# Базовый фрейм TSymphonyPlugInBaseFrame

Оглавление

[Базовый фрейм TSymphonyPlugInBaseFrame 1](#_Toc464135574)

[Назначение фрейма 2](#_Toc464135575)

[Объявление класса 2](#_Toc464135576)

[Описание фрейма 2](#_Toc464135577)

[Public методы и свойства 2](#_Toc464135578)

[Создание фрейма (конструкторы фрейма) 2](#_Toc464135579)

[Уничтожение фрейма (деструктор фрейма) 4](#_Toc464135580)

[procedure ParentActivate; 4](#_Toc464135581)

[procedure ParentDeactivate; 4](#_Toc464135582)

[procedure SetCommand ; 5](#_Toc464135583)

[procedure TakeMessage ; 5](#_Toc464135584)

[class procedure SendMessage ; 6](#_Toc464135585)

[class function GetActiveFrame; 7](#_Toc464135586)

[class function PackageDBType; 7](#_Toc464135587)

[class function ExecuteCommonMethod; 7](#_Toc464135588)

[class function ExecuteCmdLine; 7](#_Toc464135589)

[class procedure SetOpenChildFormFunc; 8](#_Toc464135590)

[class procedure OpenChildForm; 8](#_Toc464135591)

[property Caption ; 9](#_Toc464135592)

[property ContextName ; 9](#_Toc464135593)

[property Folder; 9](#_Toc464135594)

[Protected методы 11](#_Toc464135595)

[procedure Init ; virtual ; 11](#_Toc464135596)

[procedure SetSession(ASession: TObject) ; virtual ; abstract ; 11](#_Toc464135597)

[procedure SetData(AData: TObject) ; overload ; virtual ; abstract ; 11](#_Toc464135598)

[procedure SetData(AData: ISymphonyPlugInCommand) ; overload ; virtual ; abstract ; 11](#_Toc464135599)

[procedure SetParent(AParent: TWinControl); override; 11](#_Toc464135600)

[procedure AddControlLeft(AStateStorage: TStateStorage; AControl: TControl) ; 11](#_Toc464135601)

[procedure AddControlTop(AStateStorage: TStateStorage; AControl: TControl) ; 11](#_Toc464135602)

[procedure AddControlPoint(AStateStorage: TStateStorage; AControl: TControl) ; 11](#_Toc464135603)

[procedure AddControlWidth(AStateStorage: TStateStorage; AControl: TControl) ; 11](#_Toc464135604)

[procedure AddControlHeight(AStateStorage: TStateStorage; AControl: TControl) ; 11](#_Toc464135605)

[procedure AddControlSize(AStateStorage: TStateStorage; AControl: TControl) ; 11](#_Toc464135606)

[procedure AddControlVisible(AStateStorage: TStateStorage; AControl: TControl) ; 11](#_Toc464135607)

[procedure AddControl(AStateStorage: TStateStorage; AControl: TControl) ; 11](#_Toc464135608)

[procedure AddControlsToStorage(AStateStorage: TStateStorage) ; virtual ; 11](#_Toc464135609)

[procedure AddParamsToStorage(AStateStorage: TStateStorage) ; virtual ; 11](#_Toc464135610)

[procedure ApplyParamsToStorage(AStateStorage: TStateStorage) ; virtual ; 11](#_Toc464135611)

[class function DoExecuteCommonMethod(MethodName: String; Params: TObject = nil): Boolean ; virtual ; 11](#_Toc464135612)

[procedure Print(Params: TObject = nil) ; virtual ; 11](#_Toc464135613)

[procedure ExportToExcel(Params: TObject = nil) ; virtual ; 11](#_Toc464135614)

## Назначение фрейма

Фрейм TSymphonyPlugInBaseFrame предназначен для построения пользовательского интерфейса, для задач реализуемых плагином. Объекты данного класса и их потомки автоматически размещаются на дочерних окнах приложения SymMng.exe и содержат ряд методов и свойств, позволяющих взаимодействовать с окнами и другими объектами приложения SymMng.exe.

## Объявление класса

Пакет: smplBaseFrame.bpl

Модуль: SymphonyPlugIn.BaseFrame

TSymphonyPlugInBaseFrame = **class**(TFrame)

## Описание фрейма

Фрейм TSymphonyPlugInBaseFrame редко используется для построения пользовательского интерфейса плагина. Данный фрейм формирует протоколы загрузки и взаимодействия с другими объектами приложения. Фрейм класса TSymphonyPlugInBaseFrame может быть использован для построения интерфейса программы не связанного с базами данных. При использовании баз данных, как правило, используются потомки фрейма TSymphonyPlugInBaseFrame, в которых учитываются специфика доступа к данным и специфика используемых компонентов (TSymphonyPlugInORABaseFrame – для доступа к БД Oracle, TSymphonyPlugInMSSQLBaseFrame – для доступа к БД Microsoft SQL Server и т.д.).

## Public методы и свойства

### Создание фрейма (конструкторы фрейма)

Класс фрейма TSymphonyPlugInBaseFrame описывает три конструктора:

**constructor** Create(AOwner: TComponent); **override**;

**constructor** CreatePlugIn(AOwner: TComponent; ASession: TObject; AData: TObject); **overload ; virtual ;**

**constructor** CreatePlugIn(AOwner: TComponent; ASession: TObject; AData: ISymphonyPlugInCommand); **overload ; virtual ;**

Конструктор **Create** наследуется от родительского класса **TFrame** и предназначен для работы под средой разработки Delphi. Данный конструктор проверяет свойство **ComponentState**, и если оно равно **csDesigning**, то вызывается конструктор родительского класса. В RunTime режиме (**ComponentState <> csDesigning**) конструктор возбуждает исключение и фрейм не создается.

Конструкторы **CreatePlugIn** предназначены для создания объекта фрейма по время выполнения программы и различаются только типом параметра **AData**. Конструкторам **CreatePlugIn** передаются следующие параметры:

* **AOwner: TComponent** – владелец фрейма, передается конструктору родительского класса.
* **ASession: TObject** – сессия соединения с базой данных. Тип сессии зависит от используемой базы данных и компонентов доступа к ней. Поэтому в конструктор базового фрейма **TSymphonyPlugInBaseFrame** передается сессия типа **TObject** (любого типа)
* **AData: ISymphonyPlugInCommand** или **AData: TObject** – дополнительные данные для формирования фрейма. В первом случае используется интерфейс **ISymphonyPlugInCommand**, объявленный в модуле **SymphonyPlugIn.ParamInterface**, который позволяет формировать команду со списком параметров и конвертировать её строку с XML-данными. Во втором случае в качестве данных можно передать любой объект, при этом потомок **TSymphonyPlugInBaseFrame** должен самостоятельно определить тип переданных данных и правильно интерпретировать их.

Конструкторы **CreatePlugIn** выполняют следующие шаги:

1. Вызывают конструктор Create родительского класса **TFrame** с переданным параметром **AOwner: TComponent**
2. Формируют уникальное имя фрейма. Так как на количество фреймов заданного класса в системе ограничений нет, то фрейм должен иметь уникальное имя, которое формируется из имени фрейма заданного при разработке и порядкового номера создаваемого фрейма в системе.
3. Вызывается виртуальная процедура **Init**, предназначенная для инициализации фрейма (создание объектов, установка начальных значений и значений по умолчанию и т.д.) Процедура позволяет избежать переопределения конструкторов **CreatePlugIn** в дочерних классах и заменить отсутствующее событие **OnCreate**
4. Вызывается абстрактная процедура **SetSession**. Процедура предназначена для подключения фрейма к переданному в конструктор соединению с базой данных. Так как тип базы данных и тип сессии не определены, то дочерние фреймы должны переопределить процедуру **SetSession**, проверить тип соединения и подключить все необходимые компоненты к базе данных.
5. Вызывается одна из абстрактных процедур **SetData**. Конкретная процедура определяется типом переданного параметра **AData**. Дочерние классы должны переопределить данную процедуру и произвести настройку в соответствии с переданными данными.
6. Добавляет себя в список созданных в рамках запущенной системы фреймов.

### Уничтожение фрейма (деструктор фрейма)

Деструктор фрейма сохраняет пользовательские данные в файл (процедура SaveState), Удаляет себя из списка созданных фреймов и вызывает деструктор родительского класса TFrame

### procedure ParentActivate;

**Область видимости:** **public**

**Объявление:**   
**procedure ParentActivate(Sender: TObject);**

**Параметры:**

**Sender: TObject** - ссылка на форму, на которой расположен фрейм.

**Описание:** обработчик события **OnActivate** формы, на которой расположен фрейм. Данный обработчик назначается окну автоматически, при размещении фрейма. Ранее назначенный обработчик события сохраняется и вызывается в рамках данной процедуры.

Данная процедура устанавливает текущий фрейм, как активный фрейм в списке созданных фреймов в системе (аналог свойств **TForm.ActiveControl** или **TForm.ActiveMDIChild**). В момент, когда дочернее окно системы становится активным, вызывается обработчик события **OnActivate**. Если на форме имеется фрейм, унаследованный от класса **TSymphonyPlugInBaseFrame**, то вызывается его обработчик **ParentActivate**, который сообщает системе, что данный фрейм является активным.

Назначение данного обработчика и его вызов выполняет система автоматически и вызываться самостоятельно программистом не должен!

### procedure ParentDeactivate;

**Область видимости:** **public**

**Объявление:** **procedure ParentDeactivate (Sender: TObject);**

**Параметры:**

**Sender: TObject** - ссылка на форму, на которой расположен фрейм.

**Описание:** обработчик события **OnDeactivate** формы, на которой расположен фрейм. Данный обработчик назначается окну автоматически, при размещении фрейма. Ранее назначенный обработчик события сохраняется и вызывается в рамках данной процедуры.

Данная процедура помечает текущий фрейм как не активный фрейм в списке созданных фреймов в системе (аналог свойств **TForm.ActiveControl** или **TForm.ActiveMDIChild**). В момент, когда дочернее окно системы становится неактивным (закрывается или активным становится другое окно), вызывается обработчик события **OnDeactivate**. Если на форме имеется фрейм, унаследованный от класса **TSymphonyPlugInBaseFrame**, то вызывается его обработчик **ParentDeactivate**, который сообщает системе, что данный фрейм больше не является активным.

Назначение данного обработчика и его вызов выполняет система автоматически и вызываться самостоятельно программистом не должен!

### procedure SetCommand ;

**Область видимости:** **public**

**Объявление:**   
**procedure SetCommand(ACommand: ISymphonyPlugInCommand) ; virtual ;**

**Параметры:**

**ACommand: ISymphonyPlugInCommand** - интерфейс, объявленный в модуле **SymphonyPlugIn.ParamInterface**, содержит команду с набором параметров.

**Описание:** процедура должна выполнить какие либо действия в ответ на переданную команду. Фреймы, класс которых будет наследоваться от **TSymphonyPlugInBaseFrame,** должны будут (при необходимости) переопределить эту процедуру. В переопределенных методах нужно будет писать реакцию на переданную команду. Например, один и тот же фрейм может создать новую запись, или отредактировать существующую, или открыть в режиме только чтения. Для этого можно передать команду со значениями свойств Command: “new”, “edit”, “readonly” или команду с именем “show”, а в параметре команды с именем Mode одно из трех значений: “new”, “edit”, “readonly”. На состав команды не накладывается каких либо ограничений. Фрейм должен проанализировать данные команды и выполнить соответствующую настройку фрейма или любые друге действия.

### procedure TakeMessage ;

**Область видимости:** **public**

**Объявление:**   
**procedure TakeMessage(Message: ISymphonyPlugInMessage) ; virtual ;**

**Параметры:**

**Message: ISymphonyPlugInMessage** - интерфейс, объявленный в модуле **SymphonyPlugIn.MessageInterface**, содержит сообщение с набором параметров.

**Описание:** Созданные в рамках запущенной системы фреймы наследники **TSymphonyPlugInBaseFrame** могут обмениваться друг с другом сообщениями. Данная процедура ответственна за прием сообщения. Фреймы, класс которых наследуется от **TSymphonyPlugInBaseFrame**, могут переопределить данный метод, где будет проанализировано содержимое переданного параметра и по результатам анализа выполнены некоторые действия. Например, получив сообщение с таким содержимым (обновилась запись в таблице sNomen с ikodnomen = 348725):

**Message. Domain = ‘svm’** – любая строка для фильтрации сообщений (наши / не наши)

**Message. Event = ‘Change DataSet’** – любая строка описывающая событие

**Message.ParamValue[‘DataSet’] = ‘sNomen’**

**Message.ParamValue[‘Operation’] = ‘UPDATE’**

**Message.ParamValue[‘iKodNomen] = ‘348725’**

Фрейм может обновить все наборы данных, где используется таблица sNomen, или запротоколировать факт изменения данных в таблице, или отправить сообщение другим фреймам о возможной цепной реакции с документами использующих запись из таблицы sNomen и т.д.

### class procedure SendMessage ;

**Область видимости:** **public**

**Объявление:**

**class procedure SendMessage(Message: ISymphonyPlugInMessage) ; overload ;**

**class procedure SendMessage(FrameClass: TSymphonyPlugInBaseFrameClass ; Message: ISymphonyPlugInMessage) ; overload ;**

**class procedure SendMessage(FrameClass: String ; Message: ISymphonyPlugInMessage) ; overload ;**

**class procedure SendMessage(Frame: TSymphonyPlugInBaseFrame ; Message: ISymphonyPlugInMessage) ; overload ;**

**Параметры:**

**Message: ISymphonyPlugInMessage** - интерфейс, объявленный в модуле **SymphonyPlugIn.MessageInterface**, содержит отсылаемое сообщение с набором параметров.

**FrameClass: TSymphonyPlugInBaseFrameClass –** класс, фреймам которого будут разосланы сообщения

**FrameClass: String–** имя класса, фреймам которого будут разосланы сообщения

**Frame: TSymphonyPlugInBaseFrame –** фрейм, которому будут разосланы сообщения.

**Описание:**

Созданные в рамках запущенной системы фреймы наследники **TSymphonyPlugInBaseFrame** могут обмениваться друг с другом сообщениями. Данная процедура ответственна за отправку сообщения.

Первая процедура рассылает сообщение всем созданным фреймам, классы которых наследуются от **TSymphonyPlugInBaseFrame** . Вторая и третья процедуры отправляют сообщение созданным фреймам определенного класса. Например, рассылка сообщения всем отчётам. Четвертая процедура отсылает сообщение конкретному фрейму.

В коде плагина процедура может создать интерфейс **ISymphonyPlugInMessage**, наполнить его данными и разослать его, используя одну из процедур **SendMessage**. Результат обработки сообщения можно узнать, проанализировав свойство **ISymphonyPlugInMessage. Result: Variant,** куда фреймы, обработавшие сообщение могут записать какие либо данные. В случае, если сообщение обрабатывает несколько фреймов, то в свойстве **Result** будет храниться значение от последнего фрейма.

### class function GetActiveFrame;

**Область видимости:** **public**

**Объявление:**   
**class function GetActiveFrame: TSymphonyPlugInBaseFrame ;**

**Описание:** функция возвращает активный фрейм, который расположен на активном дочернем окне системы (см. procedure ParentActivate;).

### class function PackageDBType;

**Область видимости:** **public**

**Объявление:** **class function PackageDBType: String ; virtual ; abstract ;**

**Описание:** абстрактная функция, которая должна быть переопределена в класса наследниках и должна возвращать строку, описывающую тип используемой фреймом базы данных (ORA, MSSQL, IB и т.д.)

### class function ExecuteCommonMethod;

**Область видимости:** **public**

**Объявление:**   
**class function ExecuteCommonMethod(MethodName: String; Params: TObject = nil): Boolean ;**

**Параметры:**

**MethodName: String** - имя метода, который необходимо выполнить

**Params: TObject –** параметры, которые будут переданы в выполняемый метод.

**Описание:** Данная функция анализирует параметр MethodName и вызывает один из предопределенных методов активного фрейма. В текущей версии фрейма, функция работает с двумя методами:

|  |  |
| --- | --- |
| Значение параметра MethodName | Вызываемая функция |
| exportexcel |  |
| print |  |

Функция возвращает False, если имя метода не совпадает ни с одним из поддерживаемых значений или активного фрейма нет или возвращает значение, которое вернет вызываемая функция.

### class function ExecuteCmdLine;

**Область видимости:** **public**

**Объявление:**   
**class function ExecuteCmdLine(CmdLine: String; Async: Boolean = True): Boolean ;**

**Параметры:**

**CmdLine: String** - кодированная строка, определяющая список плагинов, с акциями и параметрами акций, которые будут запущены функцией.

**Async: Boolean – синхронное или** асинхронное выполнение строки CmdLine.

**Описание:**

Функция передает главному окну на выполнение строку и сразу же возвращает управление в вызывающий код, если параметр ASync = True. В случае, если ASync = False, тогда возврат управления в вызывающий код произойдет только после окончания обработки строки CmdLine главным окном приложения.

Функция возвращает True, если выполнена успешно (строка передана на управление при ASync = True или строка выполнена успешно при ASync = False) и False, если при ASync = False строка CmdLine по каким либо причинам не выполнена.

Функция используется для запуска плагинов из другого плагина. Например, различные плагины в системе могут содержать фрейм, содержащий справочник номенклатуры. При изменении структуры справочника, придется корректировать все плагины. С другой стороны можно сделать плагин, содержащий фрейм со справочником номенклатуры, а в других плагинах вызывать этот с помощью функции **ExecuteCmdLine**.

Кодирование и раскодирование строки CmdLine осуществляется с помощью объектов класса **TCMDPlugInCommandLine**, который объявлен в модуле **smplCmdParser.MainUnit** (пакет **smplCmdParser**)

### class procedure SetOpenChildFormFunc;

**Область видимости:** **public**

**Объявление:**   
**class procedure SetOpenChildFormFunc(AFunc: TOpenChildFormFunc) ;**

**Параметры:**

**AFunc: TOpenChildFormFunc** - указатель на функцию открытия дочернего окна.

**Описание:**

Функция присваивает переменной модуля указатель на функцию главного окна приложения, которая будет создавать дочерние MDI-окна. Функция вызывается автоматически и программистом вызываться не должна!

### class procedure OpenChildForm;

**Область видимости:** **public**

**Объявление:**   
**class procedure OpenChildForm(Command: ISymphonyPlugInCommand; AUnique: Boolean = True; AID: Integer = -1) ;**

**Параметры:**

**Command: ISymphonyPlugInCommand** - команда передаваемая фрейму при его создании

**AUnique: Boolean – определяет уникальность окна с заданным фреймом и заданным идентификатором AID**.

**AID: Integer – идентификатор записи в базе данных/ Совместно с параметром AUnique, определяет будет создано новое окно или активизировано одно из уже существующих.**

**Описание:**

Процедура вызывается системой автоматически при создании фрейма и программистом вызываться не должна. Процедура используется в процессе загрузки плагина. Сначала система загружает пакет и извлекает из него класс фрейма, который экспортирует плагин. Затем вызывается данная функция полученного класса. Функция создает дочернее MDI окно, создает фрейм и размещает его на созданном окне. Если параметр **AUnique = True,** то перед созданием окна система проверяет уже открытые окна на наличие в них фреймов с заданным классом, и еслди такое окно обнаруживается и его ID равен параметру AID, то система не создает нового окна, а активизирует существующее окно.

### property Caption ;

**Область видимости:** **public**

**Объявление:**   
**property Caption: String ;**

****Доступ:** r/w**

**Описание:**

Свойство содержит заголовок, который будет установлен дочернему окну, на котором будет размещен фрейм.

### property ContextName ;

**Область видимости:** **public**

**Объявление:**   
**property ContextName: String ;**

****Доступ:** r/w**

**Описание:**

Свойство содержит строку описывающую контекст, в котором выполняется фрейм.

### property Folder;

**Область видимости:** **public**

**Объявление:**   
**property Folder[FolderKind: String]: String;**

****Доступ:** ReadOnly**

**Описание:**

Свойство возвращающее полный путь к одному из служебных каталогов системы. В качестве параметра **FolderKind** можно использовать константы, объявленные в классе **TFolderKind:**

**TFolderKind = **class****

**public**

****const** Bin = 'BIN' ;**

****const** PlugIn = 'PLUGIN' ;**

****const** Report = 'REPORT' ;**

****const** Help = 'HELP' ;**

****const** Update = 'UPDATE' ;**

**end;**

## **Protected методы**

### procedure Init ;

**Область видимости:** **protected**

**Объявление:**   
**procedure Init ; virtual ;**

**Описание:**Процедура вызывается автоматически из конструкторов фрейма CreatePlugIn (см. Создание фрейма (конструкторы фрейма)). В классе TSymphonyPlugInBaseFrame метод Init вызывает метод LoadState, который восстанавливает ранее сохраненные пользовательские данные при уничтожении фрейма (ширина и положение элементов управления, ID записи с которой работал пользователь при закрытии фрейма и т.д.). Если в дочернем классе требуется сделать дополнительную инициализацию, то класс должен переопределить метод Init.

### procedure SetSession ;

**Область видимости:** **protected**

**Объявление:**  
**procedure SetSession(ASession: TObject) ; virtual ; abstract ;**

**Параметры:**

**ASession: TObject** - объект обеспечивающий соединение с базой данных.

**Описание:**   
Абстрактная процедура, которая должна быть переопределена в классах потомках. Процедура должна проверить тип соединения, и если тип соединения соответствует ожидаемому, то настроить объекты фрейма для использования переданного объекта для связи с базой данных.

Процедура вызывается автоматически из конструкторов фрейма CreatePlugIn (см. Создание фрейма (конструкторы фрейма)).

Например, фрейм класса **TSymphonyPlugInORABaseFrame** переопределяет данную процедуру, проверяет что параметр ASession должен быть класса **TOraSession**, и если проверка прошла успешно, назначает данное соединение всем компонентам библиотеки ODAC, находящимся на фрейме.

### procedure SetData ;

**Область видимости:** **protected**

**Объявление:**  
**procedure SetData(AData: TObject) ; overload ; virtual ; abstract ;**

**procedure SetData(AData: ISymphonyPlugInCommand) ; overload ; virtual ; abstract ;**

**Параметры:**

**AData: TObject** или **AData: ISymphonyPlugInCommand** - данные передаваемые фрейму при его создании.

**Описание:**   
Абстрактная процедура, которая должна быть переопределена в классах потомках. Процедура предназначена для передачи параметров фрейму при его создании. Вызывающая программа может передать параметры в двух различных форматах. Процедура должна прилучить данные и использовать их в своих целях.

Процедура вызывается автоматически из конструкторов фрейма CreatePlugIn (см. Создание фрейма (конструкторы фрейма)).

### procedure SetParent;

### procedure AddControlLeft(AStateStorage: TStateStorage; AControl: TControl) ;

### procedure AddControlTop(AStateStorage: TStateStorage; AControl: TControl) ;

### procedure AddControlPoint(AStateStorage: TStateStorage; AControl: TControl) ;

### procedure AddControlWidth(AStateStorage: TStateStorage; AControl: TControl) ;

### procedure AddControlHeight(AStateStorage: TStateStorage; AControl: TControl) ;

### procedure AddControlSize(AStateStorage: TStateStorage; AControl: TControl) ;

### procedure AddControlVisible(AStateStorage: TStateStorage; AControl: TControl) ;

### procedure AddControl(AStateStorage: TStateStorage; AControl: TControl) ;

### procedure AddControlsToStorage(AStateStorage: TStateStorage) ; virtual ;

### procedure AddParamsToStorage(AStateStorage: TStateStorage) ; virtual ;

### procedure ApplyParamsToStorage(AStateStorage: TStateStorage) ; virtual ;

### class function DoExecuteCommonMethod(MethodName: String; Params: TObject = nil): Boolean ; virtual ;

### procedure Print(Params: TObject = nil) ; virtual ;

### procedure ExportToExcel(Params: TObject = nil) ; virtual ;