unit ufrmPsnTrainTest;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DBCtrlsEh, StdCtrls, Mask, DBLookupEh, Buttons, DBCtrls,

CSDBNavigator, ExtCtrls, CSRowCountPanel, Grids, DBGridEh, dxEditor,

dxExEdtr, dxEdLib, CSCustomdxDateEdit, CSdxDBDateEdit, dxCntner, dxDBELib,

DB, ADODB, CSADOQuery, dxmdaset, ufrmPersonMng, ufrmOrganMng,

CSScrCtrlChild, CSBarMDIChild, PrnDbgeh, CSDBEhGrdStore, CSGrdColCtrl,

CSEhGrdColCtrl, CSADOFilter, CSADOEhGrdFilter, CSDBBase, CSDBFind,

CSDBEhGrdFind;

type

TfrmPsnTrainTest = class(TForm)

scrbDetail: TScrollBox;

Label14: TLabel;

Label19: TLabel;

Label27: TLabel;

Label30: TLabel;

Label36: TLabel;

mRemark: TdxDBMemo;

dteStartDate: TCSdxDBDateEdit;

Splitter1: TSplitter;

grdTrainList: TDBGridEh;

CSRowCountPanel1: TCSRowCountPanel;

Panel7: TPanel;

Panel8: TPanel;

Navigator: TCSDBNavigator;

bbtnEngageIns: TBitBtn;

bbtnEngageDel: TBitBtn;

bbtnFind: TBitBtn;

bbtnFilter: TBitBtn;

bbtnColunm: TBitBtn;

Label1: TLabel;

dteEndDate: TCSdxDBDateEdit;

Label37: TLabel;

lcmbtestorgan: TDBLookupComboboxEh;

Label2: TLabel;

eLResult: TDBEditEh;

Label3: TLabel;

eOResult: TDBEditEh;

Label4: TLabel;

lcmbName: TDBLookupComboboxEh;

lcmbLicnsKind: TDBLookupComboboxEh;

lcmbLicnsCategory: TDBLookupComboboxEh;

lcmbResult: TDBLookupComboboxEh;

psnTrainTest: TCSADOQuery;

psnTrainTestd: TDataSource;

Panel1: TPanel;

Panel4: TPanel;

bbtnHelp: TBitBtn;

bbtnClose: TBitBtn;

licnskind: TCSADOQuery;

licnskindd: TDataSource;

licnscategory: TCSADOQuery;

licnscategoryd: TDataSource;

trainresult: TCSADOQuery;

trainresultd: TDataSource;

Temp: TCSADOQuery;

mComBak: TdxMemData;

mCom: TdxMemData;

mComd: TDataSource;

mPersonBak: TdxMemData;

mPerson: TdxMemData;

mPersond: TDataSource;

mPersonperson\_sysid: TStringField;

mPersonperson\_sysid\_name: TStringField;

mPersonBakperson\_sysid: TStringField;

mPersonBakperson\_sysid\_name: TStringField;

ScrChild: TCSScrCtrlChild;

CSBarMDIChild1: TCSBarMDIChild;

ListFind: TCSDBEhGrdFind;

ListFilter: TCSADOEhGrdFilter;

ListColCtrl: TCSEhGrdColCtrl;

ListEhGrdStore: TCSDBEhGrdStore;

SaveDialog: TSaveDialog;

ListPrint: TPrintDBGridEh;

list: TCSADOQuery;

listd: TDataSource;

navList: TCSDBNavigator;

Label7: TLabel;

lcmblicnsprj: TDBLookupComboboxEh;

Label8: TLabel;

lcmblicnslevel: TDBLookupComboboxEh;

listsysid: TStringField;

listlicns\_prj\_sysid: TStringField;

listtestorgan\_sysid: TStringField;

listtestorgan\_sysid\_name: TStringField;

listtrainorgan\_sysid: TStringField;

listtrainorgan\_sysid\_name: TStringField;

listperson\_sysid: TStringField;

listperson\_sysid\_name: TStringField;

listlicns\_kind: TStringField;

listlicns\_kind\_name: TStringField;

listlicns\_category: TStringField;

listlicns\_category\_name: TStringField;

listlicns\_prj: TStringField;

listlicns\_prj\_name: TStringField;

listlicns\_level: TStringField;

listlicns\_level\_name: TStringField;

liststart\_date: TStringField;

listend\_date: TStringField;

listtrain\_result: TStringField;

listtrain\_result\_name: TStringField;

listtheory\_result: TStringField;

listoperate\_result: TStringField;

listtrain\_content: TStringField;

licnsprj: TCSADOQuery;

licnsprjd: TDataSource;

licnslevel: TCSADOQuery;

licnsleveld: TDataSource;

Label16: TLabel;

elicnsprjname: TDBEditEh;

Label5: TLabel;

mcontent: TdxDBMemo;

Label6: TLabel;

lcmbtrainorgan: TDBLookupComboboxEh;

mComtrainorgan\_sysid: TStringField;

mComtrainorgan\_sysid\_name: TStringField;

mComtestorgan\_sysid: TStringField;

mComtestorgan\_sysid\_name: TStringField;

mComBaktrainorgan\_sysid: TStringField;

mComBaktrainorgan\_sysid\_name: TStringField;

mComBaktestorgan\_sysid: TStringField;

mComBaktestorgan\_sysid\_name: TStringField;

listpsnlicns\_prjname: TStringField;

procedure FormCreate(Sender: TObject);

procedure lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure listAfterScroll(DataSet: TDataSet);

procedure lcmbLicnsKindKeyValueChanged(Sender: TObject);

procedure psnTrainTestAfterInsert(DataSet: TDataSet);

procedure psnTrainTestAfterPost(DataSet: TDataSet);

procedure psnTrainTestBeforePost(DataSet: TDataSet);

procedure psnTrainTestAfterCancel(DataSet: TDataSet);

procedure bbtnCloseClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure NavigatorBeforeAction(Sender: TObject; Button: TNavigateBtn);

procedure psnTrainTestAfterDelete(DataSet: TDataSet);

procedure bbtnEngageInsClick(Sender: TObject);

procedure bbtnEngageDelClick(Sender: TObject);

procedure lcmbLicnsCategoryKeyValueChanged(Sender: TObject);

procedure lcmblicnsprjKeyValueChanged(Sender: TObject);

procedure lcmblicnslevelKeyValueChanged(Sender: TObject);

procedure lcmbtestorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure lcmbtrainorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

private

Fb\_SelectOn: Boolean;

Fs\_CurPersonId: String;

Fs\_CurLicnsId: String;

procedure prip\_IniData;

procedure prip\_IniInterface;

procedure prip\_RefreshComListSource;

procedure prip\_RefreshPersonListSource;

procedure prip\_ListPsnTrain;

procedure prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmOrganMng);

procedure prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmOrganMng);

procedure prip\_PersonDialog(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmPersonMng);

procedure prip\_SelectPerson(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

procedure prip\_GetPsnTrainInfo;

procedure prip\_RefreshTrainTestInfo(const As\_ID: string = '');

procedure prip\_GetComInfo(Asa\_Ids: array of string);

procedure prip\_GetPersonInfo(Asa\_Ids: array of string);

function prif\_CheckBlank(Ac\_Fields: array of TField): Boolean;

function prif\_Validate: Boolean;

protected

Rb\_DataModified: Boolean;

property SelectOn: Boolean read Fb\_SelectOn;

public

procedure pubp\_Ini(const Ab\_SelectOn: Boolean; Ac\_Args: TStrings = nil);

property DataModified: Boolean read Rb\_DataModified;

end;

var

frmPsnTrainTest: TfrmPsnTrainTest;

implementation

uses

CSVCLUtils, udmData, CommonLib;

{$R \*.dfm}

{ TfrmPsnTrainTest }

function TfrmPsnTrainTest.prif\_CheckBlank(

Ac\_Fields: array of TField): Boolean;

var

li\_i: Integer;

begin

Result := True;

with psnTrainTest do

for li\_i := 0 to Length(Ac\_Fields) - 1 do

if Trim(VarToStr(Ac\_Fields[li\_i].Value)) = '' then

begin

Result := False;

Application.MessageBox(PChar(Ac\_Fields[li\_i].DisplayLabel +

'不能为空。'), PChar(Caption), MB\_OK + MB\_ICONWARNING);

Ac\_Fields[li\_i].FocusControl;

Exit;

end;

end;

function TfrmPsnTrainTest.prif\_Validate: Boolean;

begin

with psnTrainTest do //

Result := prif\_CheckBlank([FieldByName('person\_sysid'),

FieldByName('testorgan\_sysid'), FieldByName('licns\_kind'),

FieldByName('licns\_category'), FieldByName('licns\_prj'),

FieldByName('start\_date'), FieldByName('end\_date'),

FieldByName('train\_result'), FieldByName('theory\_result'),

FieldByName('operate\_result')]);

end;

procedure TfrmPsnTrainTest.prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmOrganMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectCompany(Ai\_Category, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mComd;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_GetComInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.organ\_baseinfo ' +

'WHERE sysid = ''%s'' UNION ' +

'SELECT sysid, name, 2 AS ix FROM device.organ\_baseinfo ' +

'WHERE sysid = ''%s,

[Asa\_Ids[0], Asa\_Ids[1]]);

Open;

if not mCom.IsEmpty then mCom.Delete;

mCom.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mCom.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mCom.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mCom.Post;

mComBak.LoadFromDataSet(mCom);

Close;

end;

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.prip\_GetPersonInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.person\_baseinfo ' +

'WHERE sysid = ''%s, [Asa\_Ids[0]]);

Open;

if not mPerson.IsEmpty then mPerson.Delete;

mPerson.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mPerson.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mPerson.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mPerson.Post;

mPersonBak.LoadFromDataSet(mPerson);

Close;

end;

prip\_RefreshPersonListSource;

end;

procedure TfrmPsnTrainTest.prip\_IniData;

begin

//性质

with licnskind do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 1';

Open;

end;

//类别

with licnscategory do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 2';

Open;

end;

with licnsprj do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 3';

Open;

end;

with licnslevel do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 4';

Open;

end;

with trainresult do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_trainresult';

Open;

end;

prip\_ListPsnTrain;

end;

procedure TfrmPsnTrainTest.prip\_IniInterface;

begin

end;

procedure TfrmPsnTrainTest.prip\_PersonDialog(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; Ac\_LookupComboBox: TDBLookupComboBoxEh;

Ac\_Requestor: TfrmPersonMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectPerson(Ai\_Kind, Ai\_Category, Ai\_Prj, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mPersond;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshComListSource;

begin

with lcmbtrainOrgan do

begin

ListSource := nil;

ListSource := mComd;

end;

with lcmbtestOrgan do

begin

ListSource := nil;

ListSource := mComd;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshPersonListSource;

begin

with lcmbName do

begin

ListSource := nil;

ListSource := mPersond;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmOrganMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Company: TfrmOrganMng;

begin

ls\_PrevId := '';

lc\_Company := Ac\_Requestor;

with lc\_Company, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

lc\_Arg.Add(IntToStr(Ai\_Category));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurOrganId) then

begin

li\_Index := mCom.FieldByName(As\_Field).Index;

mCom.Edit;

mCom.Fields[li\_Index].AsString := CurOrganId;

mCom.Fields[li\_Index + 1].AsString := CurOrganName;

mCom.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurOrganId;

end;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectPerson(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Person: TfrmPersonMng;

begin

ls\_PrevId := '';

lc\_Person := Ac\_Requestor;

with lc\_Person, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

// lc\_Arg.Add(IntToStr(Ai\_Kind));

// lc\_Arg.Add(IntToStr(Ai\_Category));

// lc\_Arg.Add(IntToStr(Ai\_Prj));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurPersonId) then

begin

li\_Index := mPerson.FieldByName(As\_Field).Index;

mPerson.Edit;

mPerson.Fields[li\_Index].AsString := CurPersonId;

mPerson.Fields[li\_Index + 1].AsString := CurPersonName;

mPerson.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurPersonId;

end;

end;

end;

// Ac\_Args[0] 当前系统id Ac\_Args[1] 当前系统id

procedure TfrmPsnTrainTest.pubp\_Ini(const Ab\_SelectOn: Boolean;

Ac\_Args: TStrings);

begin

ScrChild.SetSecurity(Self);

Fb\_SelectOn := Ab\_SelectOn;

if Ac\_Args <> nil then

with Ac\_Args do

begin

if Count > 0 then Fs\_CurPersonId := Strings[0];

if Count > 1 then Fs\_CurLicnsId := Strings[1];

end;

Gp\_ProgressHint('正在载入信息，请稍候...');

try

prip\_IniData;

finally

Gp\_ProgressHint;

end;

prip\_IniInterface;

if SelectOn then

begin

BorderIcons := BorderIcons - [BiMinimize];

Position := poScreenCenter;

end else begin

FormStyle := fsMDIChild;

if MAXIMIZEDBYDEFAULT then WindowState := wsMaximized else ClientTile(Self);

end;

end;

procedure TfrmPsnTrainTest.FormCreate(Sender: TObject);

begin

Rb\_DataModified := False;

Fs\_CurPersonId := '';

Fs\_CurLicnsId := '';

end;

procedure TfrmPsnTrainTest.lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_PersonDialog(0, 0, 0, lcmbName, TfrmPersonMng.Create(nil));

end;

procedure TfrmPsnTrainTest.prip\_ListPsnTrain;

begin

with List do

begin

if Active then Close;

SQL.Text := Format('device.usp\_gettraininfo ''%s,[Fs\_CurPersonId]);

Open;

if IsEmpty then prip\_GetPsnTrainInfo;

end;

end;

procedure TfrmPsnTrainTest.listAfterScroll(DataSet: TDataSet);

begin

prip\_GetPsnTrainInfo;

end;

procedure TfrmPsnTrainTest.prip\_GetPsnTrainInfo;

var

ls\_PsnTrainId: string;

begin

ls\_PsnTrainId := List.FieldByName('sysid').AsString;

with psnTrainTest do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.person\_test WHERE ' +

'sysid = ''%s, [ls\_PsnTrainId]);

Open;

if not IsEmpty then

begin

prip\_GetComInfo([VarToStr(FieldValues['testorgan\_sysid']),

VarToStr(FieldValues['trainorgan\_sysid'])]);

prip\_GetPersonInfo([VarToStr(FieldValues['person\_sysid'])]);

end;

end;

end;

procedure TfrmPsnTrainTest.lcmbLicnsKindKeyValueChanged(Sender: TObject);

var

ls\_LicnsKind: string;

begin

ls\_LicnsKind := VarToStr(lcmbLicnsKind.KeyValue);

with licnscategory do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'where code\_level = 2 and parent\_id = ''%s, [ls\_LicnsKind]);

Open;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterInsert(DataSet: TDataSet);

begin

DataSet.FieldByName('sysid').AsString := Gf\_GenSysId;

if Fb\_SelectOn then prip\_GetPersonInfo(Fs\_CurPersonId);

DataSet.FieldByName('person\_sysid').AsString := Fs\_CurPersonId;

elicnsprjname.Text := '';

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterPost(DataSet: TDataSet);

var ls\_PsnTrainID: string;

begin

mComBak.LoadFromDataSet(mCom);

Rb\_DataModified := True;

ls\_PsnTrainID:= DataSet.FieldByName('sysid').AsString;

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo(ls\_PsnTrainID);

finally

Gp\_ProgressHint;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestBeforePost(DataSet: TDataSet);

begin

if not prif\_Validate then Abort;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterCancel(DataSet: TDataSet);

begin

mCom.LoadFromDataSet(mComBak);

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.bbtnCloseClick(Sender: TObject);

begin

Close;

end;

procedure TfrmPsnTrainTest.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

if FormStyle = fsMDIChild then

begin

Action := caFree;

frmPsnTrainTest := nil;

end else Action := caHide;

end;

procedure TfrmPsnTrainTest.prip\_RefreshTrainTestInfo(const As\_ID: string);

begin

with List do

begin

DisableControls;

try

if Active then Close;

SQL.Text := Format('device.usp\_gettraininfo ''%s,[Fs\_CurPersonId]);

Open;

if not IsEmpty then Locate('sysid', As\_ID, []);

finally

EnableControls;

end;

end;

end;

procedure TfrmPsnTrainTest.NavigatorBeforeAction(Sender: TObject;

Button: TNavigateBtn);

begin

if (Button = nbRefresh) and (psnTrainTest.State = dsBrowse) then

begin

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo;

finally

Gp\_ProgressHint;

end;

Abort;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterDelete(DataSet: TDataSet);

begin

prip\_RefreshTrainTestInfo;

end;

procedure TfrmPsnTrainTest.bbtnEngageInsClick(Sender: TObject);

begin

psnTrainTest.Insert;

end;

procedure TfrmPsnTrainTest.bbtnEngageDelClick(Sender: TObject);

begin

with PsnTrainTest do if (not Active) or IsEmpty then

begin

Exit;

end;

if Application.MessageBox('警告，您确定要删除当前选择的记录吗?', '确认',

MB\_OKCANCEL + MB\_DEFBUTTON2 + MB\_ICONQUESTION) = IDCANCEL then

begin

Exit;

end;

psnTrainTest.Delete;

end;

procedure TfrmPsnTrainTest.lcmbLicnsCategoryKeyValueChanged(

Sender: TObject);

var

ls\_LicnsCategory: string;

begin

ls\_LicnsCategory := VarToStr(lcmbLicnsCategory.KeyValue);

with licnsprj do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'WHERE code\_level = 3 and parent\_id = ''%s, [ls\_LicnsCategory]);

Open;

end;

with psnTrainTest do

if VarToStr(lcmbLicnsCategory.KeyValue) = PLS\_WELDER then

begin

lcmblicnsprj.ReadOnly := True;

lcmblicnslevel.ReadOnly := True;

end

else

begin

lcmblicnsprj.ReadOnly := False;

lcmblicnslevel.ReadOnly := False;

end;

end;

procedure TfrmPsnTrainTest.lcmblicnsprjKeyValueChanged(Sender: TObject);

var

ls\_prjname, ls\_levelname : string;

ls\_Licnsprj: string;

begin

ls\_prjname := lcmblicnsprj.Text ;

ls\_levelname := lcmblicnslevel.Text ;

with psnTrainTest do

if State in [dsEdit, dsInsert] then

FieldByName('psnlicns\_prjname').AsString := ls\_prjname + ls\_levelname;

ls\_Licnsprj := VarToStr(lcmblicnsprj.KeyValue);

with licnslevel do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'WHERE code\_level = 4 and parent\_id = ''%s, [ls\_Licnsprj]);

Open;

end;

end;

procedure TfrmPsnTrainTest.lcmblicnslevelKeyValueChanged(Sender: TObject);

var

ls\_prjname, ls\_levelname : string;

begin

ls\_prjname := lcmblicnsprj.Text ;

ls\_levelname := lcmblicnslevel.Text ;

with psnTrainTest do

if State in [dsEdit, dsInsert] then

FieldByName('psnlicns\_prjname').AsString := ls\_prjname + ls\_levelname;

end;

procedure TfrmPsnTrainTest.lcmbtestorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_CompanyDialog(ASSES\_SIGN, lcmbtestOrgan, TfrmOrganMng.Create(nil));

end;

procedure TfrmPsnTrainTest.lcmbtrainorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_CompanyDialog(ASSES\_SIGN, lcmbtrainOrgan, TfrmOrganMng.Create(nil));

end;

end.

unit ufrmPsnTrainTest;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DBCtrlsEh, StdCtrls, Mask, DBLookupEh, Buttons, DBCtrls,

CSDBNavigator, ExtCtrls, CSRowCountPanel, Grids, DBGridEh, dxEditor,

dxExEdtr, dxEdLib, CSCustomdxDateEdit, CSdxDBDateEdit, dxCntner, dxDBELib,

DB, ADODB, CSADOQuery, dxmdaset, ufrmPersonMng, ufrmOrganMng,

CSScrCtrlChild, CSBarMDIChild, PrnDbgeh, CSDBEhGrdStore, CSGrdColCtrl,

CSEhGrdColCtrl, CSADOFilter, CSADOEhGrdFilter, CSDBBase, CSDBFind,

CSDBEhGrdFind;

type

TfrmPsnTrainTest = class(TForm)

scrbDetail: TScrollBox;

Label14: TLabel;

Label19: TLabel;

Label27: TLabel;

Label30: TLabel;

Label36: TLabel;

mRemark: TdxDBMemo;

dteStartDate: TCSdxDBDateEdit;

Splitter1: TSplitter;

grdTrainList: TDBGridEh;

CSRowCountPanel1: TCSRowCountPanel;

Panel7: TPanel;

Panel8: TPanel;

Navigator: TCSDBNavigator;

bbtnEngageIns: TBitBtn;

bbtnEngageDel: TBitBtn;

bbtnFind: TBitBtn;

bbtnFilter: TBitBtn;

bbtnColunm: TBitBtn;

Label1: TLabel;

dteEndDate: TCSdxDBDateEdit;

Label37: TLabel;

lcmbtestorgan: TDBLookupComboboxEh;

Label2: TLabel;

eLResult: TDBEditEh;

Label3: TLabel;

eOResult: TDBEditEh;

Label4: TLabel;

lcmbName: TDBLookupComboboxEh;

lcmbLicnsKind: TDBLookupComboboxEh;

lcmbLicnsCategory: TDBLookupComboboxEh;

lcmbResult: TDBLookupComboboxEh;

psnTrainTest: TCSADOQuery;

psnTrainTestd: TDataSource;

Panel1: TPanel;

Panel4: TPanel;

bbtnHelp: TBitBtn;

bbtnClose: TBitBtn;

licnskind: TCSADOQuery;

licnskindd: TDataSource;

licnscategory: TCSADOQuery;

licnscategoryd: TDataSource;

trainresult: TCSADOQuery;

trainresultd: TDataSource;

Temp: TCSADOQuery;

mComBak: TdxMemData;

mCom: TdxMemData;

mComd: TDataSource;

mPersonBak: TdxMemData;

mPerson: TdxMemData;

mPersond: TDataSource;

mPersonperson\_sysid: TStringField;

mPersonperson\_sysid\_name: TStringField;

mPersonBakperson\_sysid: TStringField;

mPersonBakperson\_sysid\_name: TStringField;

ScrChild: TCSScrCtrlChild;

CSBarMDIChild1: TCSBarMDIChild;

ListFind: TCSDBEhGrdFind;

ListFilter: TCSADOEhGrdFilter;

ListColCtrl: TCSEhGrdColCtrl;

ListEhGrdStore: TCSDBEhGrdStore;

SaveDialog: TSaveDialog;

ListPrint: TPrintDBGridEh;

list: TCSADOQuery;

listd: TDataSource;

navList: TCSDBNavigator;

Label7: TLabel;

lcmblicnsprj: TDBLookupComboboxEh;

Label8: TLabel;

lcmblicnslevel: TDBLookupComboboxEh;

listsysid: TStringField;

listlicns\_prj\_sysid: TStringField;

listtestorgan\_sysid: TStringField;

listtestorgan\_sysid\_name: TStringField;

listtrainorgan\_sysid: TStringField;

listtrainorgan\_sysid\_name: TStringField;

listperson\_sysid: TStringField;

listperson\_sysid\_name: TStringField;

listlicns\_kind: TStringField;

listlicns\_kind\_name: TStringField;

listlicns\_category: TStringField;

listlicns\_category\_name: TStringField;

listlicns\_prj: TStringField;

listlicns\_prj\_name: TStringField;

listlicns\_level: TStringField;

listlicns\_level\_name: TStringField;

liststart\_date: TStringField;

listend\_date: TStringField;

listtrain\_result: TStringField;

listtrain\_result\_name: TStringField;

listtheory\_result: TStringField;

listoperate\_result: TStringField;

listtrain\_content: TStringField;

licnsprj: TCSADOQuery;

licnsprjd: TDataSource;

licnslevel: TCSADOQuery;

licnsleveld: TDataSource;

Label16: TLabel;

elicnsprjname: TDBEditEh;

Label5: TLabel;

mcontent: TdxDBMemo;

Label6: TLabel;

lcmbtrainorgan: TDBLookupComboboxEh;

mComtrainorgan\_sysid: TStringField;

mComtrainorgan\_sysid\_name: TStringField;

mComtestorgan\_sysid: TStringField;

mComtestorgan\_sysid\_name: TStringField;

mComBaktrainorgan\_sysid: TStringField;

mComBaktrainorgan\_sysid\_name: TStringField;

mComBaktestorgan\_sysid: TStringField;

mComBaktestorgan\_sysid\_name: TStringField;

listpsnlicns\_prjname: TStringField;

procedure FormCreate(Sender: TObject);

procedure lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure listAfterScroll(DataSet: TDataSet);

procedure lcmbLicnsKindKeyValueChanged(Sender: TObject);

procedure psnTrainTestAfterInsert(DataSet: TDataSet);

procedure psnTrainTestAfterPost(DataSet: TDataSet);

procedure psnTrainTestBeforePost(DataSet: TDataSet);

procedure psnTrainTestAfterCancel(DataSet: TDataSet);

procedure bbtnCloseClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure NavigatorBeforeAction(Sender: TObject; Button: TNavigateBtn);

procedure psnTrainTestAfterDelete(DataSet: TDataSet);

procedure bbtnEngageInsClick(Sender: TObject);

procedure bbtnEngageDelClick(Sender: TObject);

procedure lcmbLicnsCategoryKeyValueChanged(Sender: TObject);

procedure lcmblicnsprjKeyValueChanged(Sender: TObject);

procedure lcmblicnslevelKeyValueChanged(Sender: TObject);

procedure lcmbtestorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure lcmbtrainorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

private

Fb\_SelectOn: Boolean;

Fs\_CurPersonId: String;

Fs\_CurLicnsId: String;

procedure prip\_IniData;

procedure prip\_IniInterface;

procedure prip\_RefreshComListSource;

procedure prip\_RefreshPersonListSource;

procedure prip\_ListPsnTrain;

procedure prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmOrganMng);

procedure prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmOrganMng);

procedure prip\_PersonDialog(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmPersonMng);

procedure prip\_SelectPerson(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

procedure prip\_GetPsnTrainInfo;

procedure prip\_RefreshTrainTestInfo(const As\_ID: string = '');

procedure prip\_GetComInfo(Asa\_Ids: array of string);

procedure prip\_GetPersonInfo(Asa\_Ids: array of string);

function prif\_CheckBlank(Ac\_Fields: array of TField): Boolean;

function prif\_Validate: Boolean;

protected

Rb\_DataModified: Boolean;

property SelectOn: Boolean read Fb\_SelectOn;

public

procedure pubp\_Ini(const Ab\_SelectOn: Boolean; Ac\_Args: TStrings = nil);

property DataModified: Boolean read Rb\_DataModified;

end;

var

frmPsnTrainTest: TfrmPsnTrainTest;

implementation

uses

CSVCLUtils, udmData, CommonLib;

{$R \*.dfm}

{ TfrmPsnTrainTest }

function TfrmPsnTrainTest.prif\_CheckBlank(

Ac\_Fields: array of TField): Boolean;

var

li\_i: Integer;

begin

Result := True;

with psnTrainTest do

for li\_i := 0 to Length(Ac\_Fields) - 1 do

if Trim(VarToStr(Ac\_Fields[li\_i].Value)) = '' then

begin

Result := False;

Application.MessageBox(PChar(Ac\_Fields[li\_i].DisplayLabel +

'不能为空。'), PChar(Caption), MB\_OK + MB\_ICONWARNING);

Ac\_Fields[li\_i].FocusControl;

Exit;

end;

end;

function TfrmPsnTrainTest.prif\_Validate: Boolean;

begin

with psnTrainTest do //

Result := prif\_CheckBlank([FieldByName('person\_sysid'),

FieldByName('testorgan\_sysid'), FieldByName('licns\_kind'),

FieldByName('licns\_category'), FieldByName('licns\_prj'),

FieldByName('start\_date'), FieldByName('end\_date'),

FieldByName('train\_result'), FieldByName('theory\_result'),

FieldByName('operate\_result')]);

end;

procedure TfrmPsnTrainTest.prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmOrganMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectCompany(Ai\_Category, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mComd;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_GetComInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.organ\_baseinfo ' +

'WHERE sysid = ''%s'' UNION ' +

'SELECT sysid, name, 2 AS ix FROM device.organ\_baseinfo ' +

'WHERE sysid = ''%s,

[Asa\_Ids[0], Asa\_Ids[1]]);

Open;

if not mCom.IsEmpty then mCom.Delete;

mCom.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mCom.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mCom.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mCom.Post;

mComBak.LoadFromDataSet(mCom);

Close;

end;

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.prip\_GetPersonInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.person\_baseinfo ' +

'WHERE sysid = ''%s, [Asa\_Ids[0]]);

Open;

if not mPerson.IsEmpty then mPerson.Delete;

mPerson.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mPerson.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mPerson.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mPerson.Post;

mPersonBak.LoadFromDataSet(mPerson);

Close;

end;

prip\_RefreshPersonListSource;

end;

procedure TfrmPsnTrainTest.prip\_IniData;

begin

//性质

with licnskind do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 1';

Open;

end;

//类别

with licnscategory do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 2';

Open;

end;

with licnsprj do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 3';

Open;

end;

with licnslevel do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 4';

Open;

end;

with trainresult do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_trainresult';

Open;

end;

prip\_ListPsnTrain;

end;

procedure TfrmPsnTrainTest.prip\_IniInterface;

begin

end;

procedure TfrmPsnTrainTest.prip\_PersonDialog(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; Ac\_LookupComboBox: TDBLookupComboBoxEh;

Ac\_Requestor: TfrmPersonMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectPerson(Ai\_Kind, Ai\_Category, Ai\_Prj, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mPersond;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshComListSource;

begin

with lcmbtrainOrgan do

begin

ListSource := nil;

ListSource := mComd;

end;

with lcmbtestOrgan do

begin

ListSource := nil;

ListSource := mComd;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshPersonListSource;

begin

with lcmbName do

begin

ListSource := nil;

ListSource := mPersond;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmOrganMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Company: TfrmOrganMng;

begin

ls\_PrevId := '';

lc\_Company := Ac\_Requestor;

with lc\_Company, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

lc\_Arg.Add(IntToStr(Ai\_Category));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurOrganId) then

begin

li\_Index := mCom.FieldByName(As\_Field).Index;

mCom.Edit;

mCom.Fields[li\_Index].AsString := CurOrganId;

mCom.Fields[li\_Index + 1].AsString := CurOrganName;

mCom.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurOrganId;

end;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectPerson(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Person: TfrmPersonMng;

begin

ls\_PrevId := '';

lc\_Person := Ac\_Requestor;

with lc\_Person, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

// lc\_Arg.Add(IntToStr(Ai\_Kind));

// lc\_Arg.Add(IntToStr(Ai\_Category));

// lc\_Arg.Add(IntToStr(Ai\_Prj));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurPersonId) then

begin

li\_Index := mPerson.FieldByName(As\_Field).Index;

mPerson.Edit;

mPerson.Fields[li\_Index].AsString := CurPersonId;

mPerson.Fields[li\_Index + 1].AsString := CurPersonName;

mPerson.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurPersonId;

end;

end;

end;

// Ac\_Args[0] 当前系统id Ac\_Args[1] 当前系统id

procedure TfrmPsnTrainTest.pubp\_Ini(const Ab\_SelectOn: Boolean;

Ac\_Args: TStrings);

begin

ScrChild.SetSecurity(Self);

Fb\_SelectOn := Ab\_SelectOn;

if Ac\_Args <> nil then

with Ac\_Args do

begin

if Count > 0 then Fs\_CurPersonId := Strings[0];

if Count > 1 then Fs\_CurLicnsId := Strings[1];

end;

Gp\_ProgressHint('正在载入信息，请稍候...');

try

prip\_IniData;

finally

Gp\_ProgressHint;

end;

prip\_IniInterface;

if SelectOn then

begin

BorderIcons := BorderIcons - [BiMinimize];

Position := poScreenCenter;

end else begin

FormStyle := fsMDIChild;

if MAXIMIZEDBYDEFAULT then WindowState := wsMaximized else ClientTile(Self);

end;

end;

procedure TfrmPsnTrainTest.FormCreate(Sender: TObject);

begin

Rb\_DataModified := False;

Fs\_CurPersonId := '';

Fs\_CurLicnsId := '';

end;

procedure TfrmPsnTrainTest.lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_PersonDialog(0, 0, 0, lcmbName, TfrmPersonMng.Create(nil));

end;

procedure TfrmPsnTrainTest.prip\_ListPsnTrain;

begin

with List do

begin

if Active then Close;

SQL.Text := Format('device.usp\_gettraininfo ''%s,[Fs\_CurPersonId]);

Open;

if IsEmpty then prip\_GetPsnTrainInfo;

end;

end;

procedure TfrmPsnTrainTest.listAfterScroll(DataSet: TDataSet);

begin

prip\_GetPsnTrainInfo;

end;

procedure TfrmPsnTrainTest.prip\_GetPsnTrainInfo;

var

ls\_PsnTrainId: string;

begin

ls\_PsnTrainId := List.FieldByName('sysid').AsString;

with psnTrainTest do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.person\_test WHERE ' +

'sysid = ''%s, [ls\_PsnTrainId]);

Open;

if not IsEmpty then

begin

prip\_GetComInfo([VarToStr(FieldValues['testorgan\_sysid']),

VarToStr(FieldValues['trainorgan\_sysid'])]);

prip\_GetPersonInfo([VarToStr(FieldValues['person\_sysid'])]);

end;

end;

end;

procedure TfrmPsnTrainTest.lcmbLicnsKindKeyValueChanged(Sender: TObject);

var

ls\_LicnsKind: string;

begin

ls\_LicnsKind := VarToStr(lcmbLicnsKind.KeyValue);

with licnscategory do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'where code\_level = 2 and parent\_id = ''%s, [ls\_LicnsKind]);

Open;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterInsert(DataSet: TDataSet);

begin

DataSet.FieldByName('sysid').AsString := Gf\_GenSysId;

if Fb\_SelectOn then prip\_GetPersonInfo(Fs\_CurPersonId);

DataSet.FieldByName('person\_sysid').AsString := Fs\_CurPersonId;

elicnsprjname.Text := '';

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterPost(DataSet: TDataSet);

var ls\_PsnTrainID: string;

begin

mComBak.LoadFromDataSet(mCom);

Rb\_DataModified := True;

ls\_PsnTrainID:= DataSet.FieldByName('sysid').AsString;

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo(ls\_PsnTrainID);

finally

Gp\_ProgressHint;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestBeforePost(DataSet: TDataSet);

begin

if not prif\_Validate then Abort;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterCancel(DataSet: TDataSet);

begin

mCom.LoadFromDataSet(mComBak);

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.bbtnCloseClick(Sender: TObject);

begin

Close;

end;

procedure TfrmPsnTrainTest.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

if FormStyle = fsMDIChild then

begin

Action := caFree;

frmPsnTrainTest := nil;

end else Action := caHide;

end;

procedure TfrmPsnTrainTest.prip\_RefreshTrainTestInfo(const As\_ID: string);

begin

with List do

begin

DisableControls;

try

if Active then Close;

SQL.Text := Format('device.usp\_gettraininfo ''%s,[Fs\_CurPersonId]);

Open;

if not IsEmpty then Locate('sysid', As\_ID, []);

finally

EnableControls;

end;

end;

end;

procedure TfrmPsnTrainTest.NavigatorBeforeAction(Sender: TObject;

Button: TNavigateBtn);

begin

if (Button = nbRefresh) and (psnTrainTest.State = dsBrowse) then

begin

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo;

finally

Gp\_ProgressHint;

end;

Abort;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterDelete(DataSet: TDataSet);

begin

prip\_RefreshTrainTestInfo;

end;

procedure TfrmPsnTrainTest.bbtnEngageInsClick(Sender: TObject);

begin

psnTrainTest.Insert;

end;

procedure TfrmPsnTrainTest.bbtnEngageDelClick(Sender: TObject);

begin

with PsnTrainTest do if (not Active) or IsEmpty then

begin

Exit;

end;

if Application.MessageBox('警告，您确定要删除当前选择的记录吗?', '确认',

MB\_OKCANCEL + MB\_DEFBUTTON2 + MB\_ICONQUESTION) = IDCANCEL then

begin

Exit;

end;

psnTrainTest.Delete;

end;

procedure TfrmPsnTrainTest.lcmbLicnsCategoryKeyValueChanged(

Sender: TObject);

var

ls\_LicnsCategory: string;

begin

ls\_LicnsCategory := VarToStr(lcmbLicnsCategory.KeyValue);

with licnsprj do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'WHERE code\_level = 3 and parent\_id = ''%s, [ls\_LicnsCategory]);

Open;

end;

with psnTrainTest do

if VarToStr(lcmbLicnsCategory.KeyValue) = PLS\_WELDER then

begin

lcmblicnsprj.ReadOnly := True;

lcmblicnslevel.ReadOnly := True;

end

else

begin

lcmblicnsprj.ReadOnly := False;

lcmblicnslevel.ReadOnly := False;

end;

end;

procedure TfrmPsnTrainTest.lcmblicnsprjKeyValueChanged(Sender: TObject);

var

ls\_prjname, ls\_levelname : string;

ls\_Licnsprj: string;

begin

ls\_prjname := lcmblicnsprj.Text ;

ls\_levelname := lcmblicnslevel.Text ;

with psnTrainTest do

if State in [dsEdit, dsInsert] then

FieldByName('psnlicns\_prjname').AsString := ls\_prjname + ls\_levelname;

ls\_Licnsprj := VarToStr(lcmblicnsprj.KeyValue);

with licnslevel do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'WHERE code\_level = 4 and parent\_id = ''%s, [ls\_Licnsprj]);

Open;

end;

end;

procedure TfrmPsnTrainTest.lcmblicnslevelKeyValueChanged(Sender: TObject);

var

ls\_prjname, ls\_levelname : string;

begin

ls\_prjname := lcmblicnsprj.Text ;

ls\_levelname := lcmblicnslevel.Text ;

with psnTrainTest do

if State in [dsEdit, dsInsert] then

FieldByName('psnlicns\_prjname').AsString := ls\_prjname + ls\_levelname;

end;

procedure TfrmPsnTrainTest.lcmbtestorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_CompanyDialog(ASSES\_SIGN, lcmbtestOrgan, TfrmOrganMng.Create(nil));

end;

procedure TfrmPsnTrainTest.lcmbtrainorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_CompanyDialog(ASSES\_SIGN, lcmbtrainOrgan, TfrmOrganMng.Create(nil));

end;

end.

unit ufrmPsnTrainTest;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DBCtrlsEh, StdCtrls, Mask, DBLookupEh, Buttons, DBCtrls,

CSDBNavigator, ExtCtrls, CSRowCountPanel, Grids, DBGridEh, dxEditor,

dxExEdtr, dxEdLib, CSCustomdxDateEdit, CSdxDBDateEdit, dxCntner, dxDBELib,

DB, ADODB, CSADOQuery, dxmdaset, ufrmPersonMng, ufrmOrganMng,

CSScrCtrlChild, CSBarMDIChild, PrnDbgeh, CSDBEhGrdStore, CSGrdColCtrl,

CSEhGrdColCtrl, CSADOFilter, CSADOEhGrdFilter, CSDBBase, CSDBFind,

CSDBEhGrdFind;

type

TfrmPsnTrainTest = class(TForm)

scrbDetail: TScrollBox;

Label14: TLabel;

Label19: TLabel;

Label27: TLabel;

Label30: TLabel;

Label36: TLabel;

mRemark: TdxDBMemo;

dteStartDate: TCSdxDBDateEdit;

Splitter1: TSplitter;

grdTrainList: TDBGridEh;

CSRowCountPanel1: TCSRowCountPanel;

Panel7: TPanel;

Panel8: TPanel;

Navigator: TCSDBNavigator;

bbtnEngageIns: TBitBtn;

bbtnEngageDel: TBitBtn;

bbtnFind: TBitBtn;

bbtnFilter: TBitBtn;

bbtnColunm: TBitBtn;

Label1: TLabel;

dteEndDate: TCSdxDBDateEdit;

Label37: TLabel;

lcmbtestorgan: TDBLookupComboboxEh;

Label2: TLabel;

eLResult: TDBEditEh;

Label3: TLabel;

eOResult: TDBEditEh;

Label4: TLabel;

lcmbName: TDBLookupComboboxEh;

lcmbLicnsKind: TDBLookupComboboxEh;

lcmbLicnsCategory: TDBLookupComboboxEh;

lcmbResult: TDBLookupComboboxEh;

psnTrainTest: TCSADOQuery;

psnTrainTestd: TDataSource;

Panel1: TPanel;

Panel4: TPanel;

bbtnHelp: TBitBtn;

bbtnClose: TBitBtn;

licnskind: TCSADOQuery;

licnskindd: TDataSource;

licnscategory: TCSADOQuery;

licnscategoryd: TDataSource;

trainresult: TCSADOQuery;

trainresultd: TDataSource;

Temp: TCSADOQuery;

mComBak: TdxMemData;

mCom: TdxMemData;

mComd: TDataSource;

mPersonBak: TdxMemData;

mPerson: TdxMemData;

mPersond: TDataSource;

mPersonperson\_sysid: TStringField;

mPersonperson\_sysid\_name: TStringField;

mPersonBakperson\_sysid: TStringField;

mPersonBakperson\_sysid\_name: TStringField;

ScrChild: TCSScrCtrlChild;

CSBarMDIChild1: TCSBarMDIChild;

ListFind: TCSDBEhGrdFind;

ListFilter: TCSADOEhGrdFilter;

ListColCtrl: TCSEhGrdColCtrl;

ListEhGrdStore: TCSDBEhGrdStore;

SaveDialog: TSaveDialog;

ListPrint: TPrintDBGridEh;

list: TCSADOQuery;

listd: TDataSource;

navList: TCSDBNavigator;

Label7: TLabel;

lcmblicnsprj: TDBLookupComboboxEh;

Label8: TLabel;

lcmblicnslevel: TDBLookupComboboxEh;

listsysid: TStringField;

listlicns\_prj\_sysid: TStringField;

listtestorgan\_sysid: TStringField;

listtestorgan\_sysid\_name: TStringField;

listtrainorgan\_sysid: TStringField;

listtrainorgan\_sysid\_name: TStringField;

listperson\_sysid: TStringField;

listperson\_sysid\_name: TStringField;

listlicns\_kind: TStringField;

listlicns\_kind\_name: TStringField;

listlicns\_category: TStringField;

listlicns\_category\_name: TStringField;

listlicns\_prj: TStringField;

listlicns\_prj\_name: TStringField;

listlicns\_level: TStringField;

listlicns\_level\_name: TStringField;

liststart\_date: TStringField;

listend\_date: TStringField;

listtrain\_result: TStringField;

listtrain\_result\_name: TStringField;

listtheory\_result: TStringField;

listoperate\_result: TStringField;

listtrain\_content: TStringField;

licnsprj: TCSADOQuery;

licnsprjd: TDataSource;

licnslevel: TCSADOQuery;

licnsleveld: TDataSource;

Label16: TLabel;

elicnsprjname: TDBEditEh;

Label5: TLabel;

mcontent: TdxDBMemo;

Label6: TLabel;

lcmbtrainorgan: TDBLookupComboboxEh;

mComtrainorgan\_sysid: TStringField;

mComtrainorgan\_sysid\_name: TStringField;

mComtestorgan\_sysid: TStringField;

mComtestorgan\_sysid\_name: TStringField;

mComBaktrainorgan\_sysid: TStringField;

mComBaktrainorgan\_sysid\_name: TStringField;

mComBaktestorgan\_sysid: TStringField;

mComBaktestorgan\_sysid\_name: TStringField;

listpsnlicns\_prjname: TStringField;

procedure FormCreate(Sender: TObject);

procedure lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure listAfterScroll(DataSet: TDataSet);

procedure lcmbLicnsKindKeyValueChanged(Sender: TObject);

procedure psnTrainTestAfterInsert(DataSet: TDataSet);

procedure psnTrainTestAfterPost(DataSet: TDataSet);

procedure psnTrainTestBeforePost(DataSet: TDataSet);

procedure psnTrainTestAfterCancel(DataSet: TDataSet);

procedure bbtnCloseClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure NavigatorBeforeAction(Sender: TObject; Button: TNavigateBtn);

procedure psnTrainTestAfterDelete(DataSet: TDataSet);

procedure bbtnEngageInsClick(Sender: TObject);

procedure bbtnEngageDelClick(Sender: TObject);

procedure lcmbLicnsCategoryKeyValueChanged(Sender: TObject);

procedure lcmblicnsprjKeyValueChanged(Sender: TObject);

procedure lcmblicnslevelKeyValueChanged(Sender: TObject);

procedure lcmbtestorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure lcmbtrainorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

private

Fb\_SelectOn: Boolean;

Fs\_CurPersonId: String;

Fs\_CurLicnsId: String;

procedure prip\_IniData;

procedure prip\_IniInterface;

procedure prip\_RefreshComListSource;

procedure prip\_RefreshPersonListSource;

procedure prip\_ListPsnTrain;

procedure prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmOrganMng);

procedure prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmOrganMng);

procedure prip\_PersonDialog(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmPersonMng);

procedure prip\_SelectPerson(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

procedure prip\_GetPsnTrainInfo;

procedure prip\_RefreshTrainTestInfo(const As\_ID: string = '');

procedure prip\_GetComInfo(Asa\_Ids: array of string);

procedure prip\_GetPersonInfo(Asa\_Ids: array of string);

function prif\_CheckBlank(Ac\_Fields: array of TField): Boolean;

function prif\_Validate: Boolean;

protected

Rb\_DataModified: Boolean;

property SelectOn: Boolean read Fb\_SelectOn;

public

procedure pubp\_Ini(const Ab\_SelectOn: Boolean; Ac\_Args: TStrings = nil);

property DataModified: Boolean read Rb\_DataModified;

end;

var

frmPsnTrainTest: TfrmPsnTrainTest;

implementation

uses

CSVCLUtils, udmData, CommonLib;

{$R \*.dfm}

{ TfrmPsnTrainTest }

function TfrmPsnTrainTest.prif\_CheckBlank(

Ac\_Fields: array of TField): Boolean;

var

li\_i: Integer;

begin

Result := True;

with psnTrainTest do

for li\_i := 0 to Length(Ac\_Fields) - 1 do

if Trim(VarToStr(Ac\_Fields[li\_i].Value)) = '' then

begin

Result := False;

Application.MessageBox(PChar(Ac\_Fields[li\_i].DisplayLabel +

'不能为空。'), PChar(Caption), MB\_OK + MB\_ICONWARNING);

Ac\_Fields[li\_i].FocusControl;

Exit;

end;

end;

function TfrmPsnTrainTest.prif\_Validate: Boolean;

begin

with psnTrainTest do //

Result := prif\_CheckBlank([FieldByName('person\_sysid'),

FieldByName('testorgan\_sysid'), FieldByName('licns\_kind'),

FieldByName('licns\_category'), FieldByName('licns\_prj'),

FieldByName('start\_date'), FieldByName('end\_date'),

FieldByName('train\_result'), FieldByName('theory\_result'),

FieldByName('operate\_result')]);

end;

procedure TfrmPsnTrainTest.prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmOrganMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectCompany(Ai\_Category, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mComd;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_GetComInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.organ\_baseinfo ' +

'WHERE sysid = ''%s'' UNION ' +

'SELECT sysid, name, 2 AS ix FROM device.organ\_baseinfo ' +

'WHERE sysid = ''%s,

[Asa\_Ids[0], Asa\_Ids[1]]);

Open;

if not mCom.IsEmpty then mCom.Delete;

mCom.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mCom.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mCom.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mCom.Post;

mComBak.LoadFromDataSet(mCom);

Close;

end;

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.prip\_GetPersonInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.person\_baseinfo ' +

'WHERE sysid = ''%s, [Asa\_Ids[0]]);

Open;

if not mPerson.IsEmpty then mPerson.Delete;

mPerson.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mPerson.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mPerson.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mPerson.Post;

mPersonBak.LoadFromDataSet(mPerson);

Close;

end;

prip\_RefreshPersonListSource;

end;

procedure TfrmPsnTrainTest.prip\_IniData;

begin

//性质

with licnskind do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 1';

Open;

end;

//类别

with licnscategory do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 2';

Open;

end;

with licnsprj do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 3';

Open;

end;

with licnslevel do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 4';

Open;

end;

with trainresult do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_trainresult';

Open;

end;

prip\_ListPsnTrain;

end;

procedure TfrmPsnTrainTest.prip\_IniInterface;

begin

end;

procedure TfrmPsnTrainTest.prip\_PersonDialog(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; Ac\_LookupComboBox: TDBLookupComboBoxEh;

Ac\_Requestor: TfrmPersonMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectPerson(Ai\_Kind, Ai\_Category, Ai\_Prj, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mPersond;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshComListSource;

begin

with lcmbtrainOrgan do

begin

ListSource := nil;

ListSource := mComd;

end;

with lcmbtestOrgan do

begin

ListSource := nil;

ListSource := mComd;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshPersonListSource;

begin

with lcmbName do

begin

ListSource := nil;

ListSource := mPersond;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmOrganMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Company: TfrmOrganMng;

begin

ls\_PrevId := '';

lc\_Company := Ac\_Requestor;

with lc\_Company, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

lc\_Arg.Add(IntToStr(Ai\_Category));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurOrganId) then

begin

li\_Index := mCom.FieldByName(As\_Field).Index;

mCom.Edit;

mCom.Fields[li\_Index].AsString := CurOrganId;

mCom.Fields[li\_Index + 1].AsString := CurOrganName;

mCom.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurOrganId;

end;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectPerson(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Person: TfrmPersonMng;

begin

ls\_PrevId := '';

lc\_Person := Ac\_Requestor;

with lc\_Person, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

// lc\_Arg.Add(IntToStr(Ai\_Kind));

// lc\_Arg.Add(IntToStr(Ai\_Category));

// lc\_Arg.Add(IntToStr(Ai\_Prj));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurPersonId) then

begin

li\_Index := mPerson.FieldByName(As\_Field).Index;

mPerson.Edit;

mPerson.Fields[li\_Index].AsString := CurPersonId;

mPerson.Fields[li\_Index + 1].AsString := CurPersonName;

mPerson.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurPersonId;

end;

end;

end;

// Ac\_Args[0] 当前系统id Ac\_Args[1] 当前系统id

procedure TfrmPsnTrainTest.pubp\_Ini(const Ab\_SelectOn: Boolean;

Ac\_Args: TStrings);

begin

ScrChild.SetSecurity(Self);

Fb\_SelectOn := Ab\_SelectOn;

if Ac\_Args <> nil then

with Ac\_Args do

begin

if Count > 0 then Fs\_CurPersonId := Strings[0];

if Count > 1 then Fs\_CurLicnsId := Strings[1];

end;

Gp\_ProgressHint('正在载入信息，请稍候...');

try

prip\_IniData;

finally

Gp\_ProgressHint;

end;

prip\_IniInterface;

if SelectOn then

begin

BorderIcons := BorderIcons - [BiMinimize];

Position := poScreenCenter;

end else begin

FormStyle := fsMDIChild;

if MAXIMIZEDBYDEFAULT then WindowState := wsMaximized else ClientTile(Self);

end;

end;

procedure TfrmPsnTrainTest.FormCreate(Sender: TObject);

begin

Rb\_DataModified := False;

Fs\_CurPersonId := '';

Fs\_CurLicnsId := '';

end;

procedure TfrmPsnTrainTest.lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_PersonDialog(0, 0, 0, lcmbName, TfrmPersonMng.Create(nil));

end;

procedure TfrmPsnTrainTest.prip\_ListPsnTrain;

begin

with List do

begin

if Active then Close;

SQL.Text := Format('device.usp\_gettraininfo ''%s,[Fs\_CurPersonId]);

Open;

if IsEmpty then prip\_GetPsnTrainInfo;

end;

end;

procedure TfrmPsnTrainTest.listAfterScroll(DataSet: TDataSet);

begin

prip\_GetPsnTrainInfo;

end;

procedure TfrmPsnTrainTest.prip\_GetPsnTrainInfo;

var

ls\_PsnTrainId: string;

begin

ls\_PsnTrainId := List.FieldByName('sysid').AsString;

with psnTrainTest do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.person\_test WHERE ' +

'sysid = ''%s, [ls\_PsnTrainId]);

Open;

if not IsEmpty then

begin

prip\_GetComInfo([VarToStr(FieldValues['testorgan\_sysid']),

VarToStr(FieldValues['trainorgan\_sysid'])]);

prip\_GetPersonInfo([VarToStr(FieldValues['person\_sysid'])]);

end;

end;

end;

procedure TfrmPsnTrainTest.lcmbLicnsKindKeyValueChanged(Sender: TObject);

var

ls\_LicnsKind: string;

begin

ls\_LicnsKind := VarToStr(lcmbLicnsKind.KeyValue);

with licnscategory do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'where code\_level = 2 and parent\_id = ''%s, [ls\_LicnsKind]);

Open;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterInsert(DataSet: TDataSet);

begin

DataSet.FieldByName('sysid').AsString := Gf\_GenSysId;

if Fb\_SelectOn then prip\_GetPersonInfo(Fs\_CurPersonId);

DataSet.FieldByName('person\_sysid').AsString := Fs\_CurPersonId;

elicnsprjname.Text := '';

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterPost(DataSet: TDataSet);

var ls\_PsnTrainID: string;

begin

mComBak.LoadFromDataSet(mCom);

Rb\_DataModified := True;

ls\_PsnTrainID:= DataSet.FieldByName('sysid').AsString;

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo(ls\_PsnTrainID);

finally

Gp\_ProgressHint;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestBeforePost(DataSet: TDataSet);

begin

if not prif\_Validate then Abort;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterCancel(DataSet: TDataSet);

begin

mCom.LoadFromDataSet(mComBak);

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.bbtnCloseClick(Sender: TObject);

begin

Close;

end;

procedure TfrmPsnTrainTest.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

if FormStyle = fsMDIChild then

begin

Action := caFree;

frmPsnTrainTest := nil;

end else Action := caHide;

end;

procedure TfrmPsnTrainTest.prip\_RefreshTrainTestInfo(const As\_ID: string);

begin

with List do

begin

DisableControls;

try

if Active then Close;

SQL.Text := Format('device.usp\_gettraininfo ''%s,[Fs\_CurPersonId]);

Open;

if not IsEmpty then Locate('sysid', As\_ID, []);

finally

EnableControls;

end;

end;

end;

procedure TfrmPsnTrainTest.NavigatorBeforeAction(Sender: TObject;

Button: TNavigateBtn);

begin

if (Button = nbRefresh) and (psnTrainTest.State = dsBrowse) then

begin

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo;

finally

Gp\_ProgressHint;

end;

Abort;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterDelete(DataSet: TDataSet);

begin

prip\_RefreshTrainTestInfo;

end;

procedure TfrmPsnTrainTest.bbtnEngageInsClick(Sender: TObject);

begin

psnTrainTest.Insert;

end;

procedure TfrmPsnTrainTest.bbtnEngageDelClick(Sender: TObject);

begin

with PsnTrainTest do if (not Active) or IsEmpty then

begin

Exit;

end;

if Application.MessageBox('警告，您确定要删除当前选择的记录吗?', '确认',

MB\_OKCANCEL + MB\_DEFBUTTON2 + MB\_ICONQUESTION) = IDCANCEL then

begin

Exit;

end;

psnTrainTest.Delete;

end;

procedure TfrmPsnTrainTest.lcmbLicnsCategoryKeyValueChanged(

Sender: TObject);

var

ls\_LicnsCategory: string;

begin

ls\_LicnsCategory := VarToStr(lcmbLicnsCategory.KeyValue);

with licnsprj do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'WHERE code\_level = 3 and parent\_id = ''%s, [ls\_LicnsCategory]);

Open;

end;

with psnTrainTest do

if VarToStr(lcmbLicnsCategory.KeyValue) = PLS\_WELDER then

begin

lcmblicnsprj.ReadOnly := True;

lcmblicnslevel.ReadOnly := True;

end

else

begin

lcmblicnsprj.ReadOnly := False;

lcmblicnslevel.ReadOnly := False;

end;

end;

procedure TfrmPsnTrainTest.lcmblicnsprjKeyValueChanged(Sender: TObject);

var

ls\_prjname, ls\_levelname : string;

ls\_Licnsprj: string;

begin

ls\_prjname := lcmblicnsprj.Text ;

ls\_levelname := lcmblicnslevel.Text ;

with psnTrainTest do

if State in [dsEdit, dsInsert] then

FieldByName('psnlicns\_prjname').AsString := ls\_prjname + ls\_levelname;

ls\_Licnsprj := VarToStr(lcmblicnsprj.KeyValue);

with licnslevel do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'WHERE code\_level = 4 and parent\_id = ''%s, [ls\_Licnsprj]);

Open;

end;

end;

procedure TfrmPsnTrainTest.lcmblicnslevelKeyValueChanged(Sender: TObject);

var

ls\_prjname, ls\_levelname : string;

begin

ls\_prjname := lcmblicnsprj.Text ;

ls\_levelname := lcmblicnslevel.Text ;

with psnTrainTest do

if State in [dsEdit, dsInsert] then

FieldByName('psnlicns\_prjname').AsString := ls\_prjname + ls\_levelname;

end;

procedure TfrmPsnTrainTest.lcmbtestorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_CompanyDialog(ASSES\_SIGN, lcmbtestOrgan, TfrmOrganMng.Create(nil));

end;

procedure TfrmPsnTrainTest.lcmbtrainorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_CompanyDialog(ASSES\_SIGN, lcmbtrainOrgan, TfrmOrganMng.Create(nil));

end;

end.

unit ufrmPsnTrainTest;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DBCtrlsEh, StdCtrls, Mask, DBLookupEh, Buttons, DBCtrls,

CSDBNavigator, ExtCtrls, CSRowCountPanel, Grids, DBGridEh, dxEditor,

dxExEdtr, dxEdLib, CSCustomdxDateEdit, CSdxDBDateEdit, dxCntner, dxDBELib,

DB, ADODB, CSADOQuery, dxmdaset, ufrmPersonMng, ufrmOrganMng,

CSScrCtrlChild, CSBarMDIChild, PrnDbgeh, CSDBEhGrdStore, CSGrdColCtrl,

CSEhGrdColCtrl, CSADOFilter, CSADOEhGrdFilter, CSDBBase, CSDBFind,

CSDBEhGrdFind;

type

TfrmPsnTrainTest = class(TForm)

scrbDetail: TScrollBox;

Label14: TLabel;

Label19: TLabel;

Label27: TLabel;

Label30: TLabel;

Label36: TLabel;

mRemark: TdxDBMemo;

dteStartDate: TCSdxDBDateEdit;

Splitter1: TSplitter;

grdTrainList: TDBGridEh;

CSRowCountPanel1: TCSRowCountPanel;

Panel7: TPanel;

Panel8: TPanel;

Navigator: TCSDBNavigator;

bbtnEngageIns: TBitBtn;

bbtnEngageDel: TBitBtn;

bbtnFind: TBitBtn;

bbtnFilter: TBitBtn;

bbtnColunm: TBitBtn;

Label1: TLabel;

dteEndDate: TCSdxDBDateEdit;

Label37: TLabel;

lcmbtestorgan: TDBLookupComboboxEh;

Label2: TLabel;

eLResult: TDBEditEh;

Label3: TLabel;

eOResult: TDBEditEh;

Label4: TLabel;

lcmbName: TDBLookupComboboxEh;

lcmbLicnsKind: TDBLookupComboboxEh;

lcmbLicnsCategory: TDBLookupComboboxEh;

lcmbResult: TDBLookupComboboxEh;

psnTrainTest: TCSADOQuery;

psnTrainTestd: TDataSource;

Panel1: TPanel;

Panel4: TPanel;

bbtnHelp: TBitBtn;

bbtnClose: TBitBtn;

licnskind: TCSADOQuery;

licnskindd: TDataSource;

licnscategory: TCSADOQuery;

licnscategoryd: TDataSource;

trainresult: TCSADOQuery;

trainresultd: TDataSource;

Temp: TCSADOQuery;

mComBak: TdxMemData;

mCom: TdxMemData;

mComd: TDataSource;

mPersonBak: TdxMemData;

mPerson: TdxMemData;

mPersond: TDataSource;

mPersonperson\_sysid: TStringField;

mPersonperson\_sysid\_name: TStringField;

mPersonBakperson\_sysid: TStringField;

mPersonBakperson\_sysid\_name: TStringField;

ScrChild: TCSScrCtrlChild;

CSBarMDIChild1: TCSBarMDIChild;

ListFind: TCSDBEhGrdFind;

ListFilter: TCSADOEhGrdFilter;

ListColCtrl: TCSEhGrdColCtrl;

ListEhGrdStore: TCSDBEhGrdStore;

SaveDialog: TSaveDialog;

ListPrint: TPrintDBGridEh;

list: TCSADOQuery;

listd: TDataSource;

navList: TCSDBNavigator;

Label7: TLabel;

lcmblicnsprj: TDBLookupComboboxEh;

Label8: TLabel;

lcmblicnslevel: TDBLookupComboboxEh;

listsysid: TStringField;

listlicns\_prj\_sysid: TStringField;

listtestorgan\_sysid: TStringField;

listtestorgan\_sysid\_name: TStringField;

listtrainorgan\_sysid: TStringField;

listtrainorgan\_sysid\_name: TStringField;

listperson\_sysid: TStringField;

listperson\_sysid\_name: TStringField;

listlicns\_kind: TStringField;

listlicns\_kind\_name: TStringField;

listlicns\_category: TStringField;

listlicns\_category\_name: TStringField;

listlicns\_prj: TStringField;

listlicns\_prj\_name: TStringField;

listlicns\_level: TStringField;

listlicns\_level\_name: TStringField;

liststart\_date: TStringField;

listend\_date: TStringField;

listtrain\_result: TStringField;

listtrain\_result\_name: TStringField;

listtheory\_result: TStringField;

listoperate\_result: TStringField;

listtrain\_content: TStringField;

licnsprj: TCSADOQuery;

licnsprjd: TDataSource;

licnslevel: TCSADOQuery;

licnsleveld: TDataSource;

Label16: TLabel;

elicnsprjname: TDBEditEh;

Label5: TLabel;

mcontent: TdxDBMemo;

Label6: TLabel;

lcmbtrainorgan: TDBLookupComboboxEh;

mComtrainorgan\_sysid: TStringField;

mComtrainorgan\_sysid\_name: TStringField;

mComtestorgan\_sysid: TStringField;

mComtestorgan\_sysid\_name: TStringField;

mComBaktrainorgan\_sysid: TStringField;

mComBaktrainorgan\_sysid\_name: TStringField;

mComBaktestorgan\_sysid: TStringField;

mComBaktestorgan\_sysid\_name: TStringField;

listpsnlicns\_prjname: TStringField;

procedure FormCreate(Sender: TObject);

procedure lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure listAfterScroll(DataSet: TDataSet);

procedure lcmbLicnsKindKeyValueChanged(Sender: TObject);

procedure psnTrainTestAfterInsert(DataSet: TDataSet);

procedure psnTrainTestAfterPost(DataSet: TDataSet);

procedure psnTrainTestBeforePost(DataSet: TDataSet);

procedure psnTrainTestAfterCancel(DataSet: TDataSet);

procedure bbtnCloseClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure NavigatorBeforeAction(Sender: TObject; Button: TNavigateBtn);

procedure psnTrainTestAfterDelete(DataSet: TDataSet);

procedure bbtnEngageInsClick(Sender: TObject);

procedure bbtnEngageDelClick(Sender: TObject);

procedure lcmbLicnsCategoryKeyValueChanged(Sender: TObject);

procedure lcmblicnsprjKeyValueChanged(Sender: TObject);

procedure lcmblicnslevelKeyValueChanged(Sender: TObject);

procedure lcmbtestorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure lcmbtrainorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

private

Fb\_SelectOn: Boolean;

Fs\_CurPersonId: String;

Fs\_CurLicnsId: String;

procedure prip\_IniData;

procedure prip\_IniInterface;

procedure prip\_RefreshComListSource;

procedure prip\_RefreshPersonListSource;

procedure prip\_ListPsnTrain;

procedure prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmOrganMng);

procedure prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmOrganMng);

procedure prip\_PersonDialog(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmPersonMng);

procedure prip\_SelectPerson(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

procedure prip\_GetPsnTrainInfo;

procedure prip\_RefreshTrainTestInfo(const As\_ID: string = '');

procedure prip\_GetComInfo(Asa\_Ids: array of string);

procedure prip\_GetPersonInfo(Asa\_Ids: array of string);

function prif\_CheckBlank(Ac\_Fields: array of TField): Boolean;

function prif\_Validate: Boolean;

protected

Rb\_DataModified: Boolean;

property SelectOn: Boolean read Fb\_SelectOn;

public

procedure pubp\_Ini(const Ab\_SelectOn: Boolean; Ac\_Args: TStrings = nil);

property DataModified: Boolean read Rb\_DataModified;

end;

var

frmPsnTrainTest: TfrmPsnTrainTest;

implementation

uses

CSVCLUtils, udmData, CommonLib;

{$R \*.dfm}

{ TfrmPsnTrainTest }

function TfrmPsnTrainTest.prif\_CheckBlank(

Ac\_Fields: array of TField): Boolean;

var

li\_i: Integer;

begin

Result := True;

with psnTrainTest do

for li\_i := 0 to Length(Ac\_Fields) - 1 do

if Trim(VarToStr(Ac\_Fields[li\_i].Value)) = '' then

begin

Result := False;

Application.MessageBox(PChar(Ac\_Fields[li\_i].DisplayLabel +

'不能为空。'), PChar(Caption), MB\_OK + MB\_ICONWARNING);

Ac\_Fields[li\_i].FocusControl;

Exit;

end;

end;

function TfrmPsnTrainTest.prif\_Validate: Boolean;

begin

with psnTrainTest do //

Result := prif\_CheckBlank([FieldByName('person\_sysid'),

FieldByName('testorgan\_sysid'), FieldByName('licns\_kind'),

FieldByName('licns\_category'), FieldByName('licns\_prj'),

FieldByName('start\_date'), FieldByName('end\_date'),

FieldByName('train\_result'), FieldByName('theory\_result'),

FieldByName('operate\_result')]);

end;

procedure TfrmPsnTrainTest.prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmOrganMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectCompany(Ai\_Category, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mComd;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_GetComInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.organ\_baseinfo ' +

'WHERE sysid = ''%s'' UNION ' +

'SELECT sysid, name, 2 AS ix FROM device.organ\_baseinfo ' +

'WHERE sysid = ''%s,

[Asa\_Ids[0], Asa\_Ids[1]]);

Open;

if not mCom.IsEmpty then mCom.Delete;

mCom.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mCom.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mCom.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mCom.Post;

mComBak.LoadFromDataSet(mCom);

Close;

end;

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.prip\_GetPersonInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.person\_baseinfo ' +

'WHERE sysid = ''%s, [Asa\_Ids[0]]);

Open;

if not mPerson.IsEmpty then mPerson.Delete;

mPerson.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mPerson.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mPerson.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mPerson.Post;

mPersonBak.LoadFromDataSet(mPerson);

Close;

end;

prip\_RefreshPersonListSource;

end;

procedure TfrmPsnTrainTest.prip\_IniData;

begin

//性质

with licnskind do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 1';

Open;

end;

//类别

with licnscategory do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 2';

Open;

end;

with licnsprj do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 3';

Open;

end;

with licnslevel do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where code\_level = 4';

Open;

end;

with trainresult do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_trainresult';

Open;

end;

prip\_ListPsnTrain;

end;

procedure TfrmPsnTrainTest.prip\_IniInterface;

begin

end;

procedure TfrmPsnTrainTest.prip\_PersonDialog(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; Ac\_LookupComboBox: TDBLookupComboBoxEh;

Ac\_Requestor: TfrmPersonMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectPerson(Ai\_Kind, Ai\_Category, Ai\_Prj, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mPersond;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshComListSource;

begin

with lcmbtrainOrgan do

begin

ListSource := nil;

ListSource := mComd;

end;

with lcmbtestOrgan do

begin

ListSource := nil;

ListSource := mComd;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshPersonListSource;

begin

with lcmbName do

begin

ListSource := nil;

ListSource := mPersond;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmOrganMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Company: TfrmOrganMng;

begin

ls\_PrevId := '';

lc\_Company := Ac\_Requestor;

with lc\_Company, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

lc\_Arg.Add(IntToStr(Ai\_Category));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurOrganId) then

begin

li\_Index := mCom.FieldByName(As\_Field).Index;

mCom.Edit;

mCom.Fields[li\_Index].AsString := CurOrganId;

mCom.Fields[li\_Index + 1].AsString := CurOrganName;

mCom.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurOrganId;

end;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectPerson(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Person: TfrmPersonMng;

begin

ls\_PrevId := '';

lc\_Person := Ac\_Requestor;

with lc\_Person, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

// lc\_Arg.Add(IntToStr(Ai\_Kind));

// lc\_Arg.Add(IntToStr(Ai\_Category));

// lc\_Arg.Add(IntToStr(Ai\_Prj));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurPersonId) then

begin

li\_Index := mPerson.FieldByName(As\_Field).Index;

mPerson.Edit;

mPerson.Fields[li\_Index].AsString := CurPersonId;

mPerson.Fields[li\_Index + 1].AsString := CurPersonName;

mPerson.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurPersonId;

end;

end;

end;

// Ac\_Args[0] 当前系统id Ac\_Args[1] 当前系统id

procedure TfrmPsnTrainTest.pubp\_Ini(const Ab\_SelectOn: Boolean;

Ac\_Args: TStrings);

begin

ScrChild.SetSecurity(Self);

Fb\_SelectOn := Ab\_SelectOn;

if Ac\_Args <> nil then

with Ac\_Args do

begin

if Count > 0 then Fs\_CurPersonId := Strings[0];

if Count > 1 then Fs\_CurLicnsId := Strings[1];

end;

Gp\_ProgressHint('正在载入信息，请稍候...');

try

prip\_IniData;

finally

Gp\_ProgressHint;

end;

prip\_IniInterface;

if SelectOn then

begin

BorderIcons := BorderIcons - [BiMinimize];

Position := poScreenCenter;

end else begin

FormStyle := fsMDIChild;

if MAXIMIZEDBYDEFAULT then WindowState := wsMaximized else ClientTile(Self);

end;

end;

procedure TfrmPsnTrainTest.FormCreate(Sender: TObject);

begin

Rb\_DataModified := False;

Fs\_CurPersonId := '';

Fs\_CurLicnsId := '';

end;

procedure TfrmPsnTrainTest.lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_PersonDialog(0, 0, 0, lcmbName, TfrmPersonMng.Create(nil));

end;

procedure TfrmPsnTrainTest.prip\_ListPsnTrain;

begin

with List do

begin

if Active then Close;

SQL.Text := Format('device.usp\_gettraininfo ''%s,[Fs\_CurPersonId]);

Open;

if IsEmpty then prip\_GetPsnTrainInfo;

end;

end;

procedure TfrmPsnTrainTest.listAfterScroll(DataSet: TDataSet);

begin

prip\_GetPsnTrainInfo;

end;

procedure TfrmPsnTrainTest.prip\_GetPsnTrainInfo;

var

ls\_PsnTrainId: string;

begin

ls\_PsnTrainId := List.FieldByName('sysid').AsString;

with psnTrainTest do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.person\_test WHERE ' +

'sysid = ''%s, [ls\_PsnTrainId]);

Open;

if not IsEmpty then

begin

prip\_GetComInfo([VarToStr(FieldValues['testorgan\_sysid']),

VarToStr(FieldValues['trainorgan\_sysid'])]);

prip\_GetPersonInfo([VarToStr(FieldValues['person\_sysid'])]);

end;

end;

end;

procedure TfrmPsnTrainTest.lcmbLicnsKindKeyValueChanged(Sender: TObject);

var

ls\_LicnsKind: string;

begin

ls\_LicnsKind := VarToStr(lcmbLicnsKind.KeyValue);

with licnscategory do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'where code\_level = 2 and parent\_id = ''%s, [ls\_LicnsKind]);

Open;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterInsert(DataSet: TDataSet);

begin

DataSet.FieldByName('sysid').AsString := Gf\_GenSysId;

if Fb\_SelectOn then prip\_GetPersonInfo(Fs\_CurPersonId);

DataSet.FieldByName('person\_sysid').AsString := Fs\_CurPersonId;

elicnsprjname.Text := '';

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterPost(DataSet: TDataSet);

var ls\_PsnTrainID: string;

begin

mComBak.LoadFromDataSet(mCom);

Rb\_DataModified := True;

ls\_PsnTrainID:= DataSet.FieldByName('sysid').AsString;

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo(ls\_PsnTrainID);

finally

Gp\_ProgressHint;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestBeforePost(DataSet: TDataSet);

begin

if not prif\_Validate then Abort;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterCancel(DataSet: TDataSet);

begin

mCom.LoadFromDataSet(mComBak);

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.bbtnCloseClick(Sender: TObject);

begin

Close;

end;

procedure TfrmPsnTrainTest.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

if FormStyle = fsMDIChild then

begin

Action := caFree;

frmPsnTrainTest := nil;

end else Action := caHide;

end;

procedure TfrmPsnTrainTest.prip\_RefreshTrainTestInfo(const As\_ID: string);

begin

with List do

begin

DisableControls;

try

if Active then Close;

SQL.Text := Format('device.usp\_gettraininfo ''%s,[Fs\_CurPersonId]);

Open;

if not IsEmpty then Locate('sysid', As\_ID, []);

finally

EnableControls;

end;

end;

end;

procedure TfrmPsnTrainTest.NavigatorBeforeAction(Sender: TObject;

Button: TNavigateBtn);

begin

if (Button = nbRefresh) and (psnTrainTest.State = dsBrowse) then

begin

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo;

finally

Gp\_ProgressHint;

end;

Abort;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterDelete(DataSet: TDataSet);

begin

prip\_RefreshTrainTestInfo;

end;

procedure TfrmPsnTrainTest.bbtnEngageInsClick(Sender: TObject);

begin

psnTrainTest.Insert;

end;

procedure TfrmPsnTrainTest.bbtnEngageDelClick(Sender: TObject);

begin

with PsnTrainTest do if (not Active) or IsEmpty then

begin

Exit;

end;

if Application.MessageBox('警告，您确定要删除当前选择的记录吗?', '确认',

MB\_OKCANCEL + MB\_DEFBUTTON2 + MB\_ICONQUESTION) = IDCANCEL then

begin

Exit;

end;

psnTrainTest.Delete;

end;

procedure TfrmPsnTrainTest.lcmbLicnsCategoryKeyValueChanged(

Sender: TObject);

var

ls\_LicnsCategory: string;

begin

ls\_LicnsCategory := VarToStr(lcmbLicnsCategory.KeyValue);

with licnsprj do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'WHERE code\_level = 3 and parent\_id = ''%s, [ls\_LicnsCategory]);

Open;

end;

with psnTrainTest do

if VarToStr(lcmbLicnsCategory.KeyValue) = PLS\_WELDER then

begin

lcmblicnsprj.ReadOnly := True;

lcmblicnslevel.ReadOnly := True;

end

else

begin

lcmblicnsprj.ReadOnly := False;

lcmblicnslevel.ReadOnly := False;

end;

end;

procedure TfrmPsnTrainTest.lcmblicnsprjKeyValueChanged(Sender: TObject);

var

ls\_prjname, ls\_levelname : string;

ls\_Licnsprj: string;

begin

ls\_prjname := lcmblicnsprj.Text ;

ls\_levelname := lcmblicnslevel.Text ;

with psnTrainTest do

if State in [dsEdit, dsInsert] then

FieldByName('psnlicns\_prjname').AsString := ls\_prjname + ls\_levelname;

ls\_Licnsprj := VarToStr(lcmblicnsprj.KeyValue);

with licnslevel do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'WHERE code\_level = 4 and parent\_id = ''%s, [ls\_Licnsprj]);

Open;

end;

end;

procedure TfrmPsnTrainTest.lcmblicnslevelKeyValueChanged(Sender: TObject);

var

ls\_prjname, ls\_levelname : string;

begin

ls\_prjname := lcmblicnsprj.Text ;

ls\_levelname := lcmblicnslevel.Text ;

with psnTrainTest do

if State in [dsEdit, dsInsert] then

FieldByName('psnlicns\_prjname').AsString := ls\_prjname + ls\_levelname;

end;

procedure TfrmPsnTrainTest.lcmbtestorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_CompanyDialog(ASSES\_SIGN, lcmbtestOrgan, TfrmOrganMng.Create(nil));

end;

procedure TfrmPsnTrainTest.lcmbtrainorganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_CompanyDialog(ASSES\_SIGN, lcmbtrainOrgan, TfrmOrganMng.Create(nil));

end;

end.

unit ufrmPsnTrainTest;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, DBCtrlsEh, StdCtrls, Mask, DBLookupEh, Buttons, DBCtrls,

CSDBNavigator, ExtCtrls, CSRowCountPanel, Grids, DBGridEh, dxEditor,

dxExEdtr, dxEdLib, CSCustomdxDateEdit, CSdxDBDateEdit, dxCntner, dxDBELib,

DB, ADODB, CSADOQuery, dxmdaset, ufrmPersonMng, ufrmComMng,

CSScrCtrlChild, CSBarMDIChild, PrnDbgeh, CSDBEhGrdStore, CSGrdColCtrl,

CSEhGrdColCtrl, CSADOFilter, CSADOEhGrdFilter, CSDBBase, CSDBFind,

CSDBEhGrdFind;

type

TfrmPsnTrainTest = class(TForm)

scrbDetail: TScrollBox;

Label14: TLabel;

Label19: TLabel;

Label27: TLabel;

Label30: TLabel;

Label36: TLabel;

mRemark: TdxDBMemo;

dteStartDate: TCSdxDBDateEdit;

Splitter1: TSplitter;

grdTrainList: TDBGridEh;

CSRowCountPanel1: TCSRowCountPanel;

Panel7: TPanel;

Panel8: TPanel;

Navigator: TCSDBNavigator;

bbtnEngageIns: TBitBtn;

bbtnEngageDel: TBitBtn;

bbtnFind: TBitBtn;

bbtnFilter: TBitBtn;

bbtnColunm: TBitBtn;

Label1: TLabel;

dteEndDate: TCSdxDBDateEdit;

Label37: TLabel;

lcmbOrgan: TDBLookupComboboxEh;

Label2: TLabel;

eLResult: TDBEditEh;

Label3: TLabel;

eOResult: TDBEditEh;

Label4: TLabel;

lcmbName: TDBLookupComboboxEh;

lcmbLicnsKind: TDBLookupComboboxEh;

lcmbLicnsCategory: TDBLookupComboboxEh;

lcmbResult: TDBLookupComboboxEh;

psnTrainTest: TCSADOQuery;

psnTrainTestd: TDataSource;

psnTrainTestsysid: TStringField;

psnTrainTestlicns\_sysid: TStringField;

psnTrainTestorgan\_sysid: TStringField;

psnTrainTestperson\_sysid: TStringField;

psnTrainTestlicns\_kind: TStringField;

psnTrainTestcategory: TStringField;

psnTrainTeststartdate: TStringField;

psnTrainTestenddate: TStringField;

psnTrainTesttrain\_result: TStringField;

psnTrainTesttheory\_result: TStringField;

psnTrainTestoperate\_result: TStringField;

psnTrainTesttrain\_content: TStringField;

psnTrainTestremark: TStringField;

Panel1: TPanel;

Panel4: TPanel;

bbtnHelp: TBitBtn;

bbtnClose: TBitBtn;

licnskind: TCSADOQuery;

licnskindd: TDataSource;

licnscategory: TCSADOQuery;

licnscategoryd: TDataSource;

trainresult: TCSADOQuery;

trainresultd: TDataSource;

Temp: TCSADOQuery;

mComBak: TdxMemData;

mCom: TdxMemData;

mComd: TDataSource;

mPersonBak: TdxMemData;

mPerson: TdxMemData;

mPersond: TDataSource;

mComorgan\_sysid: TStringField;

mComorgan\_sysid\_name: TStringField;

mComBakorgan\_sysid: TStringField;

mComBakorgan\_sysid\_name: TStringField;

mPersonperson\_sysid: TStringField;

mPersonperson\_sysid\_name: TStringField;

mPersonBakperson\_sysid: TStringField;

mPersonBakperson\_sysid\_name: TStringField;

ScrChild: TCSScrCtrlChild;

CSBarMDIChild1: TCSBarMDIChild;

ListFind: TCSDBEhGrdFind;

ListFilter: TCSADOEhGrdFilter;

ListColCtrl: TCSEhGrdColCtrl;

ListEhGrdStore: TCSDBEhGrdStore;

SaveDialog: TSaveDialog;

ListPrint: TPrintDBGridEh;

list: TCSADOQuery;

listd: TDataSource;

listsysid: TStringField;

listlicns\_sysid: TStringField;

listorgan\_sysid: TStringField;

listorgan\_sysid\_name: TStringField;

listperson\_sysid: TStringField;

listperson\_sysid\_name: TStringField;

listlicns\_kind: TStringField;

listlicns\_kind\_name: TStringField;

listcategory: TStringField;

listcategory\_name: TStringField;

liststartdate: TStringField;

listenddate: TStringField;

listtrain\_result: TStringField;

listtrain\_result\_name: TStringField;

listtheory\_result: TStringField;

listoperate\_result: TStringField;

listtrain\_content: TStringField;

navList: TCSDBNavigator;

procedure FormCreate(Sender: TObject);

procedure lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure lcmbOrganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

procedure listAfterScroll(DataSet: TDataSet);

procedure lcmbLicnsKindKeyValueChanged(Sender: TObject);

procedure psnTrainTestAfterInsert(DataSet: TDataSet);

procedure psnTrainTestAfterPost(DataSet: TDataSet);

procedure psnTrainTestBeforePost(DataSet: TDataSet);

procedure psnTrainTestAfterCancel(DataSet: TDataSet);

procedure bbtnCloseClick(Sender: TObject);

procedure FormClose(Sender: TObject; var Action: TCloseAction);

procedure NavigatorBeforeAction(Sender: TObject; Button: TNavigateBtn);

procedure psnTrainTestAfterDelete(DataSet: TDataSet);

procedure bbtnEngageInsClick(Sender: TObject);

procedure bbtnEngageDelClick(Sender: TObject);

private

Fb\_SelectOn: Boolean;

Fs\_CurPersonId: String;

Fs\_CurLicnsId: String;

procedure prip\_IniData;

procedure prip\_IniInterface;

procedure prip\_RefreshComListSource;

procedure prip\_RefreshPersonListSource;

procedure prip\_ListPsnTrain;

procedure prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmComMng);

procedure prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmComMng);

procedure prip\_PersonDialog(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmPersonMng);

procedure prip\_SelectPerson(const Ai\_Kind: Integer;

const Ai\_Category: Integer;const Ai\_Prj: Integer;

const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

procedure prip\_GetPsnTrainInfo;

procedure prip\_RefreshTrainTestInfo(const As\_ID: string = '');

procedure prip\_GetComInfo(Asa\_Ids: array of string);

procedure prip\_GetPersonInfo(Asa\_Ids: array of string);

function prif\_CheckBlank(Ac\_Fields: array of TField): Boolean;

function prif\_Validate: Boolean;

protected

Rb\_DataModified: Boolean;

property SelectOn: Boolean read Fb\_SelectOn;

public

procedure pubp\_Ini(const Ab\_SelectOn: Boolean; Ac\_Args: TStrings = nil);

property DataModified: Boolean read Rb\_DataModified;

end;

var

frmPsnTrainTest: TfrmPsnTrainTest;

implementation

uses

CSVCLUtils, udmData, CommonLib;

{$R \*.dfm}

{ TfrmPsnTrainTest }

function TfrmPsnTrainTest.prif\_CheckBlank(

Ac\_Fields: array of TField): Boolean;

var

li\_i: Integer;

begin

Result := True;

with psnTrainTest do

for li\_i := 0 to Length(Ac\_Fields) - 1 do

if Trim(VarToStr(Ac\_Fields[li\_i].Value)) = '' then

begin

Result := False;

Application.MessageBox(PChar(Ac\_Fields[li\_i].DisplayLabel +

'不能为空。'), PChar(Caption), MB\_OK + MB\_ICONWARNING);

Ac\_Fields[li\_i].FocusControl;

Exit;

end;

end;

function TfrmPsnTrainTest.prif\_Validate: Boolean;

begin

with psnTrainTest do //

Result := prif\_CheckBlank([FieldByName('person\_sysid'),

FieldByName('organ\_sysid'), FieldByName('licns\_kind'), FieldByName('category'),

FieldByName('startdate'), FieldByName('enddate'), FieldByName('train\_result'),

FieldByName('theory\_result'), FieldByName('operate\_result')]);

end;

procedure TfrmPsnTrainTest.prip\_CompanyDialog(const Ai\_Category: Integer;

Ac\_LookupComboBox: TDBLookupComboBoxEh; Ac\_Requestor: TfrmComMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectCompany(Ai\_Category, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mComd;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_GetComInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.organ\_baseinfo ' +

'WHERE sysid = ''%s,

[Asa\_Ids[0]]);

Open;

if not mCom.IsEmpty then mCom.Delete;

mCom.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mCom.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mCom.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mCom.Post;

mComBak.LoadFromDataSet(mCom);

Close;

end;

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.prip\_GetPersonInfo(Asa\_Ids: array of string);

var

li\_Index: Integer;

begin

with Temp do

begin

if Active then Close;

SQL.Text := Format(

'SELECT sysid, name, 1 AS ix FROM device.person\_baseinfo ' +

'WHERE sysid = ''%s, [Asa\_Ids[0]]);

Open;

if not mPerson.IsEmpty then mPerson.Delete;

mPerson.Insert;

while not Eof do

begin

li\_Index := FieldByName('ix').AsInteger;

mPerson.Fields[li\_Index \* 2 - 1].Value := FieldValues['sysid'];

mPerson.Fields[li\_Index \* 2].Value := FieldValues['name'];

Next;

end;

mPerson.Post;

mPersonBak.LoadFromDataSet(mPerson);

Close;

end;

prip\_RefreshPersonListSource;

end;

procedure TfrmPsnTrainTest.prip\_IniData;

begin

//性质

with licnskind do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where licns\_level = 1';

Open;

end;

//类别

with licnscategory do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_person\_licns where licns\_level = 2';

Open;

end;

with trainresult do

begin

if Active then Close;

SQL.Text := 'SELECT \* FROM device.code\_trainresult';

Open;

end;

prip\_ListPsnTrain;

// with psnTrainTest do

// begin

// if Active then Close;

// if Fb\_SelectOn then

// begin

// if Fs\_CurLicnsId = '' then

// SQL.Text := Format('SELECT \* FROM device.person\_assess WHERE ' +

// 'person\_sysid = ''%s, [Fs\_CurPersonId])

// else

// SQL.Text := Format('SELECT \* FROM device.person\_assess WHERE ' +

// 'person\_sysid = ''%s'' AND licns\_sysid = ''%s'' ', [Fs\_CurPersonId,

// Fs\_CurLicnsId]);

// end else SQL.Text := 'SELECT \* FROM device.person\_assess';

// Open;

// prip\_GetComInfo([VarToStr(FieldValues['organ\_sysid'])]);

// prip\_GetPersonInfo([VarToStr(FieldValues['person\_sysid'])]);

// end;

end;

procedure TfrmPsnTrainTest.prip\_IniInterface;

begin

end;

procedure TfrmPsnTrainTest.prip\_PersonDialog(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; Ac\_LookupComboBox: TDBLookupComboBoxEh;

Ac\_Requestor: TfrmPersonMng);

begin

with psnTrainTestd do if (not AutoEdit) or Ac\_LookupComboBox.ReadOnly then Exit;

with Ac\_LookupComboBox do

try

prip\_SelectPerson(Ai\_Kind, Ai\_Category, Ai\_Prj, DataField, Ac\_Requestor);

ListSource := nil;

ListSource := mPersond;

finally

if Ac\_Requestor <> nil then Ac\_Requestor.Free;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshComListSource;

begin

with lcmbOrgan do

begin

ListSource := nil;

ListSource := mComd;

end;

end;

procedure TfrmPsnTrainTest.prip\_RefreshPersonListSource;

begin

with lcmbName do

begin

ListSource := nil;

ListSource := mPersond;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectCompany(const Ai\_Category: Integer;

const As\_Field: string; Ac\_Requestor: TfrmComMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Company: TfrmComMng;

begin

ls\_PrevId := '';

lc\_Company := Ac\_Requestor;

with lc\_Company, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

lc\_Arg.Add(IntToStr(Ai\_Category));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurCompanyId) then

begin

li\_Index := mCom.FieldByName(As\_Field).Index;

mCom.Edit;

mCom.Fields[li\_Index].AsString := CurCompanyId;

mCom.Fields[li\_Index + 1].AsString := CurCompanyName;

mCom.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurCompanyId;

end;

end;

end;

procedure TfrmPsnTrainTest.prip\_SelectPerson(const Ai\_Kind, Ai\_Category,

Ai\_Prj: Integer; const As\_Field: string; Ac\_Requestor: TfrmPersonMng);

var

li\_Index: Integer;

ls\_PrevId: string;

lc\_Arg: TStrings;

lc\_Rc: TModalResult;

lc\_Person: TfrmPersonMng;

begin

ls\_PrevId := '';

lc\_Person := Ac\_Requestor;

with lc\_Person, psnTrainTest do

begin

if VarToStr(FieldValues[As\_Field]) <> '' then

ls\_PrevId := FieldByName(As\_Field).AsString;

lc\_Arg := TStringList.Create;

lc\_Arg.Add(ls\_PrevId);

// lc\_Arg.Add(IntToStr(Ai\_Kind));

// lc\_Arg.Add(IntToStr(Ai\_Category));

// lc\_Arg.Add(IntToStr(Ai\_Prj));

try

pubp\_ini(True, lc\_Arg);

lc\_Rc := ShowModal;

Self.Update;

finally

lc\_Arg.Free;

end;

if (lc\_Rc = mrOK) and (ls\_PrevId <> CurPersonId) then

begin

li\_Index := mPerson.FieldByName(As\_Field).Index;

mPerson.Edit;

mPerson.Fields[li\_Index].AsString := CurPersonId;

mPerson.Fields[li\_Index + 1].AsString := CurPersonName;

mPerson.Post;

if State = dsBrowse then if IsEmpty then Insert else Edit;

FieldByName(As\_Field).AsString := CurPersonId;

end;

end;

end;

// Ac\_Args[0] 当前系统id Ac\_Args[1] 当前系统id

procedure TfrmPsnTrainTest.pubp\_Ini(const Ab\_SelectOn: Boolean;

Ac\_Args: TStrings);

begin

ScrChild.SetSecurity(Self);

Fb\_SelectOn := Ab\_SelectOn;

if Ac\_Args <> nil then

with Ac\_Args do

begin

if Count > 0 then Fs\_CurPersonId := Strings[0];

if Count > 1 then Fs\_CurLicnsId := Strings[1];

end;

Gp\_ProgressHint('正在载入信息，请稍候...');

try

prip\_IniData;

finally

Gp\_ProgressHint;

end;

prip\_IniInterface;

if SelectOn then

begin

BorderIcons := BorderIcons - [BiMinimize];

Position := poScreenCenter;

end else begin

FormStyle := fsMDIChild;

if MAXIMIZEDBYDEFAULT then WindowState := wsMaximized else ClientTile(Self);

end;

end;

procedure TfrmPsnTrainTest.FormCreate(Sender: TObject);

begin

Rb\_DataModified := False;

Fs\_CurPersonId := '';

Fs\_CurLicnsId := '';

end;

procedure TfrmPsnTrainTest.lcmbNameEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_PersonDialog(0, 0, 0, lcmbName, TfrmPersonMng.Create(nil));

end;

procedure TfrmPsnTrainTest.lcmbOrganEditButtons0Click(Sender: TObject;

var Handled: Boolean);

begin

prip\_CompanyDialog(ASSES\_SIGN, lcmbOrgan, TfrmComMng.Create(nil));

end;

procedure TfrmPsnTrainTest.prip\_ListPsnTrain;

begin

with List do

begin

if Active then Close;

SQL.Text := 'device.usp\_gettraininfo';

Open;

if IsEmpty then prip\_GetPsnTrainInfo;

end;

end;

procedure TfrmPsnTrainTest.listAfterScroll(DataSet: TDataSet);

begin

prip\_GetPsnTrainInfo;

end;

procedure TfrmPsnTrainTest.prip\_GetPsnTrainInfo;

var

ls\_PsnTrainId: string;

begin

ls\_PsnTrainId := List.FieldByName('sysid').AsString;

with psnTrainTest do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.person\_assess WHERE ' +

'sysid = ''%s, [ls\_PsnTrainId]);

Open;

if not IsEmpty then

begin

prip\_GetComInfo([VarToStr(FieldValues['organ\_sysid'])]);

prip\_GetPersonInfo([VarToStr(FieldValues['person\_sysid'])]);

end;

end;

end;

procedure TfrmPsnTrainTest.lcmbLicnsKindKeyValueChanged(Sender: TObject);

var

ls\_LicnsKind: string;

begin

ls\_LicnsKind := VarToStr(lcmbLicnsKind.KeyValue);

with licnscategory do

begin

if Active then Close;

SQL.Text := Format('SELECT \* FROM device.code\_person\_licns ' +

'where licns\_level = 2 and parent\_id = ''%s, [ls\_LicnsKind]);

Open;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterInsert(DataSet: TDataSet);

begin

DataSet.FieldByName('sysid').AsString := Gf\_GenSysId;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterPost(DataSet: TDataSet);

var ls\_PsnTrainID: string;

begin

mComBak.LoadFromDataSet(mCom);

Rb\_DataModified := True;

ls\_PsnTrainID:= DataSet.FieldByName('sysid').AsString;

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo(ls\_PsnTrainID);

finally

Gp\_ProgressHint;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestBeforePost(DataSet: TDataSet);

begin

if not prif\_Validate then Abort;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterCancel(DataSet: TDataSet);

begin

mCom.LoadFromDataSet(mComBak);

prip\_RefreshComListSource;

end;

procedure TfrmPsnTrainTest.bbtnCloseClick(Sender: TObject);

begin

Close;

end;

procedure TfrmPsnTrainTest.FormClose(Sender: TObject;

var Action: TCloseAction);

begin

if FormStyle = fsMDIChild then

begin

Action := caFree;

frmPsnTrainTest := nil;

end else Action := caHide;

end;

procedure TfrmPsnTrainTest.prip\_RefreshTrainTestInfo(const As\_ID: string);

begin

with List do

begin

DisableControls;

try

if Active then Close;

SQL.Text := 'device.usp\_gettraininfo';

Open;

if not IsEmpty then Locate('sysid', As\_ID, []);

finally

EnableControls;

end;

end;

end;

procedure TfrmPsnTrainTest.NavigatorBeforeAction(Sender: TObject;

Button: TNavigateBtn);

begin

if (Button = nbRefresh) and (psnTrainTest.State = dsBrowse) then

begin

Gp\_ProgressHint('正在刷新，请稍候...');

try

prip\_RefreshTrainTestInfo;

finally

Gp\_ProgressHint;

end;

Abort;

end;

end;

procedure TfrmPsnTrainTest.psnTrainTestAfterDelete(DataSet: TDataSet);

begin

prip\_RefreshTrainTestInfo;

end;

procedure TfrmPsnTrainTest.bbtnEngageInsClick(Sender: TObject);

begin

psnTrainTest.Insert;

end;

procedure TfrmPsnTrainTest.bbtnEngageDelClick(Sender: TObject);

begin

with PsnTrainTest do if (not Active) or IsEmpty then

begin

Exit;

end;

if Application.MessageBox('警告，您确定要删除当前选择的记录吗?', '确认',

MB\_OKCANCEL + MB\_DEFBUTTON2 + MB\_ICONQUESTION) = IDCANCEL then

begin

Exit;

end;

psnTrainTest.Delete;

end;

end.