**ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ**

**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

**Факультет компьютерных наук**

**Кафедра «Информационные системы»**

*<Система интеграции Web-сайтов рекрутмента>*

071900 (230201) Информационные системы и технологии

*<Информационные системы и телекоммуникации>*

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_Тюкачев Н.А. к.ф.-м.н., доцент \_\_.\_\_.2013

Студент \_\_\_\_\_\_\_\_\_\_Петрушин И.А. \_\_.\_\_.2013

Руководитель \_\_\_\_\_\_\_\_\_\_Сапегин А.А. к.т.н., доцент \_\_.\_\_.2013

Воронеж2012

# ОГЛАВЛЕНИЕ

[Введение 3](#_Toc324342765)

[1. Постановка задачи 4](#_Toc324342766)

[2. Анализ задачи 6](#_Toc324342767)

[2.1 Анализ предметной области 6](#_Toc324342768)

[2.2 Основные термины 6](#_Toc324342769)

[2.3 Анализ требований 8](#_Toc324342770)

[2.3.1 Компонент “J3DT Framework” 9](#_Toc324342771)

[2.3.2 Компонент “Plugin J3DT” 9](#_Toc324342772)

[2.3.2.1 Требования к функциональным возможностям 12](#_Toc324342773)

[2.3.2.2 Требования к мастеру создания сценария 13](#_Toc324342774)

[2.3.2.3 Требования к специализированной перспективе 13](#_Toc324342775)

[2.3.2.4 Требования к специализированному редактору 14](#_Toc324342776)

[2.4 Существующие решения 14](#_Toc324342777)

[3. Аппаратные и программные средства 15](#_Toc324342778)

[4. Реализация 17](#_Toc324342779)

[4.1 Компонент “J3DT Framework” 17](#_Toc324342780)

[4.1.1 Декларативное описание графических объектов 17](#_Toc324342781)

[4.1.2 Диаграмма классов 20](#_Toc324342782)

[4.1.3 Диаграмма последовательностей. Загрузка объектов по указанному сценарию. 22](#_Toc324342783)

[Заключение 32](#_Toc324342784)

[Список литературы 33](#_Toc324342785)

[Приложения 34](#_Toc324342786)

# ВВЕДЕНИЕ

Ни для кого не секрет, что наше общество и экономика перешагнули стадию постиндустриализма и стали первооткрывателями новой ступени развития – стадии Информационного общества. С каждым днём растёт роль информации, знаний и новых технологий в жизни каждого современного человека. Рамки глобального информационного пространства расширяются с каждым годом, что позволяет людям эффективно взаимодействовать друг с другом, получать доступ к мировым информационным ресурсам и удовлетворять свои потребности в услугах.

Бурное технологическое развитие открыло миру огромное множество различных видов аппаратных и программных решений, которые в синтезе позволяют не только увеличить скорость обработки информации и сделать человека сверх коммуникабельным, но и оптимизировать затраты, связанные с эксплуатацией данных технологий.

На этом фоне появляются проблемы интеграции уже существующих продуктов, реализованных с помощью концептуально разных технологических средств и не имеющих общего, понятного всем протокола взаимодействия.

Ответной реакцией на такие удручающие перспективы послужило создание новых методологий и принципов, которые в итоге легли в основу универсальных интеграционных механизмов и стандартов.

В реальной жизни цель интеграции сводится к оптимизации каких-либо бизнес-процессов и/или объединение нескольких информационных ресурсов в одно целое. Во втором случае, количество до сих пор нерешённых задач пугает своим масштабом.

На данный момент существует много различных интернет ресурсов, на которых соискатель работы может выложить своё резюме в целях получить предложение о трудоустройстве от какой-либо заинтересованной в нём компании. Как правило, эти же ресурсы предоставляют аналогичную возможность публикации вакансий HR) отделам компаний. Как правило, соискатели\работодатели публикуют идентичные резюме\вакансии , сразу на нескольких ресурсах, чтобы увеличить вероятность ответных предложений\соглашений.

Как было отмечено ранее, на сегодняшний день существует большое количество рекрут-сайтов. Самые популярные из них:

* Русские: [headhunter.ru](http://headhunter.ru/) , [job.ru](http://job.ru/), [superjob.ru](http://superjob.ru/" \t "_blank), [rabota.ru](http://rabota.ru/" \t "_blank);
* Зарубежные: [monster.com](http://monster.com/), [dice.com](http://dice.com/);

Таким образом, для управления своим резюме\вакансией пользователю необходимо заходить на каждый сайт, где был опубликован профиль и проделывать одну и ту же работу, в целях обновления , по каким-либо причинам не актуальной на текущий момент, информации. Самые простые примеры таких ситуаций:

* Соискатель работы хочет добавить в своё резюме несколько новых навыков и изменить желаемую заработную плату;
* Работодателю необходимо изменить требования, предъявляемые к какой-либо должности.

Количество времени, требуемое для обновления информации, прямо пропорционально количеству ресурсов. Очевидно, что это далеко не оптимальный с точки зрения производительности и не удобный, из-за монотонности действий, способ. Но другого выбора у пользователей нет. Данные интернет ресурсы не предоставляют никаких программных интерфейсов для управления профилями и далеко не все поддерживают их импорт/экспорт.

В рамках данной курсовой работы будет реализована система синхронизации информации между рекрут - сайтами, услугами которых, пользуется пользователь.

# ПОСТАНОВКА ЗАДАЧИ

**Требуется разработать систему, которая позволит**

**пользователю:**

1. **Регистрироваться в системе под видом соискателя работы.**
2. **Создавать профиль своего резюме.**
3. **Управлять профилем (публикация на нескольких ресурсах, обновление, удаление) посредством единого пользовательского web-интерфейса.**
4. **Обеспечивать надежный, отказоустойчивый механизм синхронизации информации между системой и web-ресурсами.**

# АНАЛИЗ ЗАДАЧИ

* 1. **Анализ предметной области**

Прежде чем строить интеграционное решение, необходимо четко понять, какой подход будет наиболее эффективен для решения стоящей перед лицом задачи. Существует три подхода к интеграции информационных систем:

1. Интеграция на уровне данных. Суть данного подхода заключается в следующем: приложения работают независимо друг от друга, каждое использует свой набор данных. В случае необходимости осуществляется обмен данными между приложениями. При этом, если обмен данными осуществляется путем вызова сервисов или отправки/получения сообщений.
2. Интеграция на уровне бизнес-процессов. Суть данного подхода заключается в следующем: приложения выставляют сервисы, являющиеся интерфейсами к бизнес - логике данных приложений. Взаимодействие между приложениями реализовано в рамках бизнес-процесса, на отдельных шагах которого осуществляется вызов того или иного сервиса. Реализуется данный подход с помощью сервисной шины предприятия (*ESB*), которая занимается виртуализацией сервисов, предоставляемых приложениями, и решений класса *Business Process Management System (BPMS)*, как правило основанных на языках *BPEL* или*BPMN*, которые реализуют логику процесса.
3. Интеграция на уровне композитных приложений. Бизнес-логика отдельного приложения строится путем вызова сервисов, предоставляемых как данным приложением, так и другими системами. Таким образом, на одном шаге бизнес-процесса могут взаимодействовать несколько сервисов, в то время как при интеграции на уровне бизнес-процессов на одном шаге процесса вызывается один сервис. Реализация композитных приложений осуществляется с помощью использования технологий *Java Business Integration (JBI, JSR 208)* или Service Component Architecture (SCA).

Отсюда можно сделать вывод, что наиболее целесообразным медом интеграции, в рамках поставленной задачи, является подход, описанный в первом пункте.

* 1. **ESB как подход интеграции на уровне данных**

Самым оптимальным вариантом интеграции на уровне дынных является использование сервисной шины предприятия - Enterprise Service Bus (ESB).

Основной принцип сервисной шины — концентрация обмена сообщениями между различными системами через единую точку, в которой, при необходимости, обеспечивается транзакционный контроль, преобразование данных, сохранность сообщений. Все настройки обработки и передачи сообщений предполагаются также сконцентрированными в единой точке, и формируются в терминах служб, таким образом, при замене какой-либо информационной системы, подключённой к шине, нет необходимости в перенастройке остальных систем.

Важной особенностью сервисной шины является поддержка двух режимов передачи информации: синхронного и асинхронного.

В рамках поставленной задачи важно наличие именно второй характеристики, так её принцип обеспечивает надёжную передачу данных между приложениями, за счёт возможности использования сервисной шиной в качестве транспортного механизма технологии очередей сообщений (Message Queue). Таким образом , одно приложение посредством ESB может передать данные другому приложению без необходимости вызова процедуры получателя и определенно без ожидания результата. Отправитель не обязан знать, как найти получателя. Он может просто направить данные в ESB и быть уверенным, что они будут переданы.

* 1. **Очереди Сообщений**

Помимо возможности асинхронной передачи данных MQ обладают следующими полезными свойствами:

* Слабое связывание — очереди сообщений создают неявные интерфейсы обмена данными, которые позволяют процессам быть независимыми друг от друга т.е вы просто определяете формат сообщений отправляемых от одного процесса другому.
* Избыточность — Очереди позволяют избежать случаев неэкономного использования ресурсов процесса(например памяти) в результате хранения необработанной (лишней информации) информации.
* Масштабируемость — очереди сообщений позволяют распределить процессы обработки информации. Таким образом, они позволяют легко наращивать скорость, с которой сообщения добавляются в очередь и обрабатываются.
* Эластичность и возможность выдерживать пиковые нагрузки — очереди сообщений могут выполнять роль своего рода буфера для накопления данных в случае пиковой нагрузки, смягчая тем самым нагрузку на систему обработки информации и не допуская ее отказа.
* Отказоустойчивость — очереди сообщений позволяют отделить процессы друг от друга, так что если процесс, который обрабатывает сообщения из очереди падает, то сообщения могут быть добавлены в очередь на обработку позднее, когда система восстановится.
* Гарантированная доставка — использование очереди сообщений гарантирует, что сообщение будет доставлено и обработано в любом случае (пока есть хотя бы один обработчик).
* Гарантированный порядок доставки — большая часть систем очередей сообщений способны обеспечить гарантии того, что данные будут обрабатываться в определённом порядке (чаще всего в том порядке в котором они поступили).
* Буферизация — очереди сообщений позволяет отправлять и получать сообщения при этом работая с максимальной эффективностью, предлагая буферный слой — процесс записи в очередь может происходить настолько быстро, насколько быстро это в состоянии выполнить очередь сообщений, а не обработчик сообщения.
* Понимание потоков данных — очереди сообщений позволяют выявлять узкие места в потоках данных приложения, легко можно определить какая из очередей забивается, какая простаивает и определить что необходимо делать — добавлять новых обработчиков сообщений или оптимизировать текущую архитектуру.
  1. **Основные термины**

|  |  |
| --- | --- |
| **Термин** | **Описание** |
|  |  |

Таблица 2.2.1 Основные термины

* 1. **Анализ требований**

**Для реализации всей необходимой функциональности (пункт Постановка задачи) данная система должна состоять из четырёх основных компонент(рисунок )**

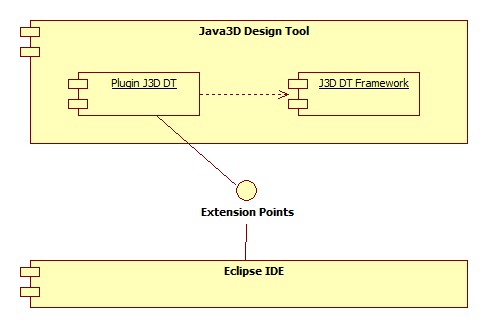


Рисунок 2.3.1 Диаграмма Компонентов

WEB CLIENT – компонент, который должен предоставлять пользователю прикладной web-интерфейс для взаимодействия с системой.

Message Broker – компонент осуществляющий асинхронную предачу сообщений между компонентом WEB SERVICE и удалёнными web-ресурсами, с предварительной трансформацией данных.

WEB SERVICE Dispatcher – промежуточное звено системы, отвечающее за принятие запросов от клиентской части и формирование/передачу запросов к компоненту Message Broker.

WEB SERVICE Router – компонент, предназначенный для рассылки HTTP-запросов к удалённым ресурсам с целью синхронизации данных.

* + 1. **Компонент “J3DT Framework”**
       1. **Диаграмма вариантов использования**

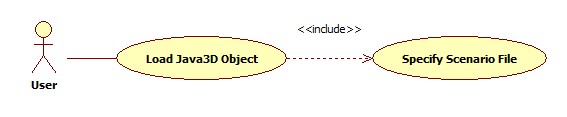


Рисунок 2.3.1.1.1 Диаграмма Use-Case

* + 1. **Компонент “Plugin J3DT”**

Как было отмечено выше, Eclipse предоставляет широкие возможности для расширения. В основе архитектуры, делающей это возможным, лежит использование плагинов*.* Вместо того чтобы находиться в одном монолитном jar-файле, код Eclipse разделен на множество модулей, загружаемых динамически и независимо друг от друга (хотя каждый плагин может указать список плагинов, которые должны быть загружены до него). Для того, чтобы сократить время запуска платформы, плагины загружаются только перед непосредственным использованием (разумеется, полный список установленных плагинов доступен всегда).

Для того, чтобы Eclipse загрузил плагин, необходимо написать дескриптор плагина – файл в формате XML под названием «plugin.xml», который можно создать как при помощи PDE, так и в любом текстовом редакторе. Дескриптор сообщает Eclipse, какие ресурсы входят в плагин и каким образом платформа может их использовать. В ресурсы обычно входят библиотеки «.jar», текстовые сообщения в файлах «.properties»,графические изображения и т.п.

Разработчики могут создавать свои плагины, которые имеют абсолютно равные права с компонентами, изначально входящими в Eclipse; фактически приложение Eclipse представляет собой набор плагинов (для удобства конфигурирования группа связанных плагинов может быть объединена в подсистему).

Eclipse поддерживает концепцию так называемых фрагментов плагинов. Фрагменты плагина дополняют основной плагин – они очень похожи на обычные плагины, за исключением того, что не является самостоятельными модулями и могут быть использованы только в комплекте с основным плагином. Eclipse рассматривает плагин и все его обнаруженные фрагменты как единый логический плагин.

Для чего нужны фрагменты плагинов? Их основным свойством является возможность установки независимо от основного плагина. Как правило, фрагменты используются для добавления локализованных ресурсов и зависимых от операционной системы компонентов. Если некоторый плагин поддерживает локализацию, то можно добавить поддержку нового языка при помощи отдельного фрагмента без изменения оригинального плагина и, более того, абсолютно независимо от его разработчиков.

Но, модульная архитектура еще не обеспечивает расширяемости платформы - она всего лишь обеспечивает механизм загрузки кода, написанного разработчиками. Новая функциональность может быть добавлена в Eclipse с помощью механизма расширений. Ключевое понятие здесь – точка расширения. Точка расширения определяет группу сервисов и дает возможность разработчикам плагинов добавлять свои сервисы в общий список.

Чтобы добавить совой плагин в Eclipse, разработчику необходимо реализовать нужную точку расширения и указать в дескрипторе плагина, какая точка расширения была реализована. При инициализации, Eclipse зарегистрирует новый плагин в общем списке плагинов и сделает его доступным для пользователей.

Что здесь является общим для всех реализаций механизма расширения, а что специфично для конкретного примера? Как правило, с точкой расширения действительно ассоциируется определенный интерфейс – через него клиенты точки расширения могут взаимодействовать с конкретными расширениями, не зная о деталях их реализации. Поскольку сценарии работы с различными точками расширениями, вообще говоря, не имеют ничего общего, большинство из точек расширений использует свой, уникальный.

Разумеется, возможны ситуации, когда использование интерфейса не является необходимым. Например, точка расширения может быть использована для сбора информации о графических ресурсах – в этом случае от расширений требуется лишь указать имя подключаемого файла.

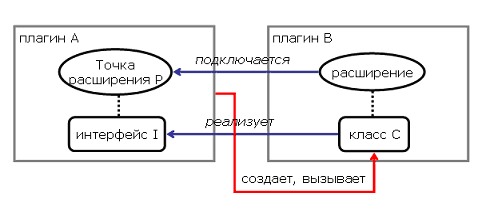


Рисунок 2.3.2.1. Механизм расширения Eclipse.

Клиенты точки расширения P (они не обязательно должны находится в плагине A, объявляющем точку расширения) могут получить список обнаруженных расширений P (в этом примере – единственное расширение в плагине B), создать их экземпляры и работать с ними через общий интерфейс I, объявленный в том же плагине, что и точка расширения (Рисунок 2.3.2.1).

Из приведенного примера может показаться, что существует жесткая взаимосвязь “один плагин – одна точка расширения”. На самом деле никакого ограничения на количество расширений, с которыми может работать один плагин, нет. Один и тот же плагин может объявлять свои точки расширения и подключаться к сторонним ( критерием объединения расширений в одном плагине является лишь логическая целесообразность).

Часто оказывается полезным создание плагинов, которые не подключаются к точкам расширения и не объявляет свои. Такие плагины содержат архивы «.jar» и используются в виде своеобразных разделяемых библиотек.

При помощи описанного механизма расширений может быть расширена практически вся функциональность Eclipse – перспективы, редакторы, виды, страницы настроек, инкрементальные билдеры.

Таким образом, механизм поддержки расширений никоим образом не ограничен стандартной функциональностью, а, напротив, поощряет разработку расширяемых приложений.

* + - 1. **Требования к функциональным возможностям**

**Для того, чтобы разрабатываемый плагин мог обеспечить взаимодействие пользователя с модулем загрузки сценариев, посредством Eclipse IDE , он должен быть встроен в среду с помощью интерфейсов расширений.**

|  |  |
| --- | --- |
| **Точка расширения** | **Функциональность** |
| **org.eclipse.ui.newWizard** | **Создание файла нового сценария в рабочем пространстве пользователя.** |
| **org.eclipse.ui.Editor** | **Редактирование сценариев.** |
| **org.eclipse.ui.Perspective** | **Композиция специализированных элементов управления.** |
| **org.eclipse.ui.Menu** | **Быстрый доступ к различным функциям плагина.** |

Таблица 2.3.1.1.1

* + - 1. **Требования к мастеру создания сценария**

Для инициализации файла сценария под платформой Java3D,пользователю должен быть доступен соответствующий мастер, который предоставит пользователю возможность указания следующих характеристик файла:

1. Название.
2. Расположение.
3. Опциональная генерация класса контроллера.

После завершения работы с мастером, в указанной рабочей области должен быть автоматически создан файл сценария. Также, по требованию пользователя , должен быть создан класс-контроллер сценария.

После инициализации стандартная перспектива должна быть автоматически изменена на специализированную.

* + - 1. **Требования к специализированной перспективе**

Перспектива должна включать в Workbench только те компоненты управления, которые необходимы для работы с активным проектом, а именно:

1. Визуальный редактор.
2. Навигатор**.**
3. **Стандартный обозреватель проектов Eclipse IDE.**
   * + 1. **Требования к специализированному редактору**

**Специализированный pредактор должен поддерживать следующие функции:**

1. **Поддержка чтения, записи и редактирования файлов** формата Java3D – сценария.
2. **Синхронизация внесенных изменений с файлом источником.**
   * + 1. **Диаграмма вариантов использования**

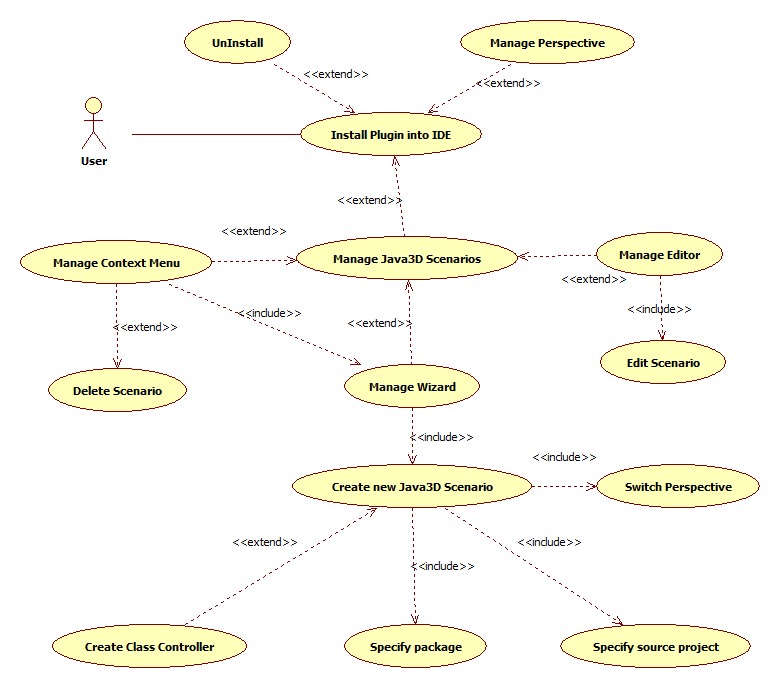


Рисунок 2.3.2.5.1 Диаграмма Use-Case

* 1. **Существующие решения**

На сегодняшний день не существует аналогов системе, которая рассматривается в данной курсовой работе.

# ****АППАРАТНЫЕ И ПРОГРАММНЫЕ СРЕДСТВА****

. В качестве средств реализации были выбраны следующие программные продукты и технологии:

|  |  |
| --- | --- |
| **Технология** | **Описание** |
| **Eclipse IDE PDE 3.7** | Платформа для создания плагинов |
| **JUnit 4.0** | Технология тестирования |
| **Apache Ant** | Инструмент для сборки |
| **Git** | Система контроля версий |
| **GitHub.com** | Хостинг – сервис для CVS Git |
| **JDOM** | Java инструмент-парсер для работы с XML |
| **Reflection** | Механизм исследования данных о программе и объектах во время её выполнения |

Таблица 3.1 Средства реализации

# Требования к составу и параметрам технических средств

|  |  |  |
| --- | --- | --- |
| **Требование** | **Минимальное значение** | **Рекомендуемое значение** |
| **Оперативная память** | 128 Мб | 1 Гб и более |
| **Свободное пространство на ЖД** | 300 Мб | 1 Гб и более |
| **Процессор** | 533 МГц | 1 Гб и более |

Таблица 3.1.1 Минимальные системные требования

* 1. **Требования к программным средствам, используемым системой**

Java Virtual Machine + Операционная система:

* Microsoft Windows;
* Linux;
* Mac OS.

Инструменты необходимые для разработки:

1. Eclipse IDE.
2. Plugin Development Kit.

|  |  |  |
| --- | --- | --- |
| **Требование** | **Минимальное значение** | **Рекомендуемое значение** |
| **Версия Java JDK , JRE** | 1.6. | 1.7.0 и выше |
| **Версия Java3D API 1.5** | 1.5 | 15.2 и выше |
| **Версия Eclipse** | 3.4 | 3.7 и выше |

Таблица 3.1.2 Минимальные программные требования

# ****РЕАЛИЗАЦИЯ****

* 1. **Компонент “J3D DT Framework”**
     1. **Декларативное описание графических объектов**

Для реализации разметки декларативного описания графических объектов было решено использовать язык XML, потому что он является платформонезависимым и хорошо подходит для решения данной задачи, так как легко читается человеком и с его помощью удобно описывать любые объекты и их свойства.

Была создана универсальная схема (см. приложение №1), которая задаёт правила написания сценариев. Ниже, приведён пример, в котором декларативно описывается один из объектов Java3D API .

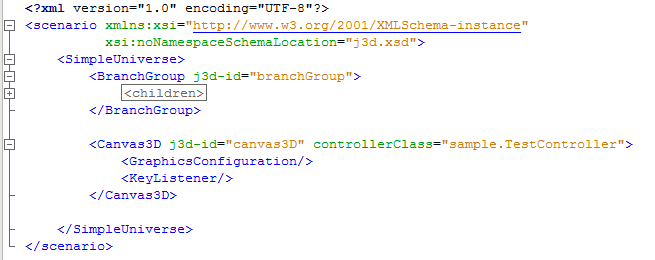


Рисунок 4.1.1.1 Пример j3dxml-сценария

Дочерние элементы каждого узла-объекта обрабатываются при его загрузке, как входные параметры для конструкторов и методов соответствующего класса.

Чтобы сделать загрузку Java3D объектов более гибкой и независимой от свойств самих объектов, был реализован механизм инжекции конфигураций, суть которого состоит в следующем: компонент содержит в себе архив XML – фалов (конфигураций), созданных по универсальной схеме (см. приложение №2). В каждой конфигурации описываются свойства элемента сценария. Таким образом, для каждого Java3D – объекта можно создать файл, в котором будет храниться следующая информация:

* название элемента сценария, к которому привязана данная конфигурация;
* тип конфигурации;
* тип Java3D – объекта;
* набор привязанных конструкторов, обработчиков и свойств.



Рисунок 4.1.1.2 Пример конфигурации для Java3D объекта - SimpleUniverse

В небольшом примере, показанном выше, демонстрируется конфигурация для объекта SimpleUniverse ,описанного в декларативном сценарии предыдущего примера. Значение атрибута корневого элемента (<config-element>) “nameOfElement” указывает на то, что конфигурация привязана к элементу сценария “SimpleUniverse”. Дочерний элемент узла <type> характеризует тип описываемого объекта. В данном случае наличие элемента <class> ,говорит о том, что описываемый объект является экземпляром класса. Далее следует набор конфигурационных элементов ,которые задают дополнительные свойства объекта. Перечень всех возможных элементов размещён в таблице ниже.

|  |  |
| --- | --- |
| **Элемент/Атрибут** | **Назначение** |
| **<class>** | элемент – экземпляр класса |
| **nameOfSource** | Название класса источника |
| **<setter>** | элемент – метод класса |
| **nameOfElement** | Название привязанного элемента |
| **<union>** | элемент - набор других объектов |
| **<controller>** | элемент – ссылка на метод привязанного к объекту контроллера |
| **<static-property>** | элемент – ссылка на статический метод или поле |
| **nameOfSource** | Название класса источника |
| **nameOfMethod** | Название метода |
| **nameOfField** | Название поля |
| **<constructor>** | Конструктор класса |
| **<handler>** | Обработчик |
| **handlerName** | Название обработчика |
| **<property>** | Параметр |
| **nameOfElement** | Название привязанного элемента |
| **propertyType** | Тип параметра |

Таблица 4.1.1.1 Элементы конфигурации

* + 1. **Диаграмма классов**

При реализации архитектуры модуля , были использованы следующие паттерны проектирования:

* Singleton;
* Abstract Factory;
* Façade.

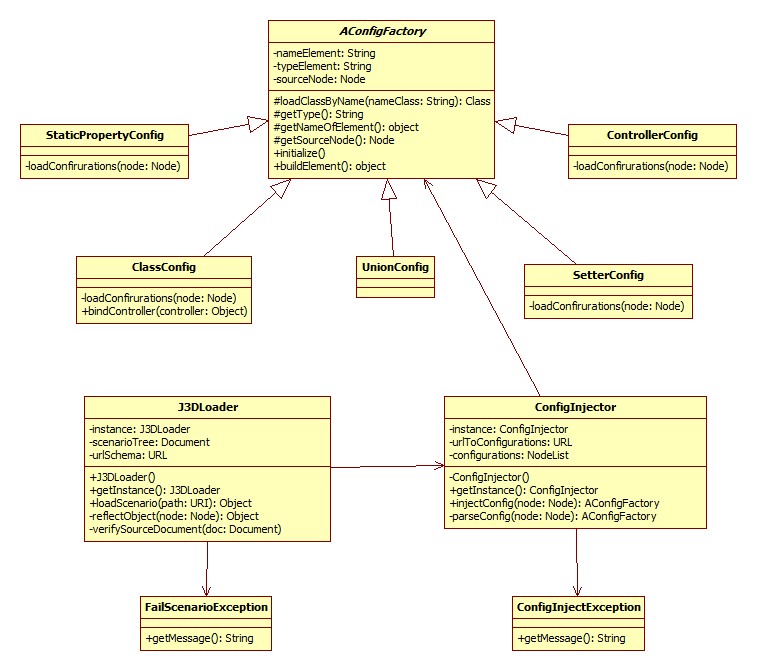
****

Рисунок 4.1.2.1 Диаграмма классов

|  |  |
| --- | --- |
| **Класс** | **Описание/Назначение** |
| **J3DLoader** | Загрузка Java3D объектов по указанному файлу сценария |
| **ConfigInjector** | Инжекция классов-конфигураций |
| **AConfigFactory** | Абстрактный класс. Базовая сигнатура методов инжекции конфигураций |
| **СlassConfig** | Реализация для элементов типа - <class> |
| **SetterConfig** | Реализация для элементов типа - <setter> |
| **UnionConfig** | Реализация для элементов типа - <union> |
| **ControllerConfig** | Реализация для элементов типа -<controller> |
| **StaticPropertyConfig** | Реализация для элементов типа -<static-property> |
| **ConfigInjectionException** | Класс-наследник Exception.Обработка ошибок, возникших во время парсинга конфигурационного файла |
| **FailScenarioException** | Класс-наследник Exception.Обработка ошибок, возникших во время парсинга файла сценария |

Таблица 4.1.2.1 Описание классов

* + 1. **Диаграмма последовательностей. Загрузка объектов по указанному сценарию.**

Для успешной загрузки Java3D-объекта, пользователю необходимо вызвать метод «loadScenario» у класса загрузчика (J3DLoader),передав ему в качестве параметра путь к файлу сценария. Далее никаких действий от пользователя не требуется. Класс J3DLoader,спроектированный по шаблону «Facade», сам позаботится о деталях загрузки, а именно:

(начиная со второго действия, в соответствии с диаграммой )

1. Проверка существования файла по указанной ссылке, и его соответствия схеме сценария. Загрузка объектной модели (дерева) документа.
2. Вызов метода «reflectObject». На вход передаётся текущий узел дерева сценария.
3. Вызов метода «loadConfig» у объекта класса ConfigInjector. На вход передаётся текущий узел дерева сценария, по имени которого загружается подходящий объект-наследник класса AConfigFactory (СlassConfig, SetterConfig, UnionConfig, ControllerConfig, StaticPropertyConfig). ConfigInjector обрабатывает файл конфигураций и возвращает подходящую для указанного элемента реализацию абстрактного класса.
4. Поиск дочерних элементов у текущего узла.
5. Если такие существуют, для каждого рекурсивный вызов метода описанного в шаге 3.
6. Вызов метода «buildElement» у объекта-наследника класса AConfigFactory. Конкретный класс-наследник по-своему реализует данный метод, но каждый из них пользуется Java – механизмом Reflection. C помощью него, динамически создаются и параметризируются Java3D-объекты.
7. Если дерево сценария полностью пройдено, объект класса J3DLoader вернёт пользователю рефлексивно созданный Java3D-объект.

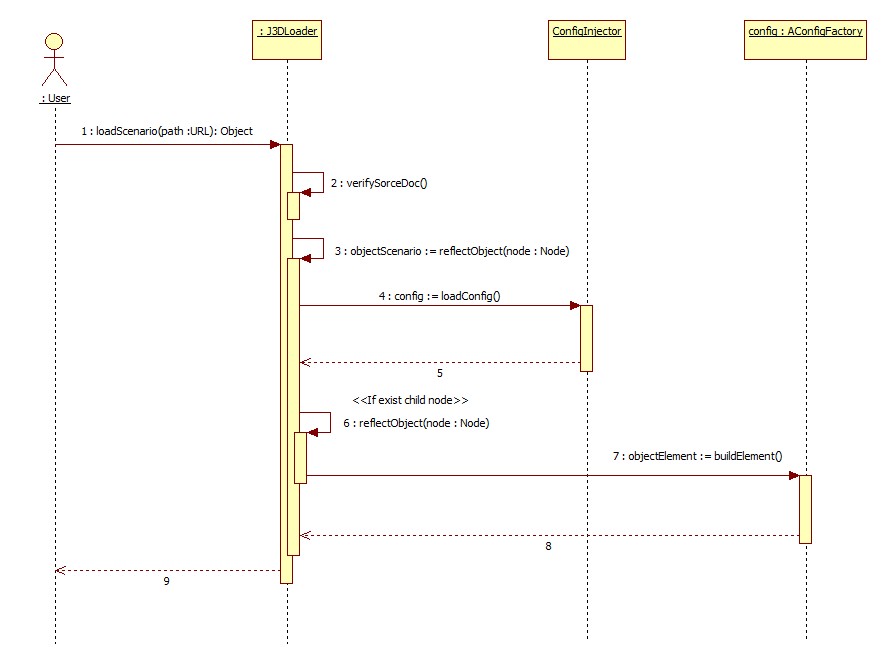


Рисунок 4.1.3.1 Диаграмма последовательностей

* + 1. **Проектирование в концепции MVC**

Для того, чтобы позволить пользователю создавать свои приложения под платформой Java3D в концепции MVC, был разработан механизм привязки класса контроллера к файлу сценария, суть которого состоит в следующем: любому Java3D-объекту, поддерживающему взаимодействие с интерфейсом EventListener, в фале сценария можно указать имя класса контроллера, который будет обрабатывать поведение объекта при наступлении заданного события. Класс контроллер, может содержать в себе реализацию нескольких интерфейсов (Рисунок 4.1.4.1).

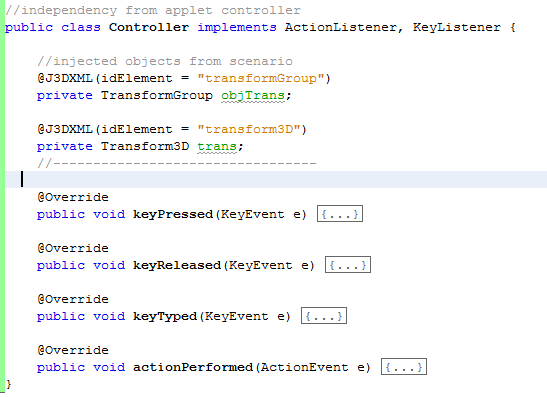


Рисунок 4.1.4.1 Пример класса контроллера

Чтобы не ограничивать пользователя в возможностях реализации поведения объекта, которое может зависеть от свойств других объектов сценария, был реализован дополнительный механизм инжекции: с помощью уникального интерфейса – аннотации @J3DXML (см. приложения) можно пометить любое поле класса контроллера и, указав в атрибуте “idElement” идентификатор необходимого объекта сценария, сигнализировать Фреймворку о том, что при создании класса в данную переменную требуется передать ссылку на соответствующий объект, предварительно загруженный по сценарию. Данный механизм так же реализован при помощи рефлексии.

* 1. **Компонент “Plugin J3D DT”**

**4.2.1 Диаграмма классов**

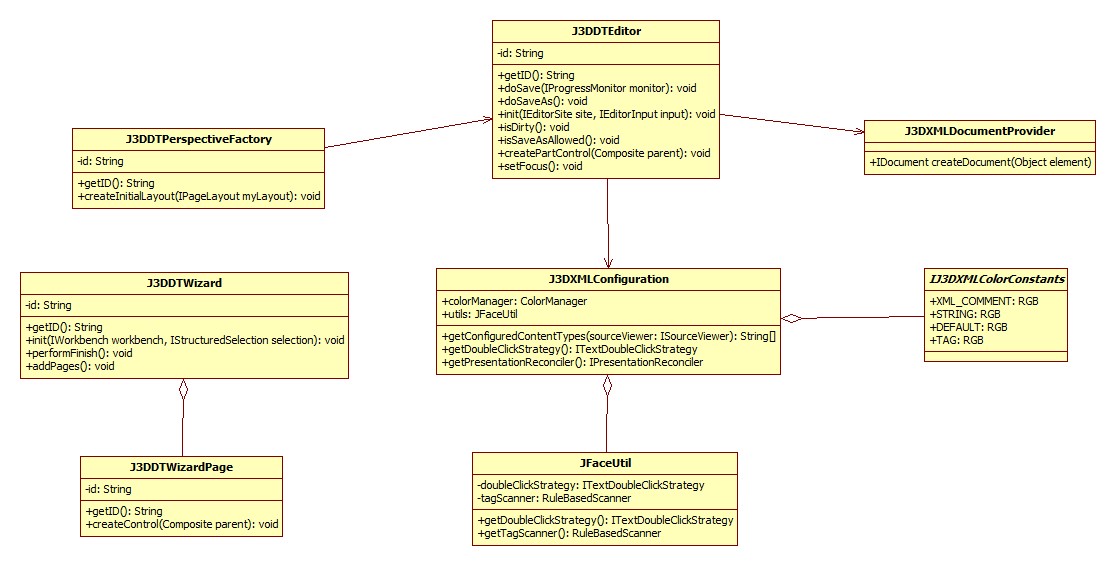


Рисунок 4.2.1.1 Диаграмма классов плагина

|  |  |
| --- | --- |
| **Класс** | **Интерфейс - расширения/Назначение** |
| **J3DDTEditor** | **Реализация - org.eclipse.ui.Editor** |
| **J3DXMLConfiguration** | **Конструирование дизайна приложения. Управление GUI объектами.** |
| **JFaceUtil** | **Набор Вспомогательных JFace - инструментов для редактора. Реализация**  **org.eclipse.jface.text.presentation.**IPresentationReconciler **- отображение документа.**  **org.eclipse.jface.text.presentation.**RuleBasedPartitionScanner - обработка контента документа |
| **IJ3DXMLColorConstants** | **Интерфейс, используемый редактором для реализации подсветки синтаксиса.** |
| **J3DXMLDocumentProvider** | **Реализация JFace IDocumet -Функция создания документа.** |
| **J3DDTWizard** | **Реализация - org.eclipse.ui.newWizard** |
| **J3DDTWizardPage** | **Реализация - org.eclipse.ui.newWizardPage** |
| **J3DDTPerspectiveFactory** | **Реализация - org.eclipse.ui.Perspective** |

Таблица 4.2.1.1 Описание классов плагина

* + 1. **Схема интерфейса**

После старта Eclipse IDE , пользователю станет доступен её GUI (Рисунок 4.2.2.1).

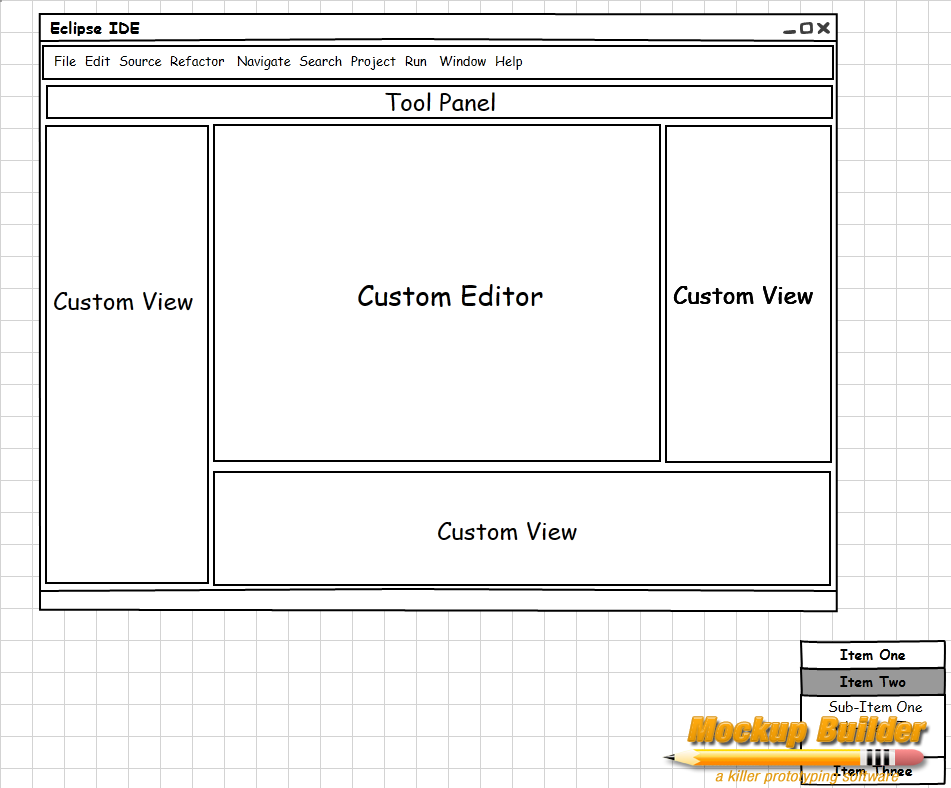


Рисунок 4.2.2.1 Workbench в стандартной компоновке

По желанию, пользователь может самостоятельно изменить перспективу (Рисунок 4.2.2.2).

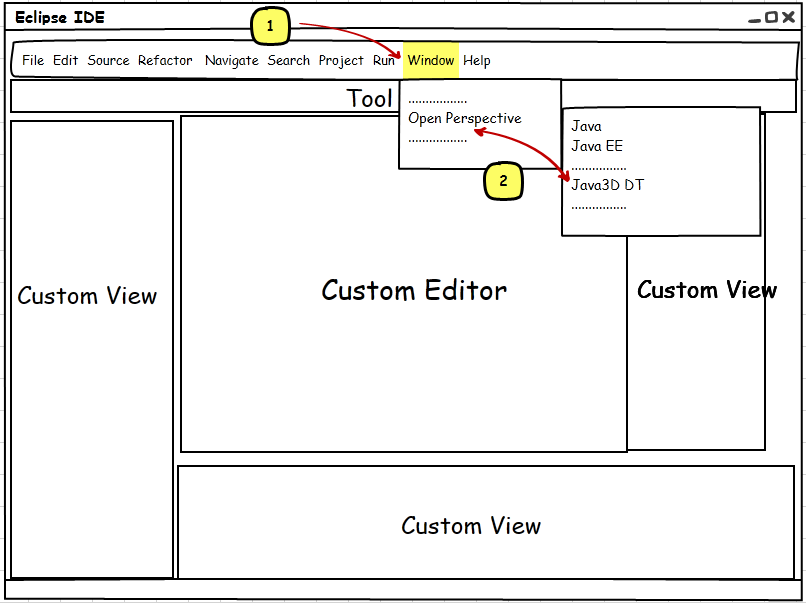


Рисунок 4.2.2.2 Изменение перспективы (Windows -> Open Perspective -> Java3D DT)

Перспектива Java3D DT компонует специализированные элементы как показано на рисунке ниже.

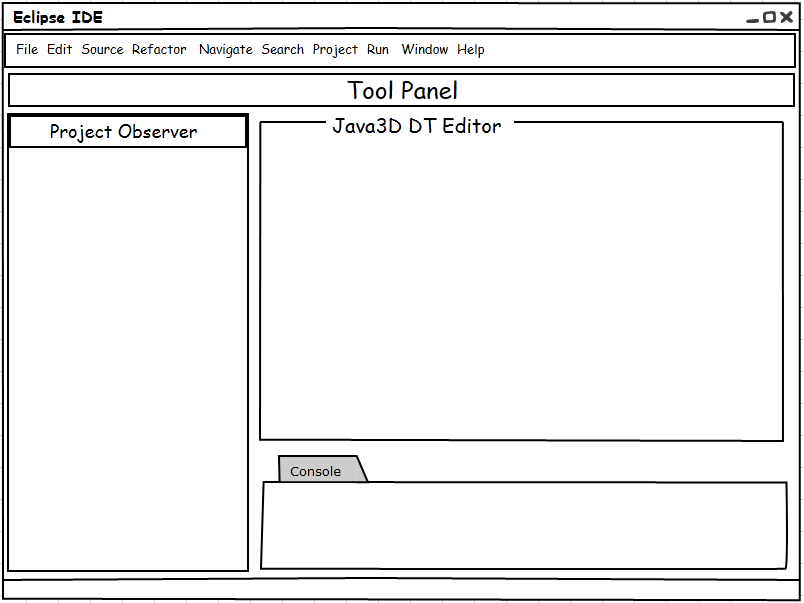


Рисунок 5.2.2.3 Компоновка перспективы J3D DT

Для создания нового сценария Java3D DT пользователю необходимо воспользоваться специализированным мастером (Рисунок 4.2.2.4).

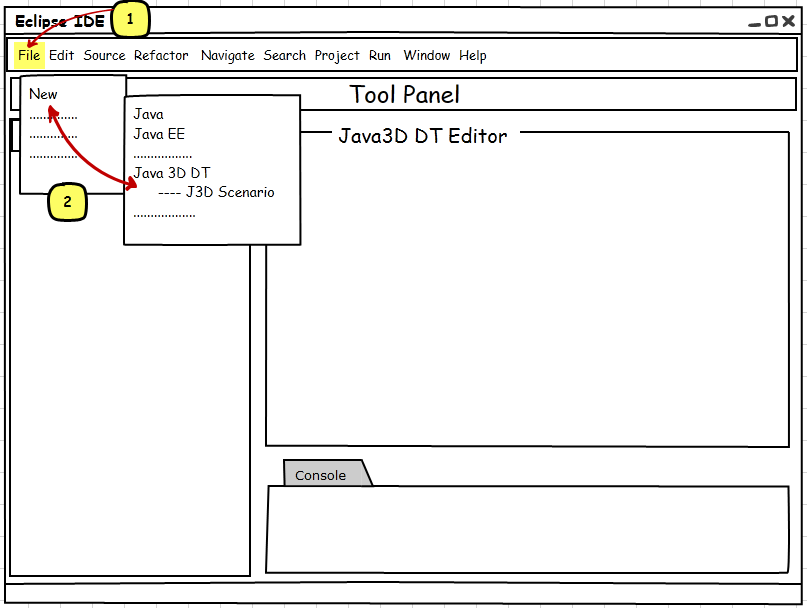


Рисунок 4.2.2.4 Создание JavaFX проекта (File -> New -> JavaFX Project)

В появившемся окне мастера, пользователь должен заполнить всю необходимую информацию о проекте, требуемую для его успешной инициализации, и нажать кнопку «Finish» (Рисунок 4.2.2.5).

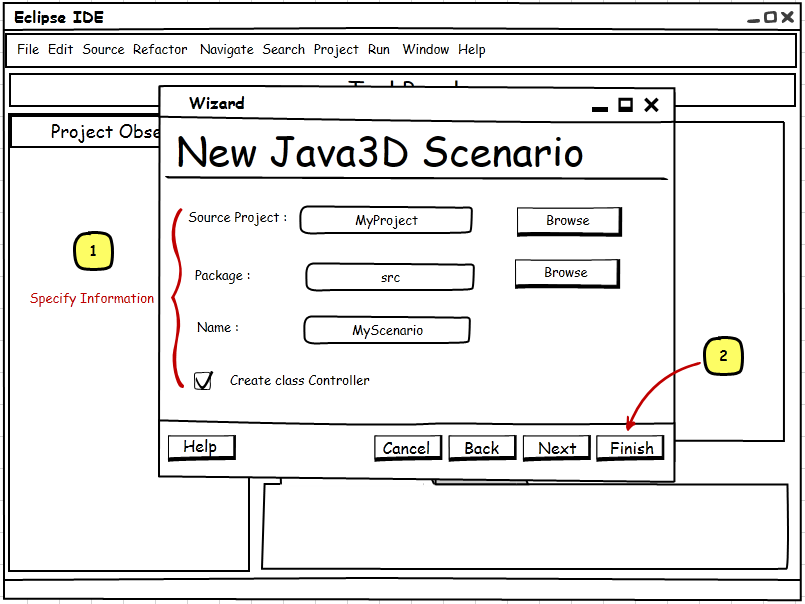


Рисунок 4.2.2.5 Мастер создания нового сценария

После успешной инициализации в рабочем пространстве будут создан файл сценария и класс контроллер (опционально). Также, Eclipse автоматически изменит текущую перспективу на специализированную и добавит в classpath проекта дистрибутив компонента «J3D DT Framework» (Рисунок 4.2.2.6) .

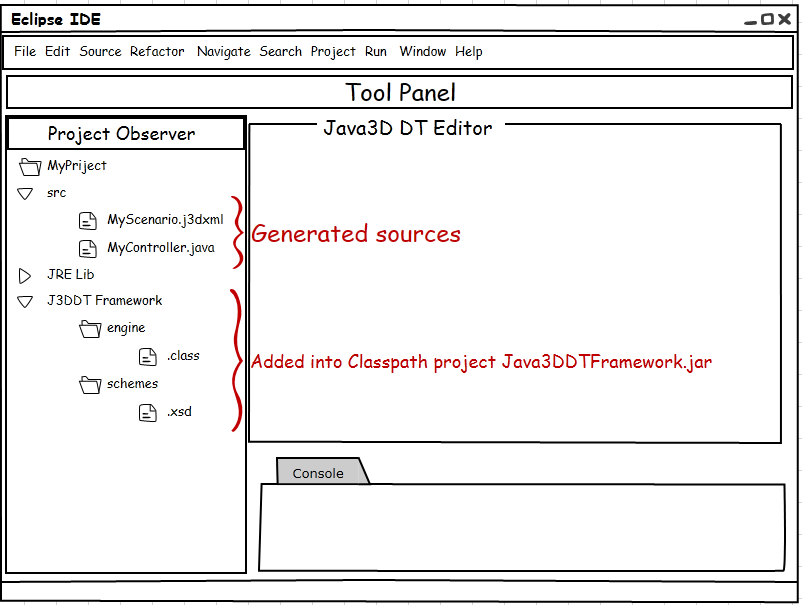


Рисунок 4.2.2.6 Созданные мастером ресурсы

Для управления сценарием, пользователю необходимо дважды кликнуть по его иконке, после чего данный файл будет открыт в специализированном редакторе (Рисунок 4.2.2.7).

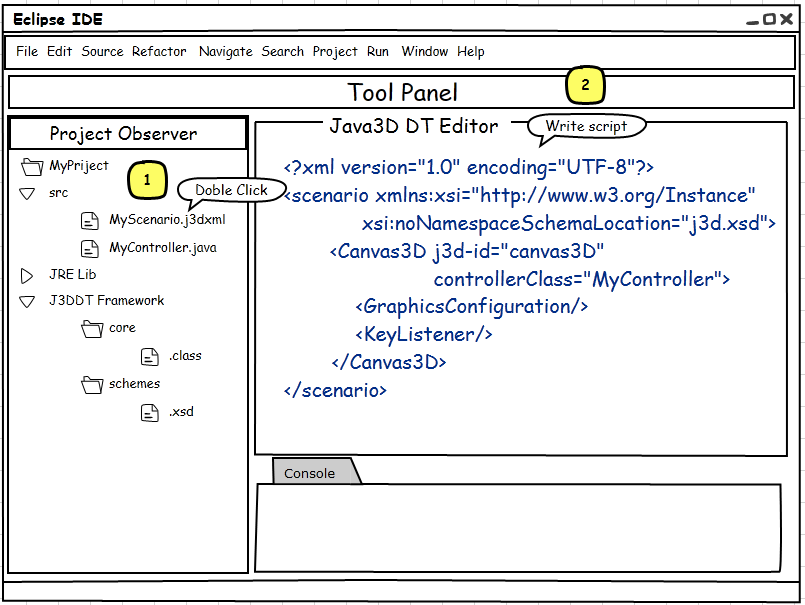
******

Рисунок 4.2.2.7 Отображаемый редактором файл сценария

# ****ЗАКЛЮЧЕНИЕ****

**В ходе проделанной работы, были изучены следующие технологии:**

* **Java-Reflection;**
* **XML-XSD;**
* **Eclipse PDE API;**
* **Java3D API;**
* **CVS Git.**

**Был разработан Фреймворк Java3DDT, который позволяет:**

* **загружать графические объекты по указанным сценариям;**
* проектировать свои приложения на основе концепции MVC.

**Также в рамках поставленной задачи был разработан плагин для Eclipse IDE, который, используя выше описанный модуль, позволяет пользователю следующие возможности по управлению Java3D-сценариями:**

* **добавление с помощью специализированного мастера фалов с уникальным расширением «.j3dxml» в Java-проект;**
* **работа с файлом в режиме активной специализированной перспективы;**
* **редактирование фала с помощью специализированного редактора;**
* **сохранение и удаление файла.**

**На данный момент схема описания сценариев позволяет использовать лишь небольшой спектр объектов платформы Java3D, но так как архитектура разработанного Фреймворка проектировалась с расчётом на будущее расширение функциональности и независимость от свойств самих объектов Java3D API , дополнить спектр необходимыми декларациями объектов не составит труда.**

# ****СПИСОК ЛИТЕРАТУРЫ****

1. . Eric Clayberg, Dan Rubel. Eclipse Plug-ins, 3rd Edition.- Addison-Wesley, 2008. – 832 c.
2. Jeff McAffer, Jean-Michel Lemieux. Eclipse Rich Client Platform,2nd Edition,. 2010. – 534 c.
3. Askar Rahimberdiev. Project Eclipse. -RSDN Magazine #4, 2004. – (<http://www.rsdn.ru/?article/devtools/eclipse_plugins.xml>)
4. Александр Цимбал. Разработка плагинов для Eclipse IDE. -RSDN Magazine #1, 2008. – (http://www.rsdn.ru/?article/devtools/eclipse\_plugins.xml)
5. Eclipse Help. – (http://help.eclipse.org/indigo/index.jsp)
6. Lars Vogel. Plugin & RCP Eclipse Development. –(http://www.vogella.de/eclipse.html)
7. Oracle (SUN). Java3D API Tutorial. – (http://java.sun.com/developer/onlineTraining/java3d)
8. Java3D API Tutorial . - ( http://www.java3d.org/tutorial.html)
9. Александр Печерский. Язык XML – практическое введение. - (http://citforum.ru/internet/xml/index.shtml)
10. MSDN-Microsoft. XML Standards Reference. – (http://msdn.microsoft.com/ru-ru/library/ms256177(v=vs.90).aspx)
11. Oracle. Java TM Tutorials, Reflection API. - (http://docs.oracle.com/javase/tutorial/reflect/index.html)

# ****ПРИЛОЖЕНИЕ 1. J3d.xsd****

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"

elementFormDefault="unqualified" >

<xs:element name="scenario">

<xs:complexType>

<xs:sequence>

<xs:element ref="SimpleUniverse" minOccurs="0" maxOccurs="1" />

<xs:element ref="BranchGroup" minOccurs="0" maxOccurs="1" />

<xs:element ref="ColorCube" minOccurs="0" maxOccurs="1" />

<xs:element ref="Cylinder" minOccurs="0" maxOccurs="1" />

<xs:element ref="Canvas3D" minOccurs="0" maxOccurs="1" />

<xs:element ref="Transform3D" minOccurs="0" maxOccurs="1" />

<xs:element ref="TransformGroup" minOccurs="0" maxOccurs="1" />

<xs:element ref="BoundingSphere" minOccurs="0" maxOccurs="1" />

<xs:element ref="DirectionalLight" minOccurs="0" maxOccurs="1" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="SimpleUniverse">

<xs:complexType>

<xs:sequence>

<xs:element ref="BranchGroup" minOccurs="0" maxOccurs="1" />

<xs:element ref="Canvas3D" minOccurs="0" maxOccurs="1" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="Canvas3D">

<xs:complexType>

<xs:sequence>

<xs:element ref="GraphicsConfiguration" minOccurs="1" maxOccurs="1"/>

<xs:element ref="KeyListener" minOccurs="0" maxOccurs="1"/>

<xs:element ref="ActionListener" minOccurs="0" maxOccurs="1"/>

<xs:element ref="MouseListener" minOccurs="0" maxOccurs="1"/>

</xs:sequence>

<xs:attribute name="j3d-id"/>

<xs:attribute name="controllerClass"/>

</xs:complexType>

</xs:element>

<xs:element name="GraphicsConfiguration">

<xs:complexType>

</xs:complexType>

</xs:element>

<xs:element name="KeyListener">

<xs:complexType>

<xs:attribute name="refKeyTyped"/>

<xs:attribute name="refKeyPressed"/>

<xs:attribute name="refKeyReleased"/>

</xs:complexType>

</xs:element>

<xs:element name="ActionListener">

<xs:complexType>

<xs:attribute name="actionReference"/>

</xs:complexType>

</xs:element>

<xs:element name="MouseListener">

<xs:complexType>

<xs:attribute name="actionReference"/>

</xs:complexType>

</xs:element>

<xs:element name="BranchGroup">

<xs:complexType>

<xs:sequence>

<xs:element ref="children" minOccurs="1" maxOccurs="1" />

</xs:sequence>

<xs:attribute name="j3d-id"/>

</xs:complexType>

</xs:element>

<xs:element name="children">

<xs:complexType>

<xs:sequence>

<xs:element ref="ColorCube" minOccurs="0" maxOccurs="unbounded" />

<xs:element ref="Cylinder" minOccurs="0" maxOccurs="unbounded" />

<xs:element ref="Sphere" minOccurs="0" maxOccurs="unbounded" />

<xs:element ref="TransformGroup" minOccurs="0" maxOccurs="unbounded" />

<xs:element ref="DirectionalLight" minOccurs="0" maxOccurs="unbounded" />

<xs:element ref="AmbientLight" minOccurs="0" maxOccurs="unbounded" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="TransformGroup">

<xs:complexType>

<xs:sequence>

<xs:element ref="writeCapability" minOccurs="0" maxOccurs="1" />

<xs:element ref="Capability" minOccurs="0" maxOccurs="1" />

<xs:element ref="Transform3D" minOccurs="0" maxOccurs="1" />

<xs:element ref="children" minOccurs="0" maxOccurs="1" />

</xs:sequence>

<xs:attribute name="j3d-id"/>

</xs:complexType>

</xs:element>

<xs:element name="Transform3D">

<xs:complexType>

<xs:sequence>

<xs:element ref="Translation" minOccurs="1" maxOccurs="1" />

</xs:sequence>

<xs:attribute name="j3d-id"/>

</xs:complexType>

</xs:element>

<xs:element name="Capability">

<xs:complexType>

<xs:attribute name="value"/>

</xs:complexType>

</xs:element>

<xs:element name="writeCapability">

<xs:complexType>

</xs:complexType>

</xs:element>

<xs:element name="Translation">

<xs:complexType>

<xs:sequence>

<xs:element ref="Vector3f" minOccurs="1" maxOccurs="1" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="Vector3f">

<xs:complexType>

<xs:sequence>

</xs:sequence>

<xs:attribute name="x"/>

<xs:attribute name="y"/>

<xs:attribute name="z"/>

<xs:attribute name="j3d-id"/>

</xs:complexType>

</xs:element>

<xs:element name="ColorCube">

<xs:complexType>

<xs:attribute name="j3d-id"/>

<xs:attribute name="scale" type="xs:double"/>

</xs:complexType>

</xs:element>

<xs:element name="Cylinder">

<xs:complexType>

<xs:attribute name="j3d-id"/>

<xs:attribute name="scale" type="xs:double"/>

</xs:complexType>

</xs:element>

<xs:element name="Sphere">

<xs:complexType>

<xs:attribute name="j3d-id"/>

<xs:attribute name="radius" type="xs:float"/>

</xs:complexType>

</xs:element>

<xs:element name="DirectionalLight">

<xs:complexType>

<xs:sequence>

<xs:element ref="Color3f" minOccurs="1" maxOccurs="1"/>

<xs:element ref="Vector3f" minOccurs="1" maxOccurs="1"/>

<xs:element ref="BoundingSphere" minOccurs="1" maxOccurs="1"/>

</xs:sequence>

<xs:attribute name="j3d-id"/>

</xs:complexType>

</xs:element>

<xs:element name="AmbientLight">

<xs:complexType>

<xs:sequence>

<xs:element ref="Color3f" minOccurs="1" maxOccurs="1"/>

<xs:element ref="BoundingSphere" minOccurs="1" maxOccurs="1"/>

</xs:sequence>

<xs:attribute name="j3d-id"/>

</xs:complexType>

</xs:element>

<xs:element name="Color3f">

<xs:complexType>

<xs:attribute name="r"/>

<xs:attribute name="g"/>

<xs:attribute name="b"/>

</xs:complexType>

</xs:element>

<xs:element name="BoundingSphere">

<xs:complexType>

<xs:sequence>

<xs:element ref="Point3d" minOccurs="0" maxOccurs="1" />

</xs:sequence>

<xs:attribute name="j3d-id"/>

<xs:attribute name="radius"/>

</xs:complexType>

</xs:element>

<xs:element name="Point3d">

<xs:complexType>

<xs:attribute name="j3d-id"/>

<xs:attribute name="x"/>

<xs:attribute name="y"/>

<xs:attribute name="z"/>

</xs:complexType>

</xs:element>

</xs:schema>

# ****ПРИЛОЖЕНИЕ 2. Config.xsd****

<?xml version="1.0" encoding="UTF-8"?>

<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"

elementFormDefault="qualified">

<xs:element name="configurations">

<xs:complexType>

<xs:sequence>

<xs:element ref="config-element" minOccurs="1" maxOccurs="unbounded" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="config-element">

<xs:complexType>

<xs:sequence>

<xs:element ref="type" minOccurs="1" maxOccurs="1" />

</xs:sequence>

<xs:attribute name="elementName" />

</xs:complexType>

</xs:element>

<xs:element name="type">

<xs:complexType>

<xs:sequence>

<xs:choice>

<xs:element ref="class" minOccurs="0" maxOccurs="1" />

<xs:element ref="static-property" minOccurs="0" maxOccurs="1" />

<xs:element ref="setter" minOccurs="0" maxOccurs="1" />

<xs:element ref="union" minOccurs="0" maxOccurs="1" />

<xs:element ref="controller" minOccurs="0" maxOccurs="1" />

</xs:choice>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="class">

<xs:complexType>

<xs:sequence>

<xs:element ref="constructor" minOccurs="1" maxOccurs="unbounded" />

<xs:element ref="set-propetry" minOccurs="0" maxOccurs="unbounded" />

<xs:element ref="setter" minOccurs="0" maxOccurs="unbounded" />

<xs:element ref="union-propetry" minOccurs="0" maxOccurs="unbounded" />

<xs:element ref="controller" minOccurs="0" maxOccurs="unbounded" />

</xs:sequence>

<xs:attribute name="nameSource" />

</xs:complexType>

</xs:element>

<xs:element name="static-property">

<xs:complexType>

<xs:sequence>

<xs:element ref="static-field" minOccurs="0" maxOccurs="1"/>

<xs:element ref="static-getter" minOccurs="0" maxOccurs="1"/>

<xs:element ref="static-setter" minOccurs="0" maxOccurs="1"/>

</xs:sequence>

<xs:attribute name="nameSource" />

</xs:complexType>

</xs:element>

<xs:element name="setter">

<xs:complexType>

<xs:sequence>

<xs:element ref="handler" minOccurs="1" maxOccurs="1" />

</xs:sequence>

<xs:attribute name="nameOfElement" />

<xs:attribute name="typeOfElement" />

</xs:complexType>

</xs:element>

<xs:element name="union">

<xs:complexType>

<xs:sequence>

<xs:element ref="handler" minOccurs="1" maxOccurs="1" />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="controller">

<xs:complexType>

<xs:attribute name ="nameArgType"/>

<xs:attribute name ="nameOfElement"/>

<xs:attribute name ="nameOfAction"/>

</xs:complexType>

</xs:element>

<xs:element name="handler">

<xs:complexType>

<xs:sequence>

<xs:element ref = "property" minOccurs="0" maxOccurs="unbounded"/>

</xs:sequence>

<xs:attribute name="handlerName" />

<xs:attribute name="returnType" />

</xs:complexType>

</xs:element>

<xs:element name="static-field">

<xs:complexType>

<xs:attribute name="nameOfField" />

<xs:attribute name="sourceType" />

</xs:complexType>

</xs:element>

<xs:element name="static-getter">

<xs:complexType>

<xs:attribute name="nameOfMethod" />

<xs:attribute name="returnType" />

</xs:complexType>

</xs:element>

<xs:element name="static-setter">

<xs:complexType>

<xs:attribute name="nameOfMethod" />

<xs:attribute name="argumentType" />

</xs:complexType>

</xs:element>

<xs:element name="bind-property">

<xs:complexType>

<xs:attribute name="properyName" />

</xs:complexType>

</xs:element>

<xs:element name="property">

<xs:complexType>

<xs:attribute name="properyType"/>

<xs:attribute name="nameOfElement" />

</xs:complexType>

</xs:element>

<xs:element name="set-propetry">

<xs:complexType>

<xs:sequence>

<xs:element ref="handler" minOccurs="1" maxOccurs="1" />

</xs:sequence>

<xs:attribute name="nameOfElement" />

</xs:complexType>

</xs:element>

<xs:element name="union-propetry">

<xs:complexType>

<xs:sequence>

<xs:element ref="handler" minOccurs="1" maxOccurs="1" />

</xs:sequence>

<xs:attribute name="nameOfElement" />

<xs:attribute name="sourceType" />

</xs:complexType>

</xs:element>

<xs:element name="constructor">

<xs:complexType>

<xs:sequence>

<xs:element ref="property" minOccurs="0" maxOccurs="unbounded"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

# ****ПРИЛОЖЕНИЕ 3. Manifest.mf****

Manifest-Version: 1.0

Bundle-ManifestVersion: 2

Bundle-Name: PluginJ3DDevT

Bundle-SymbolicName: PluginJ3DDevT; singleton:=true

Bundle-Version: 1.0.0.qualifier

Bundle-Activator: pluginj3ddevt.Activator

Bundle-Vendor: Ivan Petrushin

Require-Bundle: org.eclipse.ui,

org.eclipse.core.runtime,

org.eclipse.core.expressions;bundle-version="3.4.300",

org.eclipse.core.resources;bundle-version="3.7.101",

org.eclipse.jdt.core;bundle-version="3.7.3",

org.eclipse.ui.editors;bundle-version="3.7.0",

org.eclipse.ui.ide;bundle-version="3.7.0",

org.eclipse.jface.text;bundle-version="3.7.2",

org.eclipse.jdt.ui;bundle-version="3.7.2",

org.eclipse.ui.console;bundle-version="3.5.100"

Bundle-RequiredExecutionEnvironment: JavaSE-1.7

Bundle-ActivationPolicy: lazy

# ****ПРИЛОЖЕНИЕ 4.Plugin.xml****

<?xml version="1.0" encoding="UTF-8"?>

<?eclipse version="3.4"?>

<plugin>

<extension

point="org.eclipse.ui.editors">

<editor

class="edu.vsu.cs.petrushin\_i\_a.j3ddtplugin.editors.j3dxml\_editor.J3DXMLEditor"

contributorClass="org.eclipse.ui.texteditor.BasicTextEditorActionContributor"

default="false"

extensions="j3dxml"

icon="icons/sample.gif"

id="PluginJ3DDevT.editorJ3DXML"

name="J3DXML Editor">

</editor>

</extension>

<extension

point="org.eclipse.ui.newWizards">

<category

id="PluginJ3DDevT.category"

name="Java3D">

</category>

<wizard

category="PluginJ3DDevT.category"

class="edu.vsu.cs.petrushin\_i\_a.j3ddtplugin.wizards.create\_scenario.J3DDTWizard"

finalPerspective="PluginJ3DDevT.perspective"

icon="icons/sample.gif"

id="PluginJ3DDevT.wizardJ3DDT"

name="Java3D Scenario Wizard"

preferredPerspectives="PluginJ3DDevT.perspective">

</wizard>

</extension>

<extension

point="org.eclipse.ui.perspectives">

<perspective

class="edu.vsu.cs.petrushin\_i\_a.j3ddtplugin.perspective.J3DDTPerspectiveFactory"

icon="icons/sample.gif"

id="PluginJ3DDevT.perspective"

name="Java3DXML Perspective">

</perspective>

</extension>

</plugin>