WorkSpace Manager. Краткая характеристика.

Мотивация

Идея разработки приложения WorkSpaceManager появилась при попытке использовать приложение SolutionManager, входящее в состав профессиональной версии Visual Prolog фирмы PDC.

Нужда в его использовании была вызвана тем, что в работе находились одновременно несколько взаимозависимых проектов, которые надо было совместно компилировать, запускать на исполнение. В особенности напрягала ситуация использования системы Visual Prolog непосредственно "из под пера" ее разработчиков, когда надо было компилировать все множество проектов (их было около 50, включая различные небольшие примеры и тесты)

Не устроили несколько моментов:

- работа только с проектами системы Visual Prolog, в то время, как требовалось использовать различные командные файлы или даже различные редакторы;
- невозможно было изменить порядок обработки проектов;
- невозможность группировки проектов;
- добавление проектов из одной директории не позволяло исключить из рассмотрения некоторые из них и опять-таки изменить порядок их обработки;
- невозможность использования относительной адресации проектов в дисковом пространстве.

В то же время нас интересовали темы:

- Строгое разделение логической части приложения и его пользовательского интерфейса;
- Микросервисы;
- Использование концепции плагинов в приложениях;
- Возможность создания удаленных сервисов создания и компиляции проектов Visual Prolog (идея однажды родилась в дискуссии Лео Йенсена и Томаса Пулса).

Поэтому в проекте WorkSpaceManager (WSM) сошлись воедино наши потребности и наши архитектурные поиски, применительно к приложениям на базе Visual Prolog.

Предполагается на этом проекте отрабатывать приемы и средства для построения полноценных микросервисов.

Проект до первой рабочей версии был разработан В. Юхтенко (Prolog Development Center SPb), позднее в работе принял участие Борис Белов. В решении задач http взаимодействия оказал помощь Андрей Басукинский.

Функциональность

Файл

Основной единицей обработки в WSM является файл, расширение которого определяет его тип.

Над файлом каждого типа могут выполняться четыре операции "Оп1", "Оп2", "Оп3" "Оп4". Характер каждой из операций, применительно к файлу определенного типа, не является фиксированным и определяется пользователем.

В общем виде каждая операция выглядит как командная строка в системе MSWindows

<исполняющая программа> <префикс> <ресурс> <суффикс>

Так для операции "Build" (Построить) проектного файла **test.vipprj** системы Visual Prolog это выглядит так (с точностью до маршрутов):

vipBuilder.exe /build test.vipprj

Что вызывает построение проекта **test.vipprj** и создание исполняемого приложения **test.exe**, в директории, установленной проектом.

Однако для обработки любого типа файла может быть выбрано приложение, по ассоциации, установленой в системе Windows.

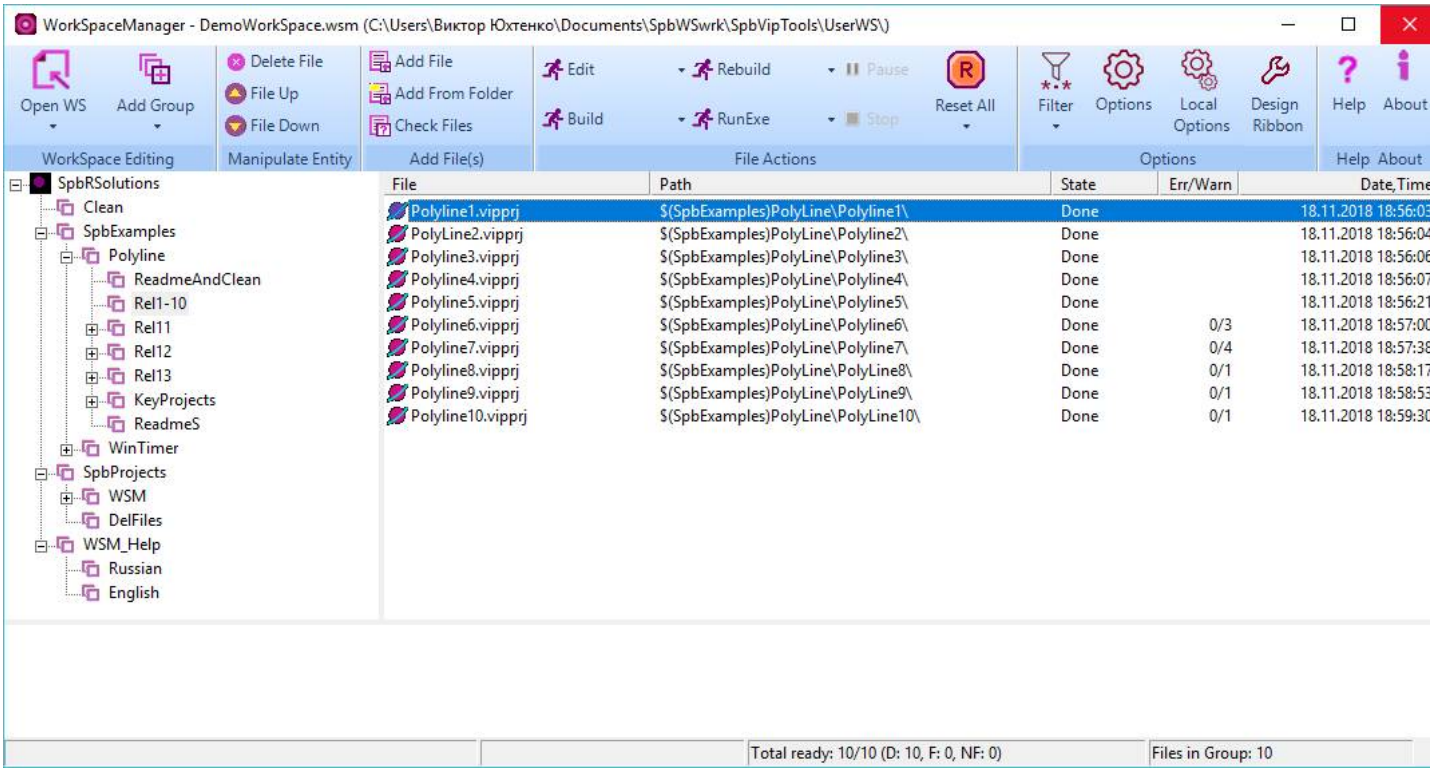
WorkSpace

WorkSpace (Рабочее Пространство) содержит множество файлов, актуальных для пользователя. Пользователь добавляет и удаляет файлы из рабочего пространства по мере необходимости.

WSM позволяет структурировать множество файлов, группируя по критериям, определяемым пользователем. Структура представляется в виде дерева и не является отображением файловой системы.

Структура WorkSpace и результаты обработки файлов сохраняются в файле с расширением .wsm в формате XML.

Общий вид приложения приведен на следующем рисунке:



Список файлов и дерево

Основным рабочим пространством пользователя является список файлов, расположенный в правой части формы.
Здесь указано имя файла , маршрут к нему, состояние обработки, число ошибок и предупреждений, в ходе обработки и время последней обработки.

Предусмотрены четыре состояния результата обработки:

- Done - выполнено успешно
- Failed - выполнено неуспешно
- NoFile - Файл не найден

Над каждым из файлов может быть выполнена одна из четырех описанных операций (Оп1,Оп2,Оп3,Оп4), допустимых для файлов соответствующего типа.
Над каждым файлом выделенного (отмеченного) списка также может выполняться одна из этих операций. Операции выполняются в порядке очередности сверху вниз.

Дерево в левой части формы является логическим представлением пользователя о своем рабочем пространстве. Первоначально дерево содержит всего один узел - корень, и все ресурсы правой части (если они есть) принадлежат этому узлу. Пользователь может создать произвольную древовидную структуру

Предусмотрены два типа узлов дерева - **группа** и **папка**.
Группа может содержать подгруппы и ей может принадлежать произвольное число файлов. В группе предусмотрены операции добавления, удаления файлов и их перемещения.
Папка соответствует директории дискового пространства и содержит только те файлы, типы которых определены в WSM, и которые содержатся в директории и в ее поддиректориях.
Папка не может содержать групп. Пользователь может перемещать файлы папки и (условно) удалять их. Удаленные файлы остаются видимыми и меняют лишь яркость отображения на экране. Пользователь может вернуть файл в активное состояние.

В правой части формы отображаются все файлы выбранной слева группы, включая все файлы всех подгрупп и папок. Это позволяет выполнять операции либо над всеми файлами рабочего пространства, либо только над частью, находящейся в фокусе пользователя в данный момент.

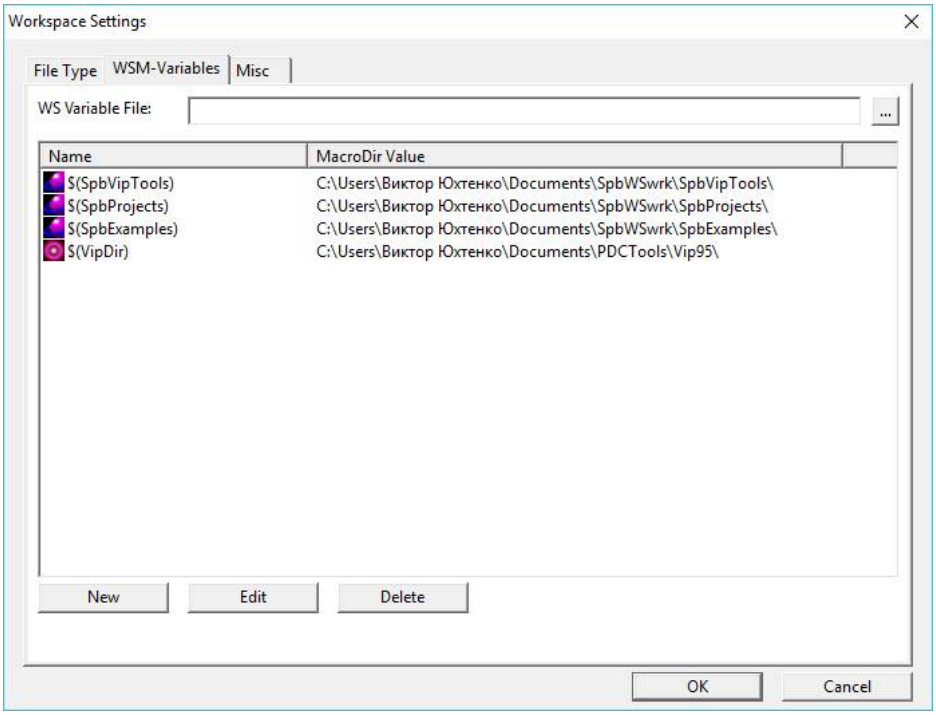
Над списком файлов и над деревом могут выполняться операции Drag-And-Drop, что позволяет оперативно перестраивать приоритеты и стратегию обработки. Перемещения с помощью клавиатуры тоже возможно.

WSM-переменные

Пользователь может использовать как абсолютную, так и относительную адресацию файлов в дисковом пространстве. Относительная адресация аналогична принятой в системе Visual Prolog. Предварительно определяется именованный маршрут имя маршрута используется в качестве префикса в маршруте файла. Такой именованный маршрут здесь называется WSM-переменной.

Относительная адресация позволяет переносить рабочее пространство с одного компьютера на другой, переопределив лишь значения WSM-переменных.

Пользователь может определить неограниченное число WSM-переменных и редактировать их значение. На следующем рисунке показан редактор WSM-переменных.



Язык

Язык пользовательского интерфейса может быть локализован применительно к предпочтениям пользователя.
Файл настройки языка LanguageWSM.xml расположен (и всегда должен там располагаться) в директории исполняемого файла.

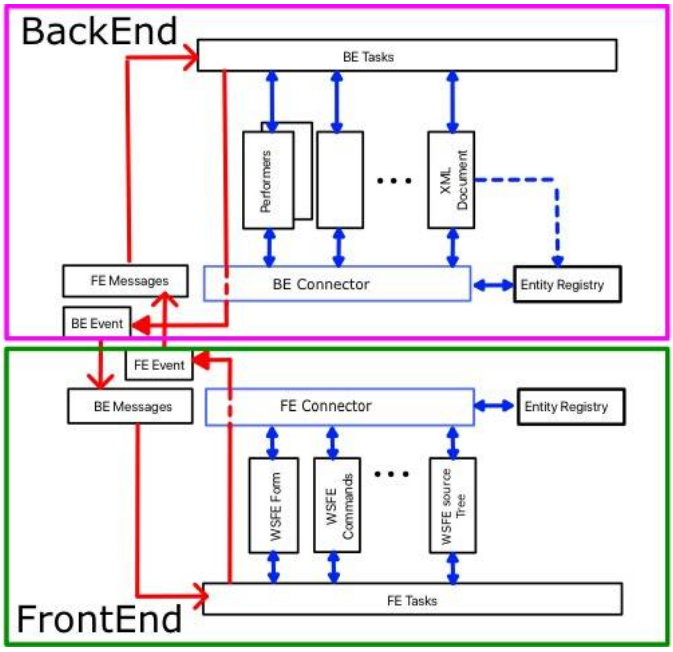
Базовым языком интерфейса является английский, но большинство надписей формы, диалогов и сообщений может быть переназначено путем редактирования файла LanguageWSM.xml.

Переключение языка пользовательского интерфейса находится в разделе **Misc** диалога **WorkSpace Settings**.

Архитектура

Архитектура приложения с самого начала разработки была ориентирована на строгое разделение пользовательского интерфейса от логической части приложения в расчете на последующее использование технлогии микросервисов.
Поэтому первая реализация приложения сразу же предстматривала его разделение на FrontEnd и BackEnd.

Структура моно приложения приведена на следущем рисунке



Здесь синими стрелками показаны связи внутри BackEnd и FrontEnd соответственно, а красными - организация связи между BackEnd и FrontEnd.

Внутри этих структур модули получают доступ к другим модулям через BE(FE)-Connector, который обращается к регистру объектов (Entity Registry).

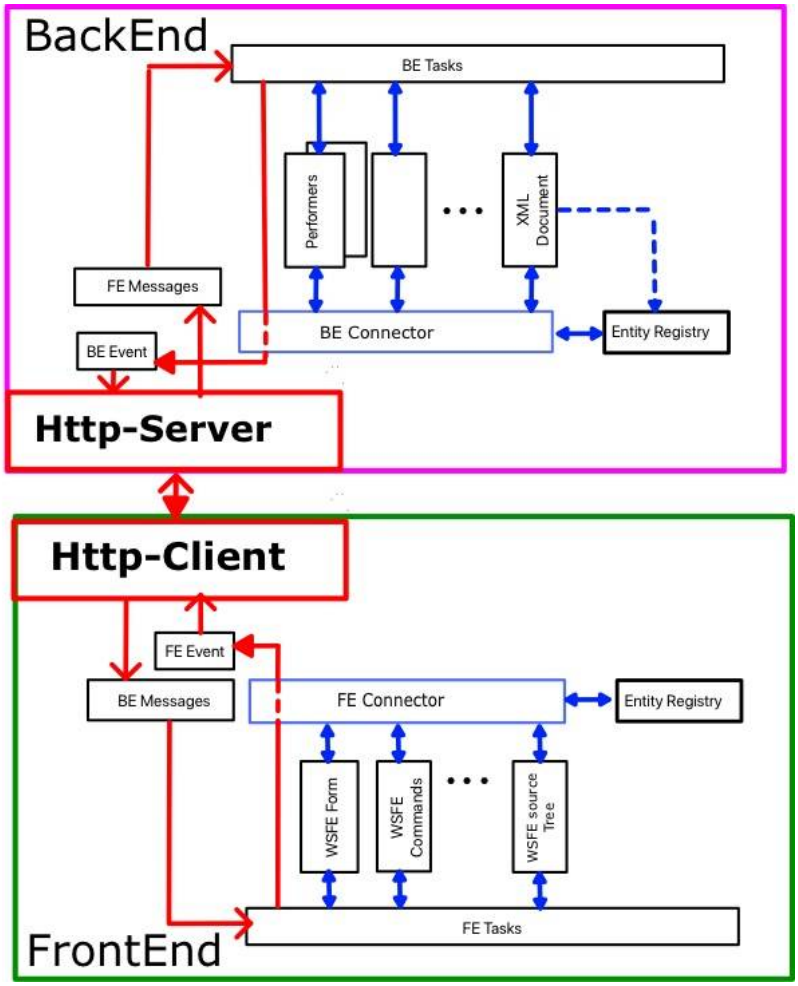
События, возникающие по инициативе противоположной стороны обрабатываются модулем BE(FE) Messages, который вызывает на исполнение соответствующие предикаты модуля BE(FE) Tasks. Тот, в свою очередь, обращается к остальным модулям BackEnd (FrontEnd). Получив данные для противоположной стороны, модуль BE(FE)Tasks посылает сообщение через модуль BE(FE) Event. Противоположная строна подписывается на события источника данных для нее.

Инициатором обмена данными всегда является FrontEnd, который инициирует выполнение задач в BackEnd.

Работа происходит в асинхронном режиме: отправив запрос, FrontEnd возвращается к своей работе. BackEnd, получив запрос выполняет задачу и, возможно, инициирует передачу данных во FrontEnd. Это может быть либо одно сообщение, либо последовательность сообщений, которые должны быть обработаны во FrontEnd.

Такая структура приложения позволила практически безболезненно перейти к варианту раздельных приложений (BackEnd и FrontEnd), взаимодействующих по http- протоколу, добавив и включив в работу Http-Сервер и Http-Клиент.

Структура такого приложения приведена на следующем рисунке



Client и **Server** сами преобразовывают данные и организуют их передачу между BackEnd и FrontEnd.

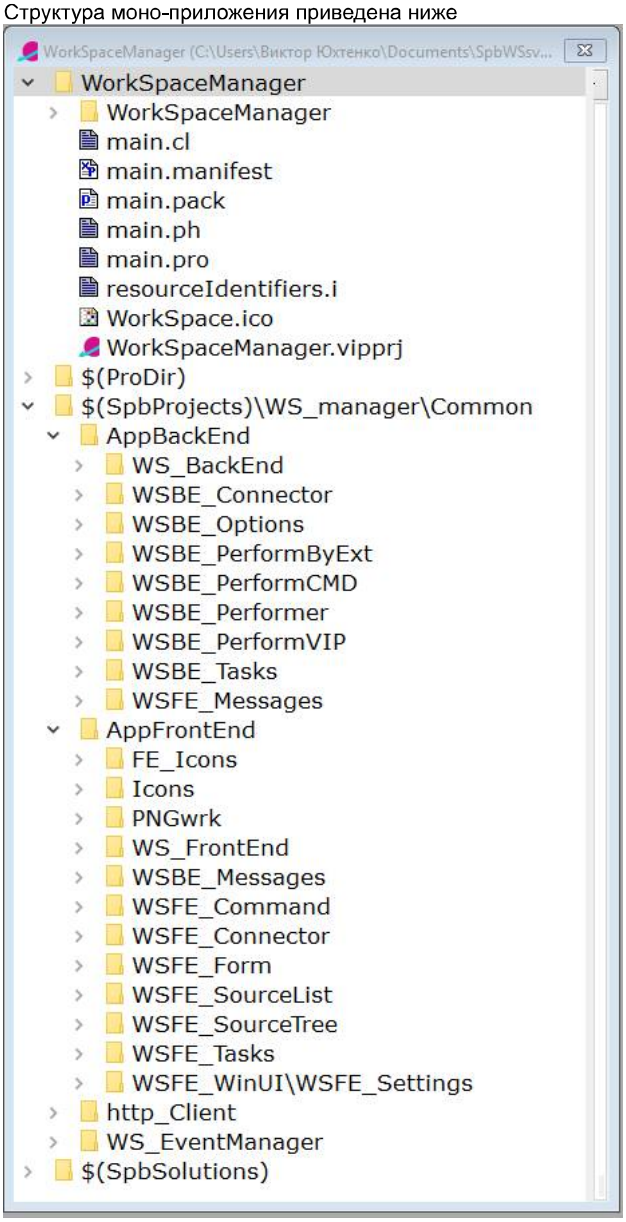
Технологии

При разработке WSM использованы технологии PDC VIP v.9, технологии и приемы собственной разработки:

- Регистрация объектов в общедоступной памяти;
- Организация хранения данных в формате XML;
- Обмен данными на основе событий (Event Messaging) в формате namedValueList;
- Организация пространства формы на базе split Screen;
- Организация управления на базе ribbon-панели;
- Клиент-серверная архитектура;
- Обмен данными в по http протоколу;
- Обмен данными в формате Json (при работе по http протоколу);
- Использование очередей сообщений в условиях многопоточного доступа (при работе по http протоколу).

Проектная структура приложений

Структура моно-приложения приведена ниже



А здесь приведены структуры проектов для Http-FrontEnd и Http-BackEnd