

日レセ アクセス クラス説明書

日本医師会総合政策研究機構

SDK_日医標準レセプトソフト_XML要素リファレンス.....	3
1 . 要素リファレンス	4
orca	4
table-name.....	4
method.....	4
functions.....	4
function	5
function-params.....	5
function-param.....	5
2 . 記述例	6
データを全件取得する	6
データを登録する.....	8
データを削除する.....	10
データを更新する.....	12
3 . コーディング例.....	14
dbsの接続	18
dbsの認証.....	18
dbsの切断.....	18
結果セットの取得.....	18
データの追加.....	19
データの更新.....	19
データの削除.....	20

Project Code OPAS

SDK_日医標準レセプトソフト_XML 要素リファレンス

OPAS SDK では XML ファイルを用い、HAKAMA (dbs) のファンクションを一連の処理 (トランザクション) として記述することができます。

XML ファイルは HAKAMA の使用定義体に対し、ペアで作成できるように設計されています。tbl_ptinf.db に対応する XML ファイルを作成する場合は、tbl_ptinf.xml というファイルを作成します。(使用定義体の拡張子を除いたファイル名と同じファイル名である必要があります。)

Project Code OPAS

1. 要素リファレンス

orca

XML のルート要素です。省略はできません。

table-name

dbms 使用定義体が定義するテーブルの名前です。

ファイル名と一致する必要があります。

method

HAKAMA のファンクションをまとめる処理を記述します。

属性名	格納オブジェクトの型	意味
name	String	トランザクション名を記述します。XML 文書内でユニークである必要があります。省略できません。

functions

ファンクションをまとめる要素です。

Project Code OPAS

function

HAKAMA (dbs) のファンクションを記述します。 <funtions>要素内で何度でも記述できます。

属性名	格納オブジェクトの型	意味
path	String	HAKAMA (dbs) のファンクションのパスを記述します。省略できません。
name	String	HAKAMA (dbs) のファンクション名を記述します。省略できません。
valueget	boolean	ファンクションを実行した結果を、トランザクションの戻り値に含めるかを記述します。true を指定した場合は戻り値に含め、false を指定した場合は戻り値に含めません。デフォルトは false です。
eofloop	boolean	状態通知コード EOF、もしくはエラーコードが返ってくるまでファンクションを実行し続けるかを記述します。true を指定した場合は EOF からエラーコードが返ってくるまでファンクションを実行し続けます。false を指定した場合は一度だけファンクションを実行します。デフォルトは false です。

function-params

ファンクションに渡すパラメータをまとめた要素です。

function-param

ファンクションに渡すパラメータです。

HAKAMA (dbs) 使用定義体に記述されているフィールド名が記述できます。

<function-params>要素内で何度でも記述できます。

Project Code OPAS

2. 記述例

データを全件取得する

データを全件取得するトランザクションの例を示します。

最初に dbs 使用定義体 tbl_ptinf.db を編集します。

以下の内容を tbl_ptinf.db に追加して下さい。

tbl_ptinf.db

```
path sample1 {
  DBSELECT      {
    DECLARE tbl_ptinf_key_csr1 CURSOR FOR
    SELECT * FROM tbl_ptinf;
  };
  DBFETCH      {
    FETCH tbl_ptinf_key_csr1
    INTO :*;
  };
};
```

Project Code OPAS

次に tbl_ptinf.xml を編集します。

以下の内容を tbl_ptinf.xml の orca 要素の子要素に追加して下さい。

tbl_ptinf.xml

```
<method name="findAll">
  <functions>
    <function path="sample1" name="DBSELECT" valueget="false"/>
    <function path="sample1" name="DBFETCH" eofloop="true" valueget="true" />
  </functions>
</method>
```

Java の API からこのトランザクションを実行する場合、method 要素の属性「findAll」を指定します。

このトランザクションは dbs 使用定義体で定義したファンクション DBSELECT、DBFETCH を実行します。

ファンクション DBSELECT は valueget 属性に「false」指定されているので、実行結果は戻り値には含まれません。

また、eofloop 属性が指定されていないので、一度しか実行されません。

ファンクション DBFETCH は eofloop 属性に「true」が指定されているので、dbs から EOF がエラーコードが返ってくるまで実行を続けます。

valueget 属性には「true」が指定されているので実行結果は戻り値に含まれます。

Project Code OPAS

データを登録する

データを全件取得するトランザクションの例を示します。

最初に dbs 使用定義体 tbl_ptinf.db を編集します。

以下の内容を tbl_ptinf.db に追加して下さい。

tbl_ptinf.db

```
path sample2 {
  DBINSERT      {
    INSERT INTO tbl_ptinf (
      HOSPID,
      PTID,
      KANANAME,
      NAME,
      SEX,
      BIRTHDAY,
      HOME_POST,
      HOME_ADRS,
      HOME_BANTI,
      HOME_TEL1
    ) VALUES (
      :HOSPID,
      :PTID,
      :KANANAME,
      :NAME,
      :SEX,
      :BIRTHDAY,
      :HOME_POST,
      :HOME_ADRS,
      :HOME_BANTI,
      :HOME_TEL1
    );
  };
};
```

Project Code OPAS

次に tbl_ptinf.xml を編集します。

以下の内容を tbl_ptinf.xml の orca 要素の子要素に追加して下さい。

tbl_ptinf.xml

```
<method name="registPatient">
  <functions>
    <function path="sample2" name="DBINSERT" valueget="false" eofloop="false">
      <function-params>
        <function-param>HOSPID</function-param>
        <function-param>PTID</function-param>
        <function-param>KANANAME</function-param>
        <function-param>NAME</function-param>
        <function-param>SEX</function-param>
        <function-param>BIRTHDAY</function-param>
        <function-param>HOME_POST</function-param>
        <function-param>HOME_ADRS</function-param>
        <function-param>HOME_BANTI</function-param>
        <function-param>HOME_TEL1</function-param>
      </function-params>
    </function>
  </functions>
</method>
```

Java の API からこのトランザクションを実行する場合、method 要素の属性「registPatient」を指定します。

このトランザクションは dbs 使用定義体で定義したファンクション DBINSERT を実行します。ファンクション DBINSERT は valueget 属性に「false」指定されているので、実行結果は戻り値には含まれません。

また、eofloop 属性が指定されていないので、一度しか実行されません。

function-params 要素が指定されているので、function-param で指定されたフィールドがファンクション DBINSERT に渡されます。

Project Code OPAS

データを削除する

データを全件取得するトランザクションの例を示します。

最初に dbs 使用定義体 tbl_ptinf.db を編集します。

以下の内容を tbl_ptinf.db に追加して下さい。

tbl_ptinf.db

```
path sample3 {
  DBDELETE      {
    DELETE FROM tbl_ptinf
    WHERE
    HOSPID =:HOSPID AND
    PTID =:PTID;
  };
};
```

Project Code OPAS

次に tbl_ptinf.xml を編集します。

以下の内容を tbl_ptinf.xml の orca 要素の子要素に追加して下さい。

tbl_ptinf.xml

```
<method name="removePatient">
  <functions>
    <function path="sample3" name="DBDELETE" valueget="false" eofloop="false">
      <function-params>
        <function-param>HOSPID</function-param>
        <function-param>PTID</function-param>
      </function-params>
    </function>
  </functions>
</method>
```

Java の API からこのトランザクションを実行する場合、method 要素の属性「removePatient」を指定します。

このトランザクションは dbs 使用定義体で定義したファンクション DBDELETE を実行します。ファンクション DBDELETE は valueget 属性に「false」指定されているので、実行結果は戻り値には含まれません。

また、eofloop 属性が指定されていないので、一度しか実行されません。

function-params 要素が指定されているので、function-param で指定されたフィールドがファンクション DBDELETE に渡されます。

Project Code OPAS

データを更新する

データを全件取得するトランザクションの例を示します。

まず dbs 使用定義体 tbl_ptinf.db を編集します。

以下の内容を tbl_ptinf.db に追加して下さい。

tbl_ptinf.db

```
path sample4 {
  DBUPDATE      {
    UPDATE tbl_ptinf SET
      HOSPID=:HOSPID,
      PTID=:PTID,
      KANANAME=:KANANAME,
      NAME=:NAME,
      SEX=:SEX,
      BIRTHDAY=:BIRTHDAY,
      HOME_POST=:HOME_POST,
      HOME_ADRS=:HOME_ADRS,
      HOME_BANTI=:HOME_BANTI,
      HOME_TEL1=:HOME_TEL1
    WHERE
      HOSPID=:HOSPID AND
      PTID=:PTID;
  };
};
```

Project Code OPAS

次に tbl_ptinf.xml を編集します。

以下の内容を tbl_ptinf.xml の orca 要素の子要素に追加して下さい。

tbl_ptinf.xml

```
<method name="updatePatient">
  <functions>
    <function path="sample4" name="DBUPDATE" valueget="false" eofloop="false">
      <function-params>
        <function-param>HOSPID</function-param>
        <function-param>PTID</function-param>
        <function-param>KANANAME</function-param>
        <function-param>NAME</function-param>
        <function-param>SEX</function-param>
        <function-param>BIRTHDAY</function-param>
        <function-param>HOME_POST</function-param>
        <function-param>HOME_ADRS</function-param>
        <function-param>HOME_BANTI</function-param>
        <function-param>HOME_TEL1</function-param>
      </function-params>
    </function>
  </functions>
</method>
```

Java の API からこのトランザクションを実行する場合、method 要素の属性「updatePatient」を指定します。

このトランザクションは dbs 使用定義体で定義したファンクション DBUPDATE を実行します。ファンクション DBUPDATE は valueget 属性に「false」指定されているので、実行結果は戻り値には含まれません。また、eofloop 属性が指定されていないので、一度しか実行されません。function-params 要素が指定されているので、function-param で指定されたフィールドがファンクション DBUPDATE に渡されます。

Project Code OPAS

3. コーディング例

サンプルプログラム

```
-----  
1: import java.io.*;  
2: import java.util.*;  
3: import java.net.*;  
4: import jp.or.med.orca.dbs.*;  
5: import jp.or.med.orca.sdk.dto.*;  
6: import jp.or.med.orca.sdk.util.*;  
7: import jp.or.med.orca.sdk.ext.*;  
8:  
9: /**  
10:  * ORCA 密結合使用例のメインクラスです。  
11:  **/  
12: public class example {  
13:  
14:     /** 環境変数名 */  
15:     public static final String ORCA_ENV_NAME = "OPAS_HOME";  
16:     /** ORCA XML デフォルトディレクトリ */  
17:     public static final String ORCA_XML_DIR = "orcaxml";  
18:     private static String user;           // dbs 認証ユーザ名  
19:     private static String password;       // dbs 認証パスワード  
20:     private static String dbDir;         // dbs 使用定義体ディレクトリ  
21:     private static String host;          // dbs サーバーアドレス  
22:     private static int port;             // dbs ポート番号  
23:     private static String version;       // dbs バージョン  
24:     private static boolean isLogin;      // ログイン状態  
25:     private static DBSAccessor accessor = null; //  
26:     /**  
27:     * dbs に対して認証を行なう。  
28:     * @exception DBSEException 認証に失敗した場合  
29:     **/  
30:     public static void login() throws Exception {  
31:         isLogin = false;  
32:         try {  
33:             host = "210.227.72.246";  
34:             port = 8013;  
35:             dbDir = OrcaUtil.getenv(ORCA_ENV_NAME) + "/" + ORCA_XML_DIR + "/";  
36:  
37:             accessor = new DBSAccessor(host, port, dbDir);  
38:             accessor.open();  
39:  
40:             user="receipt";  
41:             password="receipt";  
42:             version="1.2.0";  
43:             accessor.authenticate(user, password, version);  
44:         } catch(Exception e) {  
45:             System.out.println("ログインに失敗しました。");  
46:             throw e;  
47:         }  
48:         isLogin = true;  
49:         System.out.println("ログインに成功しました。");  
50:     }  
51:  
52:     /**  
53:     * dbs を切断する。  
*/
```

Project Code OPAS

```
54:     **/
55:     public static void dbsLogout() {
56:         if (accessor != null) {
57:             try {
58:                 accessor.dbDisconnect();
59:             } catch(Exception e) {
60:             } finally {
61:                 try {
62:                     accessor.close();
63:                 } catch(Exception ignore) {}
64:             }
65:         }
66:         accessor = null;
67:     }
68:
69:     //-----
70:     // 患者情報のコレクション（結果セット）を取得する
71:     //-----
72:     static public void findAll() throws Exception {
73:         try {
74:             //-----
75:             // 患者情報のオブジェクトを渡す
76:             //-----
77:             TblPtinfDTO dto = new TblPtinfDTO();
78:             String methodName = "findAll";
79:             DTOCollection col = accessor.executeMethod(methodName, dto);
80:             if (col != null) {
81:                 System.out.println("[ " + col.size() + " ]件取得しました。");
82:
83:                 //-----
84:                 // 患者情報を出力する
85:                 //-----
86:                 while(col.hasNext()) {
87:                     TblPtinfDTO retDTO = (TblPtinfDTO)col.next();
88:                     System.out.println("PTID=" + retDTO.getPtid());
89:                     System.out.println("PTID=" + retDTO.getPtid());
90:                     System.out.println("カナ=" + retDTO.getKananame());
91:                     System.out.println("氏名=" + retDTO.getName());
92:                 }
93:             }
94:         } catch(Exception e){
95:             System.out.println("患者情報の取得に失敗しました。");
96:             throw e;
97:         }
98:     }
99:
100:    //-----
101:    // 患者情報を追加する
102:    //-----
103:    static public void regist() throws Exception {
104:
105:        try {
106:            //-----
107:            // 値をセットする
108:            //-----
109:            TblPtinfDTO dto = new TblPtinfDTO();
110:            dto.setHospid("0");
```

Project Code OPAS

```

111:         dto.setPtid(new Integer(0));
112:         dto.setKananame("ヤマダ タロウ");
113:         dto.setName("山田 太郎");
114:         String methodName = "registPatient";
115:         DTOCollection col = accessor.executeMethod(methodName, dto);
116:
117:         if (accessor.getLastStatusCode()==0) {
118:             System.out.println("患者情報の追加に成功しました。");
119:         } else {
120:             System.out.println("患者情報の追加に失敗しました。");
121:         }
122:     } catch(Exception e) {
123:         System.out.println("患者情報の追加に失敗しました。");
124:         throw e;
125:     }
126: }
127:
128: //-----
129: // 患者情報を変更する
130: //-----
131: static public void update() throws Exception {
132:
133:     try {
134:         //-----
135:         // 条件を指定する
136:         //-----
137:         TblPtinfdTO dto = new TblPtinfdTO();
138:         dto.setHospid("0");
139:         dto.setPtid(new Integer(0));
140:         String methodName = "updatePatient";
141:         DTOCollection col = accessor.executeMethod(methodName, dto);
142:
143:         if (accessor.getLastStatusCode()==0) {
144:             // 件数が欲しいところ
145:             System.out.println("患者情報の変更成功しました。");
146:         } else {
147:             System.out.println("患者情報の変更失敗しました。");
148:         }
149:     } catch(Exception e) {
150:         System.out.println("患者情報の変更失敗しました。");
151:         throw e;
152:     }
153: }
154:
155: //-----
156: // 患者情報を削除する
157: //-----
158: static public void remove() throws Exception {
159:
160:     try {
161:         //-----
162:         // 条件を指定する
163:         //-----
164:         TblPtinfdTO dto = new TblPtinfdTO();
165:         dto.setHospid("0");
166:         dto.setPtid(new Integer(0));
167:         String methodName = "removePatient";
    
```

Project Code OPAS

```
168:         DTOCollection col = accessor.executeMethod(methodName, dto);
169:
170:         if (accessor.getLastStatusCode()==0) {
171:             // 件数が欲しいところ
172:             System.out.println("患者情報の削除に成功しました。");
173:         } else {
174:             System.out.println("患者情報の削除に失敗しました。");
175:         }
176:     } catch(Exception e) {
177:         System.out.println("患者情報の削除に失敗しました。");
178:         throw e;
179:     }
180: }
181:
182:
183: //-----
184: // 使用例。
185: //-----
186: static public void main(String args []) {
187:     try {
188:
189:         login();
190:
191:         //-----
192:         // 接続
193:         //-----
194:         if (accessor.dbOpen() == 0) {
195:             //-----
196:             // トランザクション開始
197:             //-----
198:             if (accessor.beginTransaction() == 0) {
199:                 //-----
200:                 // 患者情報を出力する
201:                 //-----
202:                 findAll();
203:             }
204:         }
205:
206:     } catch(Exception e) {
207:         e.printStackTrace();
208:         dbsLogout();
209:     }
210: }
211:
212: }
```

Project Code OPAS

dbs の接続

```
37: accessor = new DBSAccessor(host, port, dbDir);  
38: accessor.open();
```

ホスト名・ポート番号・XML ファイルのディレクトリを指定し DBSAccessor オブジェクトを作成します。データベースの接続には open メソッドを実行します。

dbs の認証

```
43:accessor.authenticate(user, password, version);
```

認証を行うには、authenticate メソッドを実行します。

User :ユーザー名

Password :パスワード名

Version :バージョン番号

dbs の切断

```
58:accessor.dbDisconnect();
```

dbs の切断を行うには、dbDisconnect メソッドを実行します。

結果セットの取得

```
77: TblPtinfDTO dto = new TblPtinfDTO();  
78: String methodName = "findAll";  
79: DTOCollection col = accessor.executeMethod(methodName, dto);
```

結果セットの取得を行う場合には、executeMethod メソッドを実行します。(79 行目)

methodName :メソッド名(ここでは、"findAll")

dto :TblPtinfDTO オブジェクト

```
86: while(col.hasNext()) {  
87: TblPtinfDTO retDTO = (TblPtinfDTO)col.next();  
88: System.out.println("PTID=" + retDTO.getPtid());  
89: System.out.println("PTID=" + retDTO.getPtid());  
90: System.out.println("カナ=" + retDTO.getKananame());  
91: System.out.println("氏名=" + retDTO.getName());  
92: }
```

各レコードのデータ(TblPtinfDTO オブジェクト)を取得するには、DTOCollection クラスの next() メソッドを実行します。(87 行目)

データの追加

```
109: TblPtinfDTO dto = new TblPtinfDTO();
110: dto.setHospid("0");
111: dto.setPtid(new Integer(0));
112: dto.setKananame("ヤマダ タロウ");
113: dto.setName("山田 太郎");
114: String methodName = "registPatient";
115: DTOCollection col = accessor.executeMethod(methodName, dto);
116:
117: if (accessor.getLastStatusCode()==0) {
118:     System.out.println("患者情報の追加に成功しました。");
119: } else {
120:     System.out.println("患者情報の追加に失敗しました。");
121: }
```

- ・まずデータを表すオブジェクトを作成します。(109~113行目)
 - ・次に作成したオブジェクトを引数にして、executeMethod メソッドを実行します。(115行目)
- methodName :実行するメソッド名(個々では"registPatient"を指定)
- dto :追加するデータ(TblPtinfDTO のオブジェクト)
- ・実行結果は、getLastStatusCode()メソッドで確認することができます。(117行目)

データの更新

```
137: TblPtinfDTO dto = new TblPtinfDTO();
138: dto.setHospid("0");
139: dto.setPtid(new Integer(0));
140: String methodName = "updatePatient";
141: DTOCollection col = accessor.executeMethod(methodName, dto);
142:
143: if (accessor.getLastStatusCode()==0) {
144:     // 件数が欲しいところ
145:     System.out.println("患者情報の変更成功しました。");
146: } else {
147:     System.out.println("患者情報の変更失敗しました。");
148: }
```

- ・更新したいデータのオブジェクトを作成して、条件値を設定します。(137~139行目)
 - ・次に作成したオブジェクトを引数にして、executeMethod メソッドを実行します。(141行目)
- methodName :実行するメソッド名(個々では"updatePatient"を指定)
- dto :追加するデータ(TblPtinfDTO のオブジェクト)
- ・実行結果は、getLastStatusCode()メソッドで確認することができます。(143行目)

データの削除

```
164:         TbIPtinfDTO dto = new TbIPtinfDTO();
165:         dto.setHospid("0");
166:         dto.setPtid(new Integer(0));
167:         String methodName = "removePatient";
168:         DTOCollection col = accessor.executeMethod(methodName, dto);
169:
170:         if (accessor.getLastStatusCode()==0) {
171:             // 件数が欲しいところ
172:             System.out.println("患者情報の削除に成功しました。");
173:         } else {
174:             System.out.println("患者情報の削除に失敗しました。");
175:         }
```

削除したいデータのオブジェクトを作成し、条件値を設定します。(164~167行目)

- ・次に作成したオブジェクトを引数にして、executeMethodメソッドを実行します。(168行目)
methodName :実行するメソッド名(個々では"removePatient"を指定)

dto :追加するデータ(TbIPtinfDTOのオブジェクト)

- ・実行結果は、getLastStatusCode()メソッドで確認することができます。(170行目)