

Краткие технические требования к библиотеке конструкторов “Constructiva”

I. Версии библиотеки

Библиотека должна быть выполнена из максимально независимых друг от друга модулей зависимости между которыми должны быть определены на этапе объектного анализа. Библиотека должна быть представлена в следующих вариантах:

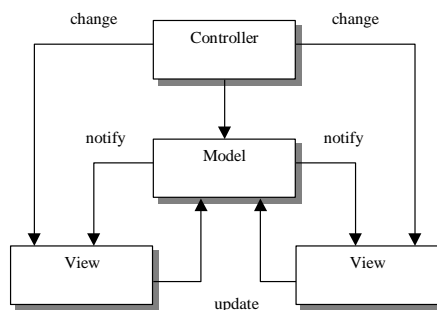
1. Облегченный вариант для использования в Internet. Конструктор и облегченная версия библиотеки скачиваются по сети на клиентский браузер, и полностью функционируют в Java машине клиента. (условное название “LITE”).
2. Полный вариант для локального использования. Все классы находятся на локальном диске пользователя (условное название “FULL”).
3. Вариант тонкого клиента, когда симулятор и элементы находятся на сервере, а репрезентации на клиентской машине. Клиент должен скачать только классы отвечающие за графическую репрезентацию элементов и небольшую библиотеку, отвечающую за отображение данных и коммуникацию с сервером, элементы и вычислительные алгоритмы располагаются на сервере и функционируют в Java машине сервера, а графические репрезентации функционируют в Java машинах клиентов (условное название “REMOTE”).
4. Вариант терминального использования, когда весь конструктор с элементами, видами и базовыми сервисами находится на сервере, а на клиенте работает только сервис GUI, отвечающий за прорисовку. В этом варианте на клиентскую машину скачивается постоянное количество классов, вне зависимости от конструктора и его сложности. Данный вариант требует высоко скоростного подключения. (условное название “TERMINAL”).

Симулятор, созданный на полной версии библиотеки, должны работать на всех остальных вариантах.

II. Базовые концепции (Core)

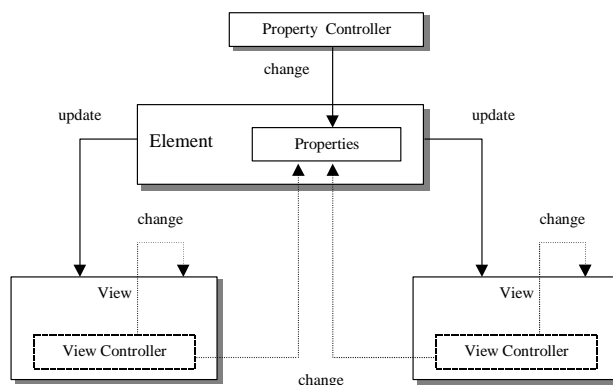
Core.1 Реализация модели MVC

Для сложных объектно-ориентированных систем хорошо зарекомендовал себя подход по разделению функциональной части системы от ее пользовательского интерфейса. Довольно распространенным является подход на основе модели Model/View/Controller. Традиционная схема модели Model/View/Controller выглядит следующим образом:



Цель этой модели обеспечить отделение прикладного объекта (Model) от способа его представления его пользователю (View) и от способа воздействия на него (Controller). Данная модель является наиболее подходящей концепцией для создания конструкторов. Основным понятием конструктора являются элементы, из которых создаются необходимые пользователю конфигурации. Элементы - это абстракция неких объектов обладающих свойствами и функциональностью определенной над этими свойствами. Для визуализации свойств, присущих элементам используются виды или репрезентации. У одного элемента может быть несколько видов. Для изменения свойств элемента используется два пути, первый через специальный редактор свойств, позволяющий наиболее полно и точно настроить объект и с помощью интерактивного воздействия пользователем с помощью мыши или клавиатуры на

репрезентацию объекта. Таким образом схема MVC в применении к конструкторам преобразуется в следующую:



ViewController может не присутствовать в View, поэтому он помечен пунктирной линией.

Таким образом библиотека должна содержать ключевых понятия:

1. Элемент содержащий свойства и функциональность (Element)
2. Свойства элемента (Property)
3. Вид элемента (View)
4. Контроллер свойств элемента (Property Controller & View Controller)

Элементы должны поддерживать иерархическое владение по схеме: один родитель (Parent) – несколько детей (Child). В вершине дерева владения стоит корневой элемент (Root) с нулевым родителем. Виды также обладают иерархическим владением, дерево владения видом должно совпадать с деревом владения элементов.

Core.2 Object Request Broker

Как следует из сущности проектируемой системы, в ней будет содержаться значительное количество объектов, взаимодействующих друг с другом. Элементы взаимодействуют со своими контроллерами, видами и с другими элементами. Библиотека должна предоставлять универсальный и удобный механизм общения объектов друг с другом, в тоже время взаимодействие объектов не должно происходить прямыми вызовами методов используя явные ссылки на Java объекты. Таким образом библиотека должна включать в себя средства идентификации объектов и средства универсального исследования объектов во время исполнения, вызова их методов или посылки сообщений. В мировой практике такие системы принято называть ORB (Object Request Broker). Реализация общения объектов с использованием ORB несколько усложнит программирование, но обеспечивает прозрачное и контролируемое библиотекой взаимодействие объектов не только в пределах одной машины, но и в пределах нескольких машин в сети, что позволит реализовать версию библиотеки REMOTE с масштабируемой архитектурой.

Core.3 Инструменты и Панели Инструментов (Tools and ToolBars)

Инструменты (Tools) в понятии конструкторов, это средства воздействия на элементы. Инструмент визуально представляет собой кнопку с картинкой в области конструктора, нажимая на которую пользователь может производить определенные изменения в объектах или всем конструкторе, активизировать моды, либо активизировать определенную функциональность, ассоциированную с данным инструментом. Инструменты объединяются в панели инструментов (ToolBar). В библиотеке должны быть реализованы следующие стандартные инструменты:

1. Select Tool – инструмент для выделения элементов, вызова редактора свойств элемента, удаления элементов, (передвижения элементов).
2. Delete Tool – инструмент для удаления элементов.
3. Create Tool – инструмент для создание новых элементов в симуляторе.

Также должна быть реализована стандартная панель управления конструктором, включающая кнопки:

1. Run – активизирует симулятор

2. Stop – останавливает симулятор
3. Pause – приостанавливает симулятор с возможностью дальнейшего продолжения.
Данная кнопка не является обязательной для панели управления.

Стандартные инструменты и палитры должны присутствовать в обеих версиях библиотеки, но должны входить туда как дополнительные компоненты, а не как неотъемлемая часть.

III. Сервисы (Services)

Сервис – некоторый объект существующий независимо от элементов конструктора, реализующий некоторую функциональность в дополнение к базовым концепциям. Все взаимодействие пользователя и сущностей конструктора (элементы, виды, свойства, инструменты, управляющие элементы) происходит через систему сервисов. Сервис может быть легко добавлен и удален из системы, при этом система должна продолжать функционировать. Для большей интерактивности отдельные сервисы будут работать в своих потоках. Все сервисы будут доступны для использования из кода симулятора, а также некоторые из них будут доступны через GUI пользователю. Каждый сервис может получить доступ к всей совокупности элементов в системе.

Services.1 Сервис публикации событий (Event Publishing Service)

Помимо взаимодействий объектов друг с другом через ORB в конструкторах необходима поддержка широковебчатых сообщений. Такие сообщения посылаются одним объектом через сервис публикации событий сразу нескольким подписчикам на сообщение данного типа. Сервис может реализовывать как синхронную рассылку сообщений, так и асинхронную.

Services.2 Именованние объектов (Naming Service)

Сервис именованния объектов ответственен за поддержку человеческих имен объектов. Объекты должны обладать уникальными именами в рамках конструктора. Возможны различные реализации системы имен, например доменная система, когда полное имя объекта формируется как доменное имя, составленное из имен родителей вплоть до корневого элемента, при этом объекты должны обладать уникальным именем в пределах родителя, или система глобальных уникальных имен, когда каждый объект имеет короткое имя уникальное на уровне всего конструктора. Сервис должен представлять следующую функциональность (API):

1. Возможность установки человеческого имени объекта по идентификатору объекта.
2. Возможность по идентификатору объекта получать его человеческое имя.
3. Возможность по человеческому имени получать идентификатор объекта.

Как дополнительное расширение сервиса может быть рассмотрена возможность изменения человеческого имени пользователем через GUI.

Services.3 Буфер обмена (Clipboard)

Сервис буфера обмена должен позволять копировать объекты или группу объектов в буфер обмена, вставлять объекты в конструктор из буфера обмена, вырезать объекты из конструктора в буфер обмена. Буфер обмена должен быть доступен как для программного использования из кода конструктора, так и с использованием Select tool посредством выделения объектов и использования команд Ctrl-X, C, V.

Services.4 Запись/Воспроизведение сценариев (Scripting Service)

Сервис сценариев должен позволять записывать и воспроизводить последовательность операций производимых пользователем над конструктором. Записанная последовательность должна уметь представляться в виде текстового файла с возможностью редактирования вручную, причем в таком виде последовательность действий базируется на NamingService для указания объектов над которыми проводятся действия. Сервис сценариев должен быть доступен как из кода конструктора, так и для пользователя с помощью окна сценариев.

Services.5 Запись/Считывание конфигурации (IO Service)

Сервис ввода/вывода отвечает за сохранение конфигурации объектов созданных в конструкторе в абстрактное хранилище и за восстановление этой конфигурации в дальнейшем. Сервис возможно будет поддерживать несколько форматов

1. Текстовый (Базируется на Naming Service для идентификации объектов).
2. Сжатый бинарный файл
3. XML документ (Базируется на Naming Service для идентификации объектов).

Services.6 Откат действий пользователя (Undo/Redo Service)

Undo/Redo сервис позволяет совершать откат последних действий пользователя, либо при ошибочном откате повторять действия. По всей видимости данный сервис вместе с сервисом обслуживания сценариев является одним из наиболее сложных частей системы. Оба сервиса должны быть заработаны таким образом, чтобы от программиста не требовалось дополнительной поддержки для функционирования этих сервисов, т.е. объект не должен ничего знать о их существовании, а сервисы лишь обеспечивают некоторую дополнительную функциональность.

Services.7 Сервис пользовательского интерфейса (GUI Service)

Сервис занимается отрисовкой графических объектов (виды элементов, элементы управления), поддерживает иерархическое владение графических объектов, обработку и передачу сообщений мыши и клавиатуры, в дерево графических объектов. Данный сервис должен быть расширяем для использования динамического масштабирования графики и удаленной отрисовки.

IV. **Стандартные управляющие элементы (Standard Controls)**

С библиотекой конструкторов могут быть использованы управляющие элементы из вспомогательного пакета Controls, входящего в библиотеку для создания симуляций. Данная библиотека не включается ни в один вариант библиотеки, а используется при необходимости в конкретных симуляторах. Управляющие элементы являются контроллерами для элементов в смысле модели MVC. Также на основе стандартных управляющих элементов может быть создан PropertyController для элементов.

В стандартные управляющие элементы включаются:

1. Button
2. Check Box
3. Checkbox Group (Radio Button)
4. Slider
5. Text Filed
6. Text Area
7. Choice
8. Spin Box

V. **Конфигурации (Configurations)**

Все конфигурации будут включать в себя реализацию модели MVC, некоторый вариант ORB, GUI сервис, поддержку инструментов и панелей инструментов, сервис широковещательной публикации событий.

Configurations.1 LITE

Включает только минимальный комплект модулей.

1. MVC and Tools implementation
2. Local ORB
3. GUI service
4. Event Publishing Service

Клиент скачивает по сети все указанные модули + код симулятора.

Configurations.2 FULL

Включает следующие модули:

1. MVC and Tools implementation
2. Local ORB
3. GUI service
4. Event Publishing Service
5. Naming Service

6. Clipboard
7. IO Service
8. Scripting Service
9. Undo/Redo Service

Все модули находятся локально на машине пользователя.

Configurations.3 REMOTE

Клиентская часть включает следующие модули:

1. GUI Service
2. Distributed ORB

Клиент скачивает указанные модули + код отвечающий за отрисовку Views.

Сервер включает следующие модули:

1. MVC and Tools implementation
2. Distributed ORB
3. Event Publishing Service
4. Naming Service
5. Clipboard
6. IO Service
7. Scripting Service
8. Undo/Redo Service

Configurations.4 TERMINAL

Клиентская часть включает модуль Remote GUI Service, причем весь код симулятора находится и функционирует на сервере, а клиент для любого конструктора скачивает небольшое константное количество класс файлов. Серверная часть будет включать:

1. MVC and Tools implementation
2. Local ORB
3. Remote GUI service
4. Event Publishing Service
5. Naming Service
6. Clipboard
7. IO Service
8. Scripting Service
9. Undo/Redo Service