**Левин В.Е., Шульга Т.Э.**

**Создание и настройка среды для работы с онтологиями используя фреймворк Jena**

**Учебное пособие**

# Введение

После подготовительной работы, когда Вы спроектировали схему онтологии и создали её, возникает вопрос – «Что же дальше?». Вопрос правильный, ведь для последующей работы с онтологией её надо где-то хранить, куда-то надо слать запросы на выборку, на добавление. А онтология – вещь очень гибкая и, если вы планируете расширение в будущем, могут потребоваться программы-черви, которые будут заполнять онтологию и связывать её с другими.

Средства для подобного уже существуют, и Вы способны самостоятельно создать окружение для Вашей онтологии используя язык Java и библиотеку Jena.

Созданию такого окружения, разделённого на сервер, хранящий онтологию, а также веб-интерфейс для выборки данных, и посвящено это учебное пособие.

# Необходимые инструменты

Для создания среды потребуется следующие инструменты:

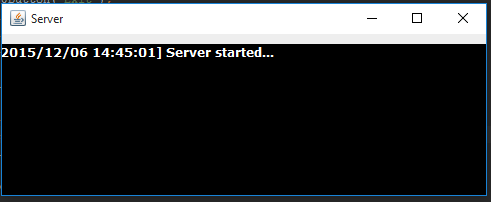
1. **Среда разработки Java приложений:** Eclipse, Intellij IDEA или NetBeans (в данном пособии используется Intellij IDEA)
2. **Runtime среда:** Java(<http://www.java.com/ru/>) и Java Development Kit (<http://www.oracle.com/technetwork/java/javase/downloads/index.html>)
3. **Фреймворк для работы с онтологиями:** для языка Java имеется Jena (<https://jena.apache.org/documentation/ontology/>)
4. **Вспомогательные библиотеки для облегчения и ускорения работы:**

* Simple-JSON (<https://github.com/fangyidong/json-simple>) – Легковесная и мощная библиотека для форматирования данных в формат JSON
* opa-core (<https://github.com/opaoz/opa-core>) – Библиотека с функциями для упрощения разворачивания клиент-серверного приложения и работы с Jena

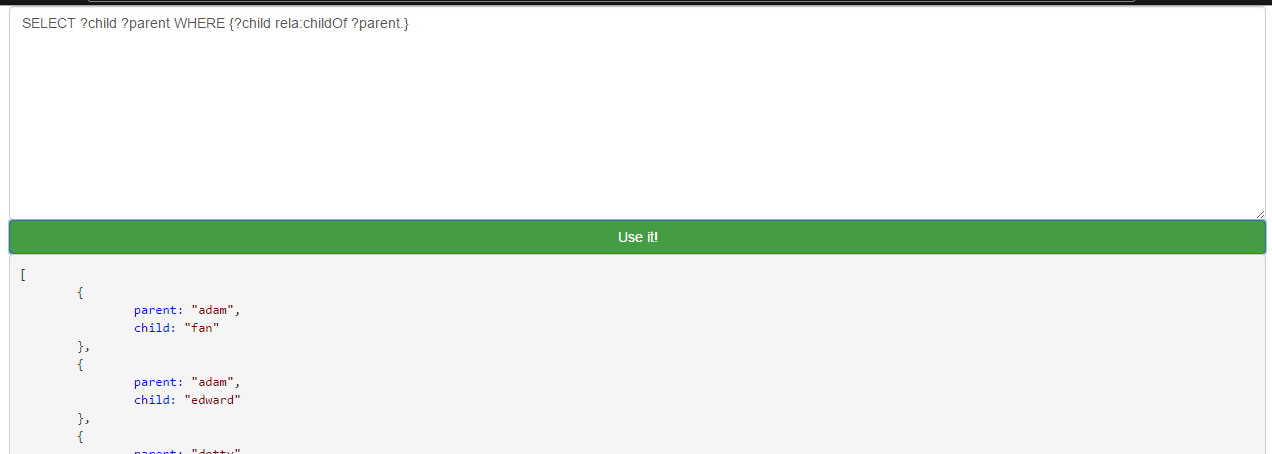
# Схема разрабатываемого приложения

Среда будет состоять из двух частей:

**Сервер**. Часть, написанная на языке Java, которая хранит в себе онтологию, обрабатывает запросы и производит выборку и добавление данных.



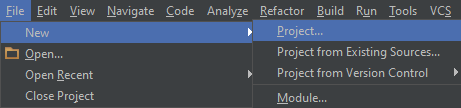
**Клиент**. Веб-интерфейс для приложения, позволяющий формировать и отсылать запросы к серверу, а также принимать результаты и выводить их в качестве наглядных данных.



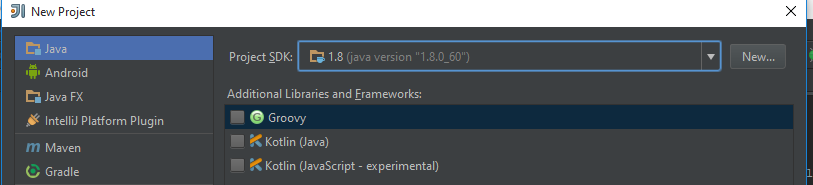
# Разработка

## Создание проекта и подключение требуемых библиотек

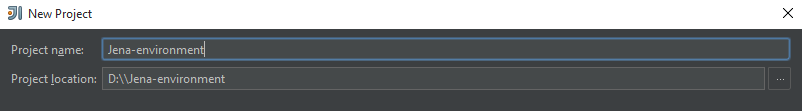
Командой меню «File > New > Project…» открываем диалоговое окно с выбором типа создаваемого проекта



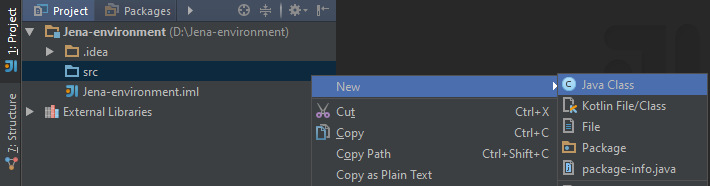
Выбираем тип проекта - «Java» и версию SDK из выпадающего списка и жмём кнопку «Next»

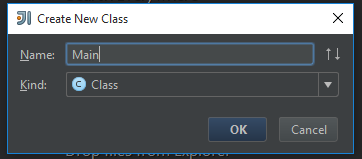


Пропускаем следующий экран с выбором темплейта, затем вводим название и расположение проекта и жмём «Finish»



Вот мы и создали проект, теперь добавим главный файл, настроим параметры запуска и выведем в консоль заветное «Hello, World!».

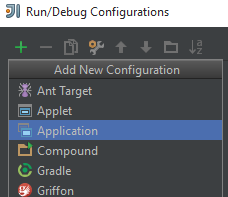




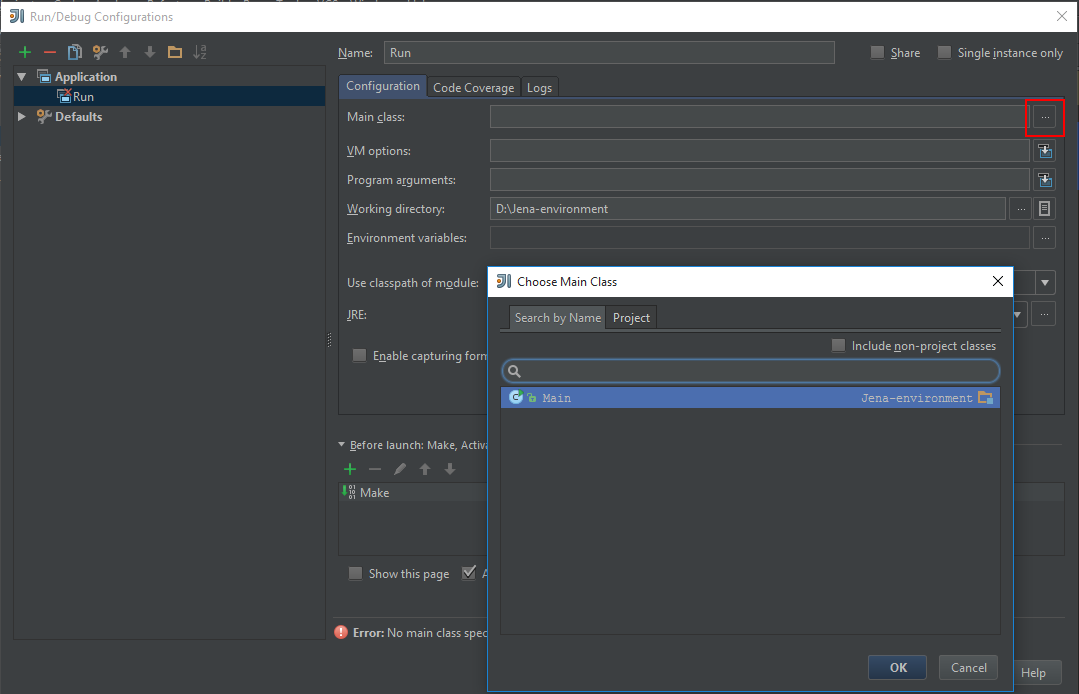
Содержимое класса:

public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Hello, world!");  
 }  
}

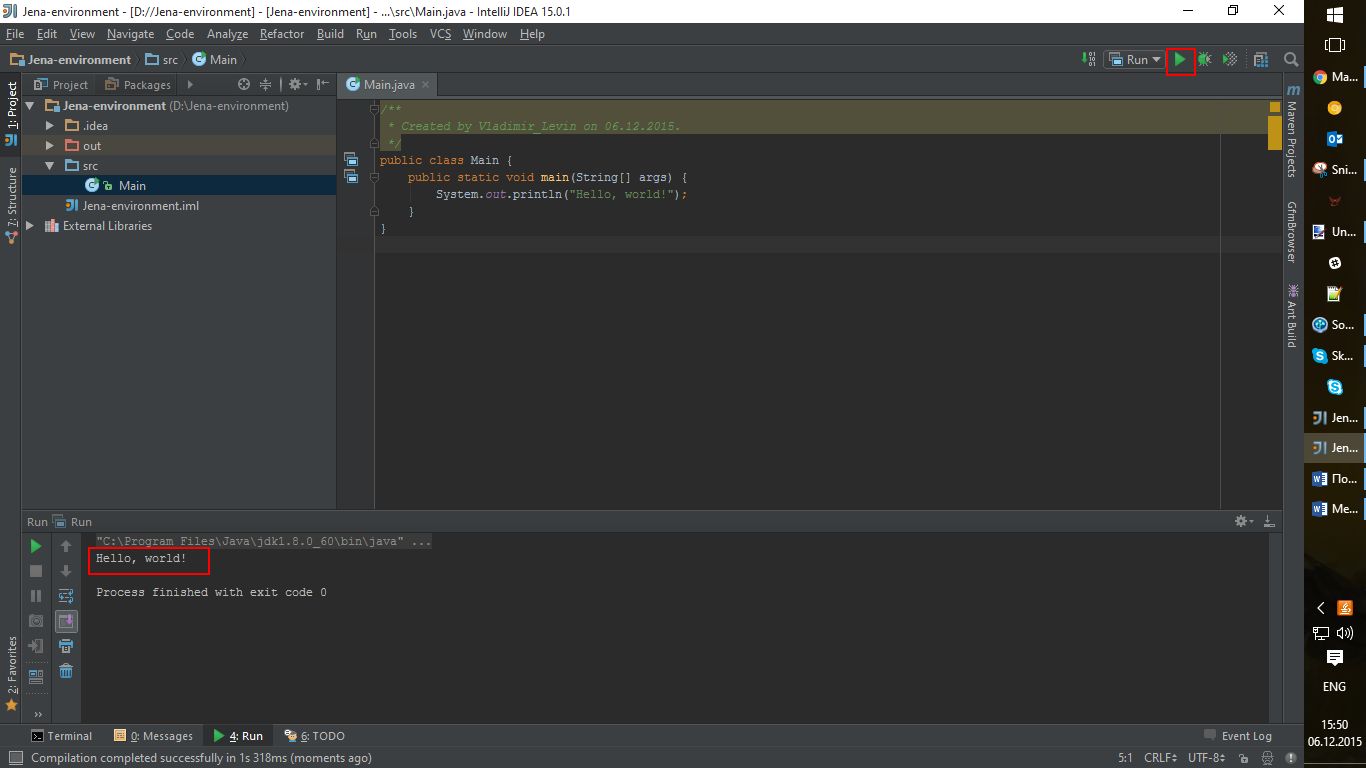
Далее настроим конфигурацию запуска. Для этого жмём выпадающее меню справа сверху и выбираем пункт «Edit Configurations…» , затем зелёный «+» и выбираем «Application»



Нажимаем «…» возле поля «Main class» и выбираем ново созданный класс Main



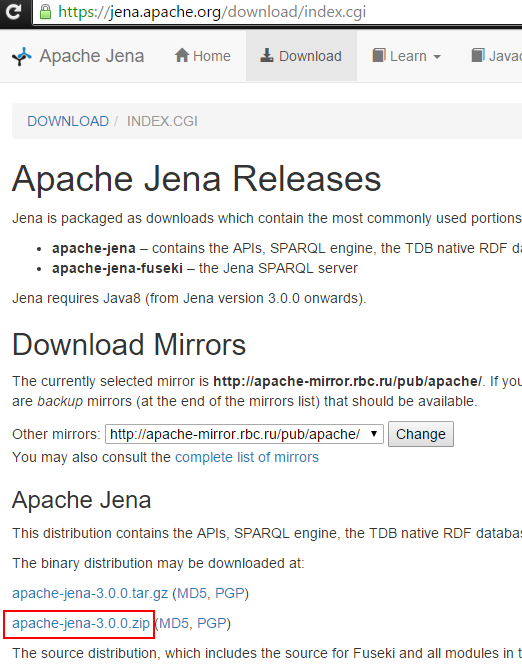
Теперь, по нажатию кнопки «Run» наш проект запустится



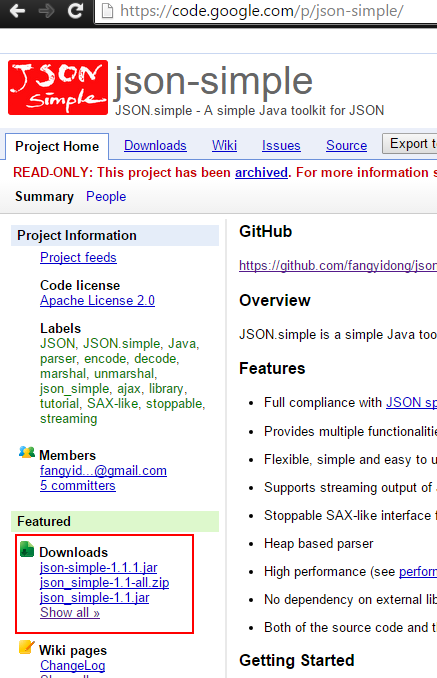
## Подключение библиотек

Скачиваем все библиотеки:

1. Jena - <https://jena.apache.org/download/index.cgi>

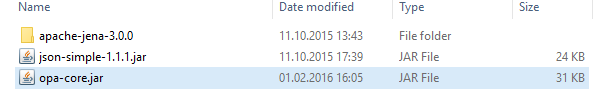


1. Simple-JSON - <https://code.google.com/p/json-simple/>

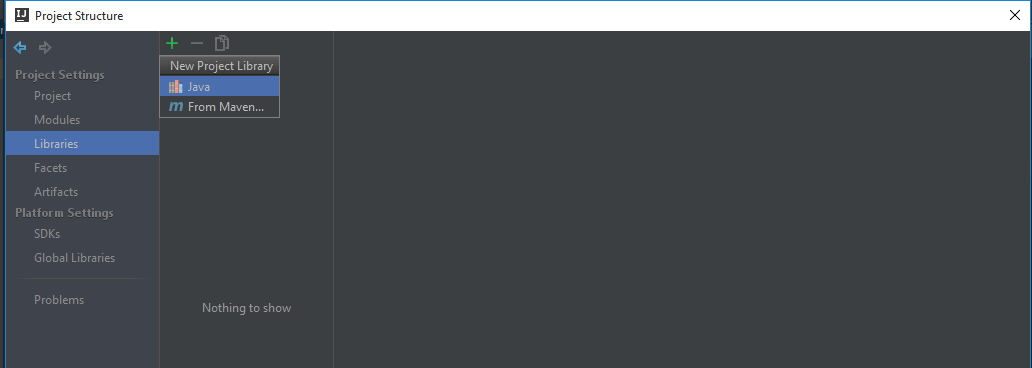


1. opa-core - <https://github.com/opaoz/opa-core/blob/master/dist/opa-core.jar?raw=true>

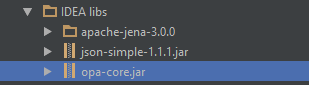
Складываем их в одну папку



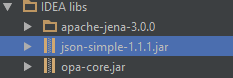
Открываем «File -> Project Structure -> Libraries»



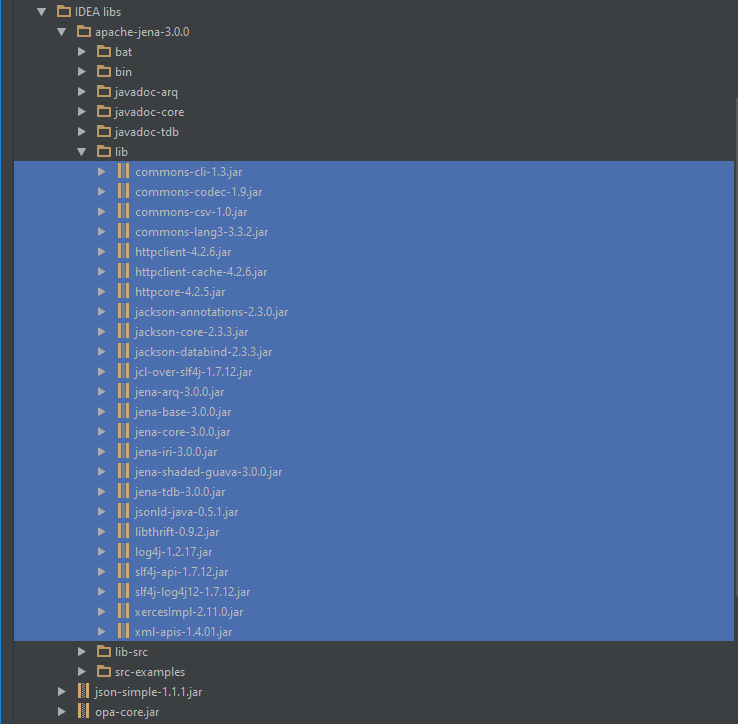
42Нажимаем на «+» и выбираем «Java», затем выбираем первый файл



Повторяем для второго



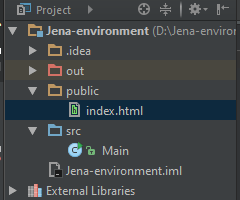
А для apache-jena выбираем всё в папке “lib”



## Структура проекта

Так как разрабатываемый проект является «клиент-серверным», возведём соответствующую структуру папок.

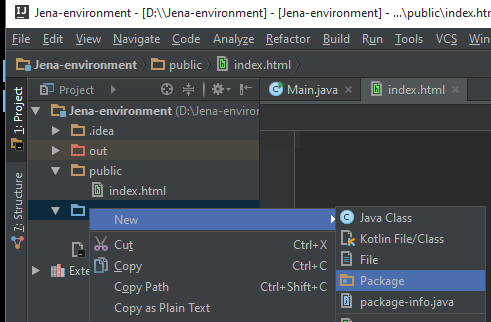
Сначала создаём папку “public” в корне проекта. Здесь будет находится вся клиентская часть проекта. Чтобы обозначить это, создадим файл “index.html”.



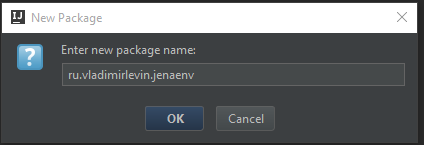
Вся серверная часть будет располагаться в папке “src”. Внутри будет возводится структура из пэкэджей ([*Package (Java)*](https://ru.wikipedia.org/wiki/Package_(Java))*)* в соответствии с популярным [стайлгайдом](https://docs.oracle.com/javase/tutorial/java/package/namingpkgs.html). Суть его состоит в том, что имя корневого пакета составляется из 3 частей:

1. **Домен** (org, ru, com, ua)
2. **Автор**
3. **Название проекта**

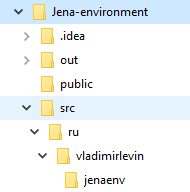
В итоге получаем *«ru.vladimirlevin.jenaenv»*



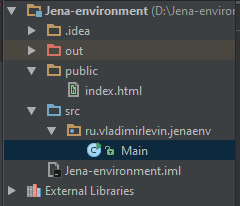
При создании можно писать наименование сразу с «.»



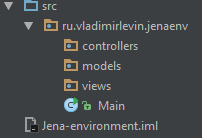
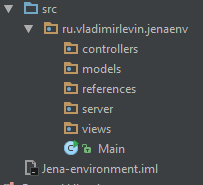
В итоге создастся соответствующая структура папок:



В созданный package и перенесём наш класс “Main”



Продолжим структурирование проекта, добавим MVC составляющую и пару вспомогательных пакетов

Суть такого подхода состоит в следующем:

1. **Controllers** – папка, содержащая логику конкретных запросов к серверу.
2. **Models** – хранилище RDF-моделей
3. **References** – папка с константами, используемыми в проекте
4. **Server** – двигатель приложения, обработка всех входящих запросов и подключений
5. **Views** – UI-компоненты, для визуализации действий сервера.
6. **Main**.java – стартер проекта