタマイズ

解析クラス（mdmAnalyzeSelectShapeDataEx）を extens して図形情報の検索 SQL に基本情報の条件追加します。

リスト 9　SQL イメージ（図形情報検索時に条件を付加）

```
// ※の部分を追加する
select
   ...
from
   ... ,BASIC01（※）
where
BASIC01.EXTEND2!=-1 and （※）...
```

リスト 10　コーディング例（図形情報検索時に条件を付加）

```
// ファイル：addSubAnalyzeSelectShapeData2Ex.java

   ・・・（省略）・・・

// 図形情報検索 SQL に基本情報のあるカラムを条件付加
public class addSubAnalyzeSelectShapeData2Ex
```

```
   extends addSubAnalyzeSelectShapeDataEx {

   ・・・（省略）・・・

   public addSubAnalyzeSelectShapeData2Ex () {super();}

    // SQL 作成メソッド
    public String[] toSQL (mdmRequestItem item) {
       String[] sql = super.toSQL(item);
       String add_from     = ",BASIC01 ";
       String add_where    =
 " SHAPEDATA.SEQUENCE_NO=BASIC01.SEQUENCE_NO”+
 “ and BASIC01.EXTEND2!=-1 and ";

    // for j2sdk1.4
    sql[0]= sql[0].replaceAll("where", add_from+"where"+add_where);

    return sql;
   }
}
```

（2）プロパティ定義、作成クラスの配置

作成したクラスをサーバ側の MapDataManager **が参照可能な場所に配置します。プロパティファイルを修正し、既存クラス定義を作成クラス定義に修正します。**

Callback **定義の以下の個所を修正します。**

***リスト** 11　プロパティファイル修正（カスタマイズクラスの設定）*

```
// MapDataManager プロパティファイル：mdmprop.xml

[解析クラス定義修正]
<CallbackClass callbackname="AnalyzeSelectShapeDataEx"
   classify="DATABASE40">
   addSubAnalyzeSelectShapeDataEx
</CallbackClass>
          ↓（修正）
<CallbackClass callbackname="AnalyzeSelectShapeDataEx"
   classify="DATABASE40">
   addSubAnalyzeSelectShapeData2Ex
</CallbackClass>
```

1.7 解析クラスで結果を返さない処理の可否について

Ver.4 Ver.5

Q 解析クラス（mdmAnalyzeRequestEx 実装クラス）で、データベースにアクセスしない、というのは、PreSerV の仕様上不可なのでしょうか？

A 解析クラスでは１つ以上の SQL 文または Statement を返す必要があります。サーバ側で条件により DB トランザクションの発行を制御するような場合は mdmCallback インターフェース実装クラス（以下、処理クラスと表記）を作成し、その処理クラス内で判断する必要があります。

具体的なサンプルは添付の MapDataManager Q&A サンプル集の”¥Sample1.7”ディレクトリ以下を参照して下さい。

1.8 引数に出力パラメータを持つストアドプロシージャの使用可否について

Ver.4 Ver.5

Q MapDataManager で引数に出力パラメータを持つストアドプロシージャを使用することは出来ますか？

A 現状 MDM では CallableStatement について、出力パラメータを持つ形式をサポートしておりません。
また、CallableStatement を呼び出した場合、
返答クラス：
mdmUpdateResponseData
mdmUpdateResponseData#get_result()メソッドの戻り値：
-1
mdmUpdateResponseData#get_status()メソッドの戻り値：
true
となります。

1.9 図形読み込み失敗の原因調査方法について

Ver.5

Q ブラウザを再起動し同じ地域を表示しても図形が表示されません。
このような現象は、何が原因なのでしょうか？
また、原因を調査する場合どこを調べるとよいのでしょうか？

A 図形が表示されない原因にも様々に存在します。

プログラムの問題、サーブレットの問題、クライアントの問題を分けて、
下記の手順で検索系の処理が呼ばれているか確認してください。
データベースには図形が登録されている状態とします。

＊ 注：プログラム、クライアントの問題については、MapViewer Q&A 集
「図形読み込み失敗の原因調査方法について」を参照してください。

1. サーバ側の処理の確認
   a) サーブレットが参照している JAR ファイルとクライアントが使用している JAR ファイルのバージョンが異なっていると、クラスの互換が取れなくなる場合があります。
   サーブレットが参照している JAR ファイルとクライアントが使用している JAR ファイルが同一のバージョンであることを確認してください。

   b) 下記の手順でサーブレットのログの確認してください。
   サーブレットのログをクリア（サーブレット動作前にログファイルを削除または移動）しサーブレットを再起動してください。
   「再起動し同じ地域を表示」という処理を行い、検索系の処理がサーバ側で動作しているかログを確認します。

   図形の検索処理である[mdmSelectShapeExConvertShape]で検索し、その前後が、正常に処理されているか確認してください。
   又、例外は出力されていないことを確認してください。

1.10 図形が削除できない現象について

Ver.5

Q 図形の削除処理をした所、画面では削除できているのですが、ブラウザを開きなおすと、削除したはずの図形が表示されてしまいます。
どんな原因があるのでしょうか？

A

1. データベース定義の記述された mdmprop.xml ファイルに、記述されたテーブルが間違って記述されている可能性があります。
   使用するテーブルが記述されていない場合や、使用していないテーブルを記述している場合などが、ないか確認してください。

2. DB テーブル内の char のサイズが正しいか確認してください。
   データの登録時に、カラムデータが長いとテーブル内で切り捨てられる場合があります。

   例）
   カラムのサイズが 10 だとすると、図形を作成し挿入する時に[0123456789ABC]だった値は、レコードにセットされると[0123456789]となり、[ABC]の部分は切り捨てられてしまいます。
   これを値[0123456789ABC]で検索、削除を行おうとして失敗します。

1.11 データベース登録時にエラーが発生する現象について

Ver.5

Q **データベースに登録時に、時折下記のエラーが発生し** DB **登録に失敗します。**

```
jp.co.melco.preserv.mdm.mdmServlet][main]receiveerror.(requestData):
java.io.StreamCorruptedException: Type code out of range, is 47
   at java.io.ObjectInputStream.peekCode(ObjectInputStream.java)
   at java.io.ObjectInputStream.readObject(ObjectInputStream.java)
   at java.io.ObjectInputStream.readObject(ObjectInputStream.java)
   at java.io.ObjectInputStream.inputArray(ObjectInputStream.java)
                ：
                ：
```

尚、実行環境は下記の様になっています。

[**サーバ**]

アプリケーションサーバ	：WebSphere ver.4.0.1
DataBase	：Oracle 8i
Java VM	：J2RE 1.3.0 IBM build

[**クライアント**]

Java VM	：SUN J2RE 1.3.1_02

A **サーブレットに** IBM **の** VM(J2RE 1.3.0 IBM build)**を使用すると発生するようです。**VM **を変更して試してください。**

1.12 独自図形クラスをデータベース保存後に変更して復元する方法について

Ver.4 Ver.5

Q 独自図形クラス(ユーザが独自に psvShape インタフェースを実装して作成したクラス)を作成し、MDM を使用してシリアライズ方式でデータベースに保存しました。
その後､独自図形クラスに変更を加えると以前データベースに保存した独自図形クラスが復元出来ません。以前保存した独自図形クラスを復元する方法は無いのでしょうか？

A シリアライズする Java オブジェクトは、予め serialver コマンドで取得した serialVersionUID を独自クラスのソースに設定しておくことで、シリアライズの際に Java が同じクラスとして認識し処理します。
serialver コマンドについては Java SDK ドキュメントのツールドキュメントを参照して下さい。

図形オブジェクトに serialVersionUID を設定しないでデータベースに登録し、その図形クラスを変更（コンストラクタの追加等）した場合、図形オブジェクトの検索時のサーバ側ログに以下のような例外が出ている場合があります。

```
------------------------------------------------------------
<サーバログ抜粋>
------------------------------------------------------------
Local class not compatible:
      stream classdesc serialVersionUID=XXXXXXXXXXXXXXXXXXX
      local class serialVersionUID=YYYYYYYYYYYYYYYYYYY
------------------------------------------------------------
```

※但し、XXXX...及び YYYY...はオブジェクトにより変化します。

この場合は stream 側の serialVersionUID を登録後更新した独自図形オブジェクトに設定することで復元できる場合もあります。

また、serialVersionUID を設定したクラスはクライアント側、サーバ側のクラス共に同じクラスファイルに入れ替えるようにして下さい。

1.13 自由なＭＤＭキーの採番について

Ver.4 Ver.5

Q ＭＤＭキーについて、自動採番機能を使用せず、アプリケーション側で自由に指定して作成することは可能ですか？ＭＤＭが自動的に割り振ったＭＤＭキーとアプリケーション側で任意に割り振ったＭＤＭキーを混在させることは可能ですか？

A ＭＤＭキーはアプリケーションで任意に設定することは出来ません。
ＭＤＭが自動的に割り振った値を使用して下さい。

1.14 テーブルグループ単位でのＭＤＭキーの採番について

Ver.4 Ver.5

Q ＭＤＭキーについて、テーブルグループ単位にＭＤＭキーを振り出すことは可能ですか？
例：レイヤ分類（地形図、設備図等）ごとにテーブルグループを分けておき、その単位でＭＤＭキーを付番する

A MDM キーは、テーブルグループに関係なくシステムに対して図形オブジェクト単位でユニークな値となっております。
従ってテーブルグループ単位にＭＤＭキーを割り振ることは、現状できません。

1.15 独自の形式のＭＤＭキーの採番について

Ver.4 Ver.5

Q ＭＤＭキーについて、ＭＤＭキーの型またはカラム構成を変えることは可能でしょうか？
例：図形種別（CHAR(4)）＋サイクル番号(INTEGER)＋振り出し番号（INTEGER）

A 現状では、ＭＤＭキーのデータ型やカラム構成を変更して使用することはできません。

1.16 MDM キーの空き番管理について

Ver.4 Ver.5

Q ＭＤＭキー振り出しは空き番管理していますか？大量のデータを投入するためＭＤＭキーが枯渇しないか心配です。

A ＭＤＭキーは、Ｊａｖａの４バイトで約２１億まで使用可能です。またＶ５．０よりさらにＪａｖａの４バイト分追加されました（親ＭＤＭキー＋ＭＤＭキー）。
このためＭＤＭキーは枯渇しないと考えております。
また、空き番管理は行っておりません。

1.17 ＥＪＢの対応について

Ver.4 Ver.5

Q ＭＤＭには、ＥＪＢを定義し使用出来るような仕組みはありますか？

A ＭＤＭに直接ＥＪＢを定義して利用する仕組みはありません。ただし、処理クラス内からＥＪＢを呼びだすことは可能です。この場合、ＭＤＭプロパティで以下の情報を定義します。

- 初期コンテキストファクトリー名称
  (ＭＤＭプロパティの BehaviorUnit 要素 initial_context_factory 属性)
- ネーム・サービスのロケーション
  (ＭＤＭプロパティの BehaviorUnit 要素 provider_url 属性)
- セキュリティユーザ名
  (ＭＤＭプロパティの BehaviorUnit 要素 security_principal 属性)
- セキュリティパスワード
  (ＭＤＭプロパティの BehaviorUnit 要素 security_credentials 属性)

1.18 ＭＤＭで使用するテーブル名、カラム名について

Ver.4 Ver.5

Q ＭＤＭで定義する基本情報テーブルや図形情報テーブルなどの物理名を客先の規約に合わせたいのですが可能でしょうか？また、テーブルグループ単位で別の基本情報テーブル、図形情報テーブルを定義出来ますが、それら全ての物理名を変更することは可能でしょうか？

A ＭＤＭプロパティで定義することで可能です。

1.19 未使用のカラム利用について

Ver.4 Ver.5

Q 図形情報テーブル内の地図種別、サブレイヤのカラムはアプリケーションで使用してもよいでしょうか？

A 該当カラムはＭＤＭで使用していません。アプリケーション側での使用が可能です。ただし、更新処理等はユーザで作成する必要があります。

1.20 複数テーブルグループの１トランザクション化可否について

Ver.4 Ver.5

Q 複数テーブルグループのデータを１トランザクションで処理することは可能でしょうか？

A テーブルグループはテーブル構造の取得のみ使用されています。
このため複数テーブルグループに対して１トランザクションとして処理する事は可能です。

1.21 図形オブジェクトのメッシュ跨り取得仕様について

Ver.4 Ver.5

Q MDM における図形のメッシュ跨り情報は外接長方形で判定されているようです。なぜこのような仕様になっているのでしょうか？

A 跨り情報に関しては、Ver.3 の時にメッシュをまたがる図形を管理するために追加された情報です。外接長方形にで跨り情報を判定する仕様になった理由としては、「Ver.3 の開発当時(2000 年前後)は、JAVA の処理性能はまだ低く、JAVA で設計をする際は出来るだけ簡略化された処理を実装していました。

また跨りの図形も何メッシュにも跨る図形ではなく２メッシュに跨る程度の図形しか予測していなかったため、処理性能のトレードオフの面から、外接長方形の取得⇒メッシュとの重なり判定処理を行うように設計されています。

何メッシュにも跨る図形を登録する場合には不向きですので、使用するメッシュ体系を予めよくご検討下さい。

1.22 ユーザ情報のデータベース登録制御について

Ver.4 Ver.5

Q psvShape#set_value()メソッドを用いて設定したユーザ情報を持つ図形オブジェクトをシリアライズ方式でデータベースに保存します。この場合、データベースに登録する、しないを制御する方法はあるでしょうか？

A ユーザ情報クラスの psvArgs#setSerial()メソッドを利用する事でシリアライズを行うか行わないかを制御することが出来ます。シリアライズを行わない事でサーバ側にそのユーザ情報が送信されませんので、データベースにそのユーザ情報が登録されません。

1.23 クローンで作成された図形オブジェクトのデータベース登録について

Ver.4 Ver.5

Q 図形オブジェクトを Object#clone()メソッドでクローンを作成し、データベースに登録する場合の注意点はありますか？

A 図形オブジェクトのクローンを作成した場合、ユーザ情報も複製されます。この場合、ユーザ情報の値で Cloneable インタフェースを実装しないインスタンスの複製（コピー)を試みると、例外 CloneNotSupportedException がスローされます。

また、ＭＤＭキーもユーザ情報として保持しているので、データベースから検索してきた図形オブジェクトを複製する場合は、重複を避けるためにユーザ情報からＭＤＭキーを削除してから新規登録処理を行って下さい。

コーディング例：

```
psvArgs user_data = psvShape.get_values("mdm_key");
user_data.del("mdm_key");
```

1.24 コネクションプール利用時のプロパティ定義について

Ver.4 Ver.5

Q コネクションプールを利用する場合のコネクション定義方法を教えて下さい。

A MDM で JDBC 経由でのコネクション定義（MDM プロパティの DBConnection）は以下の設定項目になっております。

- ドライバ
- URL（jdbc で始まるもの）
- ユーザ
- パスワード

コネクションプール使用時は URL のみ使用し（V3.0 SP9～）、それ以外はダミー文字列を入れてください。URL にはコネクションプールをルックアップするための文字列を指定してください。

（よくサンプル提供されているソース、以下の場合_url の部分）

```
DataSource ds = (DataSource) context.lookup(_url) ;
```

例：（V4.0 インストール手順書付属プロパティ、JRun3.0 にて Oracle へのコネクションプールを定義した場合）

```
<DBConnection name="ConnectionPoolOracle">
  <driver>dummyDriver</driver>
  <dburl>java:comp/env/jdbc/CPoolOracle</dburl>
  <user>dummyUser</user>
  <password>dummyPass</password>
</DBConnection>
```

1.25 ＭＤＭのテーブルカスタマイズについて

Ver.4 Ver.5

Q 基本情報テーブルと図形情報テーブルのメッシュ番号（大縮尺）"big_scale" カラムにプライマリキー属性を追加で付けたいと思っています。
ＭＤＭの内部動作から見て問題は発生するでしょうか？

A 上記のような想定外の使用方法をされる場合について、問題が発生するか否かについては判断致しかねます。
想定外のご使用のされ方についてはサポート致しかねます旨、ご了承ください。

1.26 テーブルグループ毎の接続データベース変更可否について

Ver.4 Ver.5

Q テーブルグループ毎に接続するデータベースを変更したいのですが、可能でしょうか？

A ＭＤＭでは、テーブルグループ単位で接続するデータベースを切り替えることは出来ません。

1.27 サブ情報テーブルに対するＭＤＭ提供機能について

Ver.4 Ver.5

Q サブ情報テーブルは、アプリケーションで作成した解析クラスのみがトランザクションを行うのでしょうか？ＭＤＭが提供する機能でトランザクションを行う事はあるのでしょうか？

A 同一テーブルグループのサブ情報テーブルについては、カスケード削除という機能のみＭＤＭから操作します。

この機能は同じ MDM キーのレコードを基本情報テーブル、図形情報テーブル、サブ情報テーブルから全て削除する機能です。

例えば、あるテーブルグループの MDM キー100 番のカスケード削除を行う場合、

・基本情報テーブル（classify="BASIC"）

・図形情報テーブル（classify="SHAPE"）

・サブ情報テーブルとして定義したテーブル（classify="SUBDATAXXX" 但し、XXX は任意）

の MDM キー＝100 番のレコードについて削除を行います。

サブ情報テーブルへのアクセス（検索等）の処理についてはアプリケーション側で何らかの要求クラス、解析クラス等のロジックを必要としますので、アプリケーションのロジックからのアクセスのみとなります。

1.28 コネクションプールを利用しないコネクションについて

Ver.4 Ver.5

Q ＭＤＭは、コネクションプールを使用せずデータベースにアクセスした場合、コネクションをどのように管理しているのでしょうか？

A コネクションプールを使わない場合は、最初にデータベースにアクセスした際に作成したコネクションオブジェクトをＭＤＭが保管します。そして次回からのアクセスに同じコネクションオブジェクトを使用しています。
従って、複数クライアントからアクセスする場合は、コネクションプールの利用が必要となります。

2 その他の質問

その他の質問についての Q&A です。

改定

CHANGE

副番 REV.	項目 ITEM	内容 CONTENTS	日付 DATE
ver.1		**新規作成**	2004.4
ver.2	1.12-1.28	**内容追加**	2004.6