Серверное программирование

- Серверная часть приложения может быть создана с помощью разных программных технологий. Рассмотрим, как создаются веб-приложения на платформе Java. Платформа Java делится на несколько компонентов, основными из которых являются SE (Standard Edition) и EE (Enterprise Edition). До сих пор вы работали в пределах платформы SE

Основные компоненты Java EE

- Для обработки клиентских HTTP запросов Java EE предлагает такой API, как Servlet. Servlet — это Java класс, умеющий принимать клиентские запросы (request), обрабатывать их и отправлять клиенту ответы (response). Servlet разворачивается и выполняется на сервере приложений — например, на Tomcat или GlassFish.
- Для создания интерактивных, динамических html страниц используется API Java Server Pages (JSP). Это своего рода серверный язык Java. JSP используется совместно с Servlet.

- Для создания веб-сервисов Java EE предлагает технологию Enterprise JavaBeans (EJB). Если вам понятно утверждение, что обычное Java приложение состоит из классов, то, по аналогии, можете сейчас запомнить, что веб-сервис состоит из EJB компонентов, которые очень напоминают классы, но более приспособлены для веб-сервисов и потому обладают большим числом преимуществ. EJB изначально поддерживают удаленный доступ, являются многопоточными, поддерживают транзакции, обеспечивают целостность данных и обладают еще рядом полезных свойств.

- ■ Java EE также предлагает собственные API для работы с источниками данных, для работы с форматом JSON, для обработки электронных писем, для работы с REST-службами и другие. О некоторых из этих компонетов мы также поговорим в наших уроках.

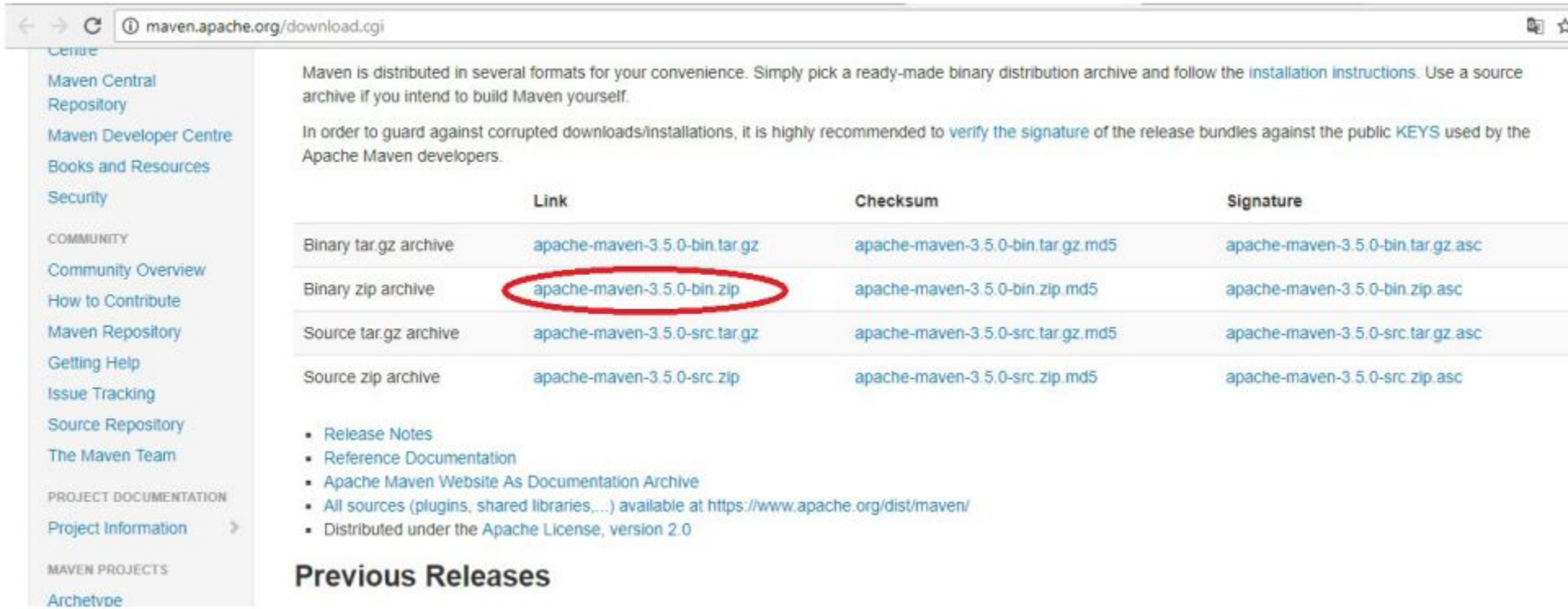
Фреймворки и библиотеки

Maven

- Из каких этапов состоит создание проекта? Это компиляция исходных файлов, создание jar-файла, дистрибутива и документации. Вообще говоря, это большой объем работы, особенно в том случае, когда собираемый проект состоит из большого числа файлов. Эта работа выпоняется специальными инструментами. Сейчас мы с вами рассмотрим один из них — утилиту Maven

- Создание Java проекта с помощью Maven выполняется на основании сценария, записанного в специальном xml файле. Этот сценарий не зависит от платформы и позволяет собирать проект в ОС Windows или в ОС Linux без изменения файла сценария. Как правило, создание проекта требует подключения различных сторонних библиотек. В такой ситуации необходимо учитывать, что разные версии таких библиотек могут конфликтовать друг с другом

Для установки Maven перейдите на страницу http://maven.apache.org/download.cgi и загрузите дистрибутив:



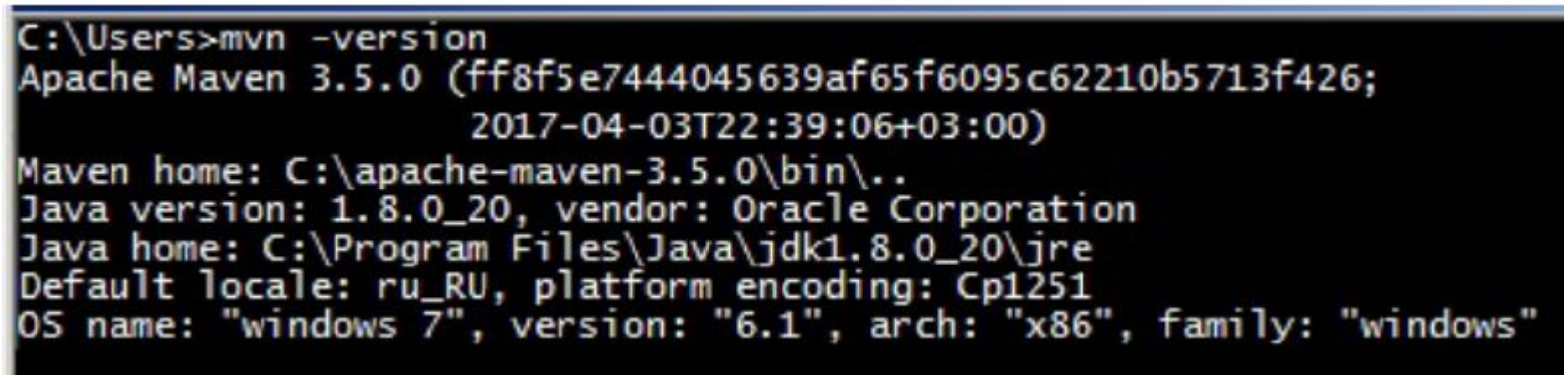
- Теперь распакуйте zip архив в любую директорию на своем диске — например, прямо на системный диск по пути C:\apache-maven-3.5.0. Далее нажмите Win+Pause, перейдите в «Дополнительные параметры» и в появившемся окне кликните внизу кнопку «Переменные среды». Затем в окне «Системные переменные» создайте следующие переменные с указанными значениями:

- переменную M2_HOME со значением пути, куда вы скопировали Maven; в моем случае — C:\apachemaven-3.5.0;
- переменную M2 со значением %M2_HOME%\bin;
- переменную MAVEN_OPTS со значением -Xms256m или – Xms512m;
- переменную Path со значением %M2%.

- Теперь проверьте, существует ли у вас переменная с именем JAVA_HOME и со значением пути, по которому установлен JDK. В моем случае это путь C:\Program Files\ Java\jdk1.8.0_25. Если этой переменной нет — создайте ее. Перегрузите компьютер.

Выполните в консольном окне команду:

```
mvn -version
```



- Теперь можно перейти к сборке проекта. Создайте где-нибудь папку, например, с названием test, перейдите в нее в консольном окне и выполните такую команду:

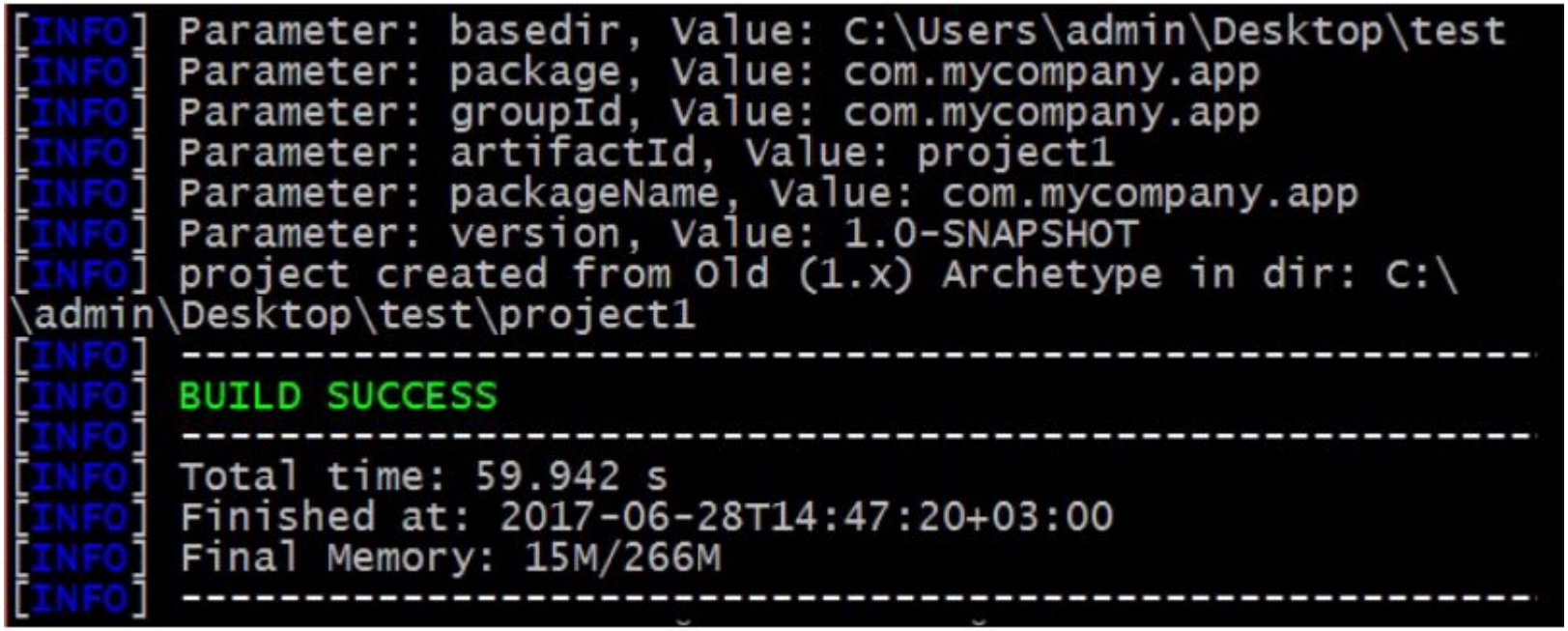
```
mvn archetype:generate -DgroupId=com.mycompany.app -
DartifactId=project1 -DarchetypeArtifactId=maven-
archetype-quickstart -DinteractiveMode=false
```

- Самый первый запуск Maven может занять много времени, так как в этот момент выполняется подгрузка актуальных версий всех необходимых инструментов. Надо немного подождать. Кроме того, если вы выходите в Интернет через прокси, вы должны указать это в настройках конфигурации Maven. В этом случае откройте файл {M2_HOME}/conf/settings.xml и внесите в него информацию для прохождения прокси. Найдите в этом файле раздел proxy, раскомментируйте его и приведите к такому виду:

```
<proxies>
    <!-- proxy
     | Specification for one proxy, to be used in
     | connecting to the network.
     | -->
    <proxy>
      <id>optional</id>
      <active>true</active>
      <protocol>http</protocol>
      <username>proxy_user</username>
      <password>proxy_pass</password>
      <host>10.3.0.3</host>
      <port>3838</port>
      <nonProxyHosts>local.net|some.host.com</
                     nonProxyHosts>
    </proxy>
</proxies>
```

- Вместо proxy_user и proxy_pass надо указать пользователя и пароль для прохождения прокси, а вместо 10.3.0.3 и 3838 — адрес и номер порта прокси-сервера.

После успешного завершения этой команды вы увидите в своем консольном окне нечто такое:



- После выполнения этой команды в папке test будет создана папка project1, в соответствии со значением, указанным в командной строке в атрибуте DartifactId. В этой папке и будет находиться создаваемый проект. Вы увидите там две папки с именами src и target, в которых будут располагаться файлы с исходными кодами для проекта и файлы с unit тестами, соответственно. Создание проекта будет выполняться в соответствии со сценарием, записанным в файле pom.xml и расположенном непосредственно в папке project1. Изначально pom. xml выглядит так:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/
              XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/
              POM/4.0.0
    http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.app</groupId>
    <artifactId>project1</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>project1</name>
    <url>http://maven.apache.org</url>
    <dependencies>
       <dependency>
          <groupId>junit</groupId>
          <artifactId>junit</artifactId>
          <version>3.8.1</version>
          <scope>test</scope>
       </dependency>
    </dependencies>
</project>
```

- Элемент groupId, как правило, задает доменное имя для проекта, например com.mycompany.app. Элементы artifactId и version формируют полное название файла проекта, а элемент packaging — его расширение. В нашем случае (а мы используем значения этих элементов по умолчанию, кроме artifactId) файл проекта получит имя project1-1.0-SNAPSHOT.jar.

Заготовка главного класса созданного приложения выглядит так:

```
package com.mycompany.app;
/*
 * Hello world!
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

Теперь для сборки проекта в консольном окне надо выполнить команду:

```
mvn clean package
```

- Вообще говоря, эта команда выполняет сразу два действия: удаляет компоненты создаваемого проекта, если они уже создавались ранее (clean), и создает проект (package). Если вы создаете проект в первый раз, команду clean можно не указывать. В папке target теперь должен располагаться созданный jar файл проекта с именем project1-1.0-SNAPSHOT.jar. Это имя создается в результате объединения значений элементов artifactId и version из файла pom.xml. Понятно, что приведенные значения являются значениями по умолчанию, и их можно изменять.

- Maven поддерживает два режима сборки проектов: автоматический и интерактивный. За режим сборки отвечает так называемый archetype. Если в консольной команде присутствует команда «mvn archetype:generate», то содержимое файла pom.xml формируется на основании данных из консольной команды. Именно это имеет место в нашем случае. При использовании интерактивного режима инициализация необходимых параметров выполняется поэтапно вручную. Мы не будем рассматривать этот режим сборки.

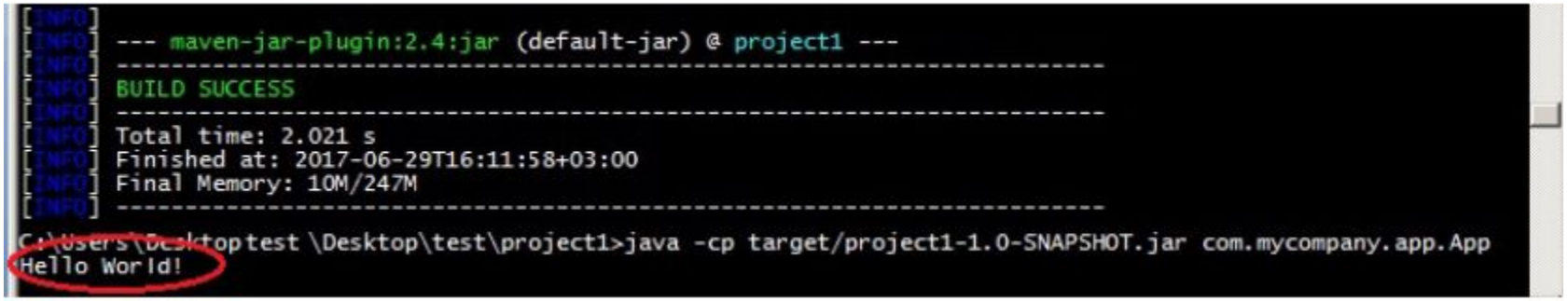
Обратите внимание, что после выполнения последней команды в папке target произошел еще ряд изменений, среди которых:

- была создана папка surefire-reports, в которой создан файл с отчетом com.mycompany.app.AppTest.txt,
- была создана папка classes, в которой создан скомпилированный класс приложения App.class,
- была создана папка maven-archiver с файлом свойств созданного проекта pom.properties.

Теперь в консольном окне надо выполнить команду:

```
java -cp target/project1-1.0-SNAPSHOT.jar
com.mycompany.app.App
```

- чтобы проверить созданный проект. При активации команды вы увидите в консольном окне результат выполнения метода main() из созданного Maven главного класса приложения App.java.



- Теперь посмотрим, как с помощью Maven можно добавить в созданный проект новый класс. Создайте рядом с файлом App.java файл с именем Fish.java и вставьте в него определение класса:

```
public class Fish
{
    private String name;
    private double weight;
    private double price;

    public Fish(String name, double weight, double price)
    {
```

```
        this.name=name;
        this.weight=weight;
        this.price=price;
    }

    public double getWeight()
    {
        return this.weight;
    }

    public double getPrice()
    {
        return this.price;
    }

    @Override
    public String toString()
    {
        return this.name+"  weight:"+this.weight+"
        price:"+this.price;
    }
}
```

- Далее надо внести изменения в App.java, чтобы каким-либо образом использовать добавленный класс. Мы просто создадим объект этого класса и выведем его описание в консольное окно. Для вывода описания объекта используем переопределенный метод toString():

```
package com.mycompany.app;

/**
 *
 * Hello world!
 */
public class App
{
    public static void main( String[] args )
    {
        Fish f = new Fish("salmon",2.5,180);
        System.out.println( f );

    }
}
```

- Теперь нам надо собрать измененный проект. Однако его состав изменился после предыдущих сборок, ведь мы добавили новый класс. Поэтому сейчас надо указать Maven, что он должен выполнить компиляцию:

```
mvn clean compile
```

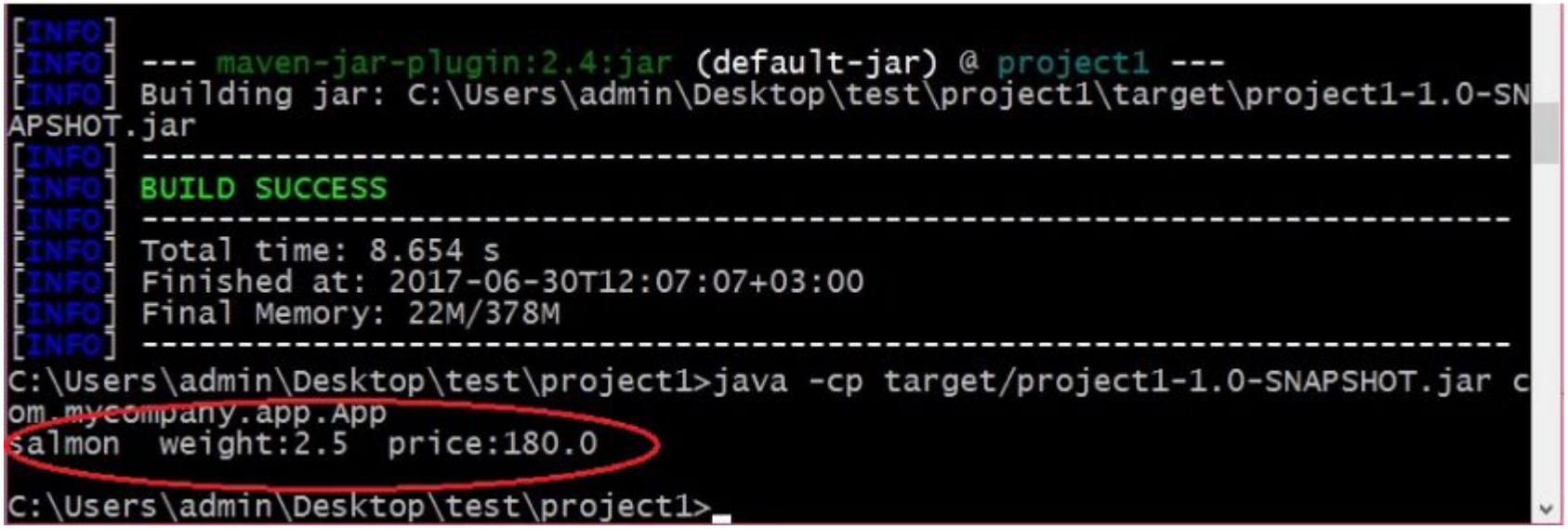
- После успешной компиляции можно выполнять сборку проекта. В этом случае команда clean будет более чем уместной, поскольку перед этим мы уже выполняли сборки, и в папках остались результаты этих действий, которые лучше удалить перед новой сборкой:

```
mvn clean package
```

Теперь снова выполним наш проект:

```
java -cp target/project1-1.0-SNAPSHOT.jar
        com.mycompany.app.App
```

Вы должны получить в консольном окне описание созданного объекта класса Fish:



- В ОС Windows вы можете создать пакетный файл и занести в него несколько Maven команд, которые будут выполняться при активации такого пакетного файла. Создайте файл project1.bat и занесит в него такие команды

```
call mvn clean
call mvn package
```

- Теперь вам достаточно в консольном окне активировать файл project1.bat, и будут выполнены все Maven команды, указанные в этом файле.

- Maven также умеет генерировать документацию по созданным проектам. Выполните в консольном окне команду

```
mvn site
```

- При первой активации она будет выполняться долго, поскольку Maven будет обновлять требуемые зависимости. В результате выполнения команды в папке target будет создана еще одна папка с именем site, а в ней будет размещаться сайт, содержащий описание проекта. Посмотреть этот сайт вы можете, активировав файл index.html. Например, в моем случае одна из страниц созданного сайта выглядит так:

project1

Last Published: 2017-07-07 | Version: 1.0-SNAPSHOT

Built by: maven

Project Information

This document provides an overview of the various documents and links that are part of this project's general information. All of this content is automatically generated by Maven on behalf of the project.

Overview

Document	Description
Dependencies	This document lists the project's dependencies and provides information on each dependency.
Dependency Convergence	This document presents the convergence of dependency versions across the entire project, and its sub modules.
Dependency Information	This document describes how to to include this project as a dependency using various dependency management tools.
About	There is currently no description associated with this project.
Plugin Management	This document lists the plugins that are defined through pluginManagement.
Plugins	This document lists the build plugins and the report plugins used by this project.
Summary	This document lists other related information of this project

- Этот сайт создается на основе pom.xml, и сейчас он не содержит никакого «персонального» описания проекта, помимо общей технической реализации. Чтобы добавить на сайт больше информации о проекте, надо внести изменения в pom.xml. Приведите этот файл к такому виду:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.mycompany.app</groupId>
    <artifactId>project1</artifactId>
    <packaging>jar</packaging>
    <version>1.0-SNAPSHOT</version>
    <name>project1</name>
    <url>http://maven.apache.org</url>
    <description>
        This project helps me to understand the basis
        of Maven leverage
    </description>
```

```
    <developers>
        <developer>
            <id>Indy</id>
            <name>Henry Walton "Indiana" Jones Jr</name>
            <email>fourthdimension@world.com</email>
            <roles>
            <role>Project Manager</role>
            <role>Developer</role>
         </roles>
         <organization>selfemployed</organization>
         <timezone>-9</timezone>
        </developer>
    </developers>
    <dependencies>
        <dependency>
          <groupId>junit</groupId>
          <artifactId>junit</artifactId>
          <version>4.8.2</version>
        <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

- В отличие от предыдущей версии этого файла, вы видите два новых добавленных элемента: description и developers. Хотя эти элементы вставлены после элемента url, это не значит, что они должны располагаться именно здесь. Все элементы можно добавлять в любом месте файла. Содержимое элемента description отображается на главной странице сайта. В этот элемент можете вставить подробное описание проекта. При наличии элемента developers в главном меню сайта появляется пункт Team, в котором отображается информация о разработчиках проекта.

Project Team

A successful project requires many people to play many roles. Some members write code or documentation, while others are valuable as testers, submitting patches and suggestions.

The project team is comprised of Members and Contributors. Members have direct access to the source of a project and actively evolve the code-base. Contributors improve the project through submission of patches and suggestions to the Members. The number of Contributors to the project is unbounded. Get involved today. All contributions to the project are greatly appreciated.

Members

The following is a list of developers with commit privileges that have directly contributed to the project in one way or another.

Image	Id	Name	Email	Organization	Roles	Time Zone
	Indy	Henry Walton "Indiana" Jones Jr	fourthdimension@world.com	selfemployed	Project Manager, Developer	-9

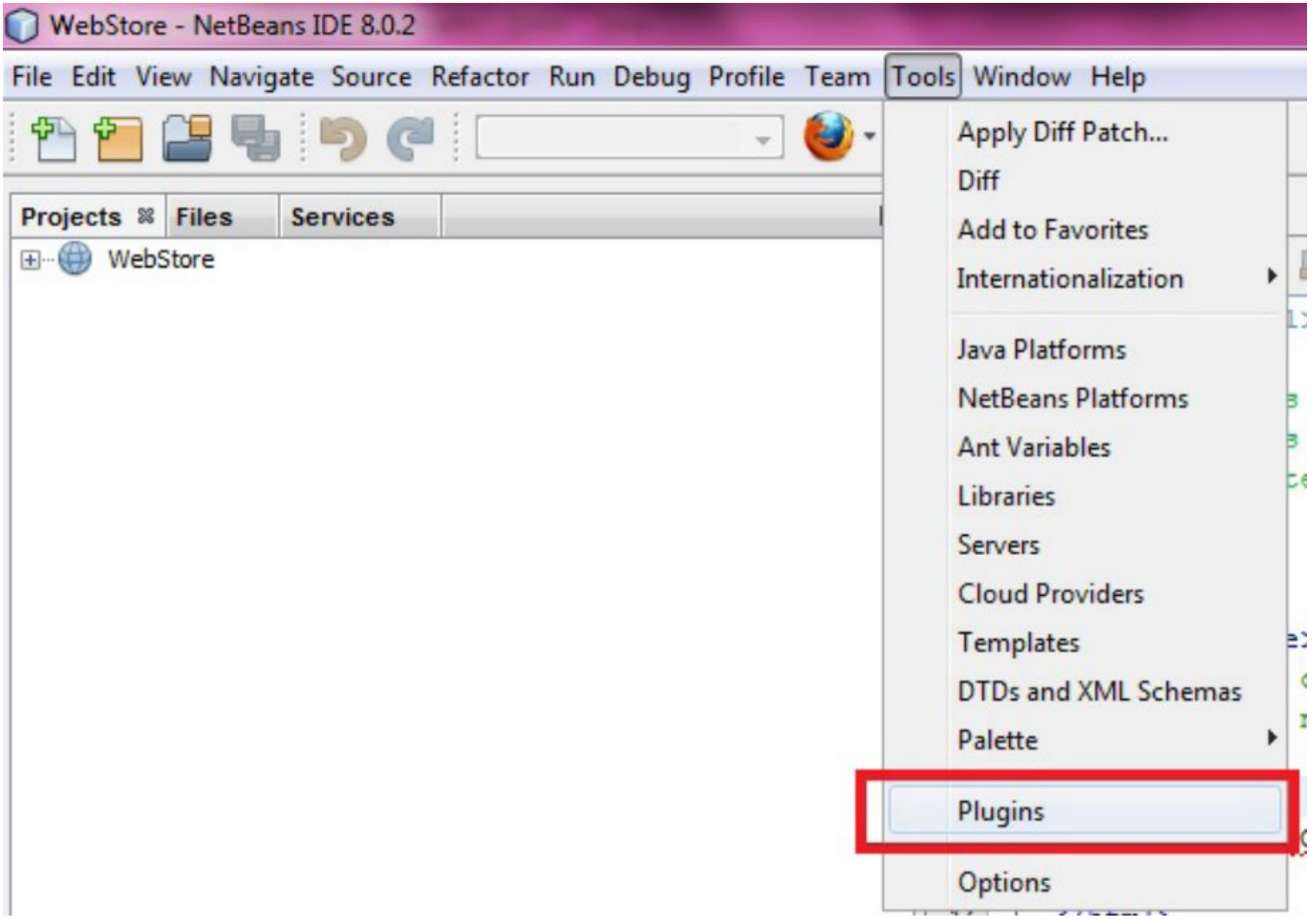
Tomcat

- Если говорить коротко, Tomcat — это сервер приложений, предназначенный для развертывания и выполнения Servlet и других компонент веб-приложений. Это продукт фирмы Apache Software Foundation, разработчика самого популярного веб-сервера Apache. Tomcat, как и большинство других продуктов этого известного разработчика, является open source продуктом.

- Посмотреть подробное описание и скачать его дистрибутив можно на сайте разработчика по адресу: http://Tomcat.apache.org/index.html. В этом уроке будет описана установка версии Tomcat 8, дистрибутив которой можно скачать со страницы: http://Tomcat.apache.org/download-80.cgi.

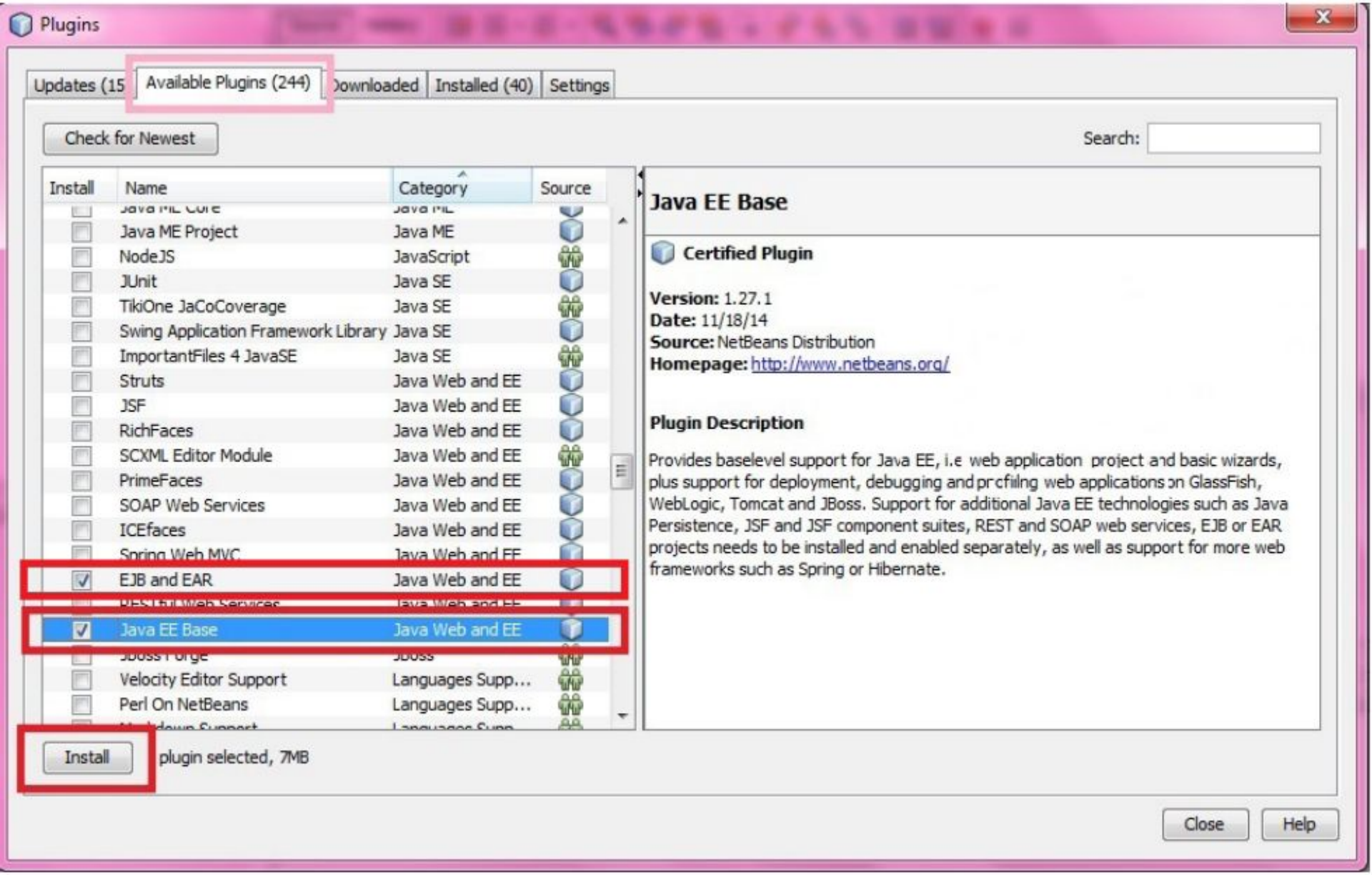
- Выберите требуемую вам версию с учетом разрядности вашей операционной системы и скачайте на свой диск. Для операционной системы Windows это будет zip архив. Разверните архив в какую-нибудь папку.

- Перед продолжением работы обратите внимание на то, что существует известная проблема совместимости Netbeans 8.1 и Tomcat, которая часто приводит к невозможности запуска Tomcat в указанной версии Netbeans. В сети предлагается несколько способов ее решения. Мы предложим вам лучшее — просто обновите Netbeans до версии 8.2, и проблемы совместимости больше не будет. Поверьте, любой способ устранения этой проблемы для Netbeans 8.1 намного более трудоемкий, чем штатное обновление Netbeans до версии 8.2. При таком обновлении все ваши настройки и предыдущие проекты не пострадают.

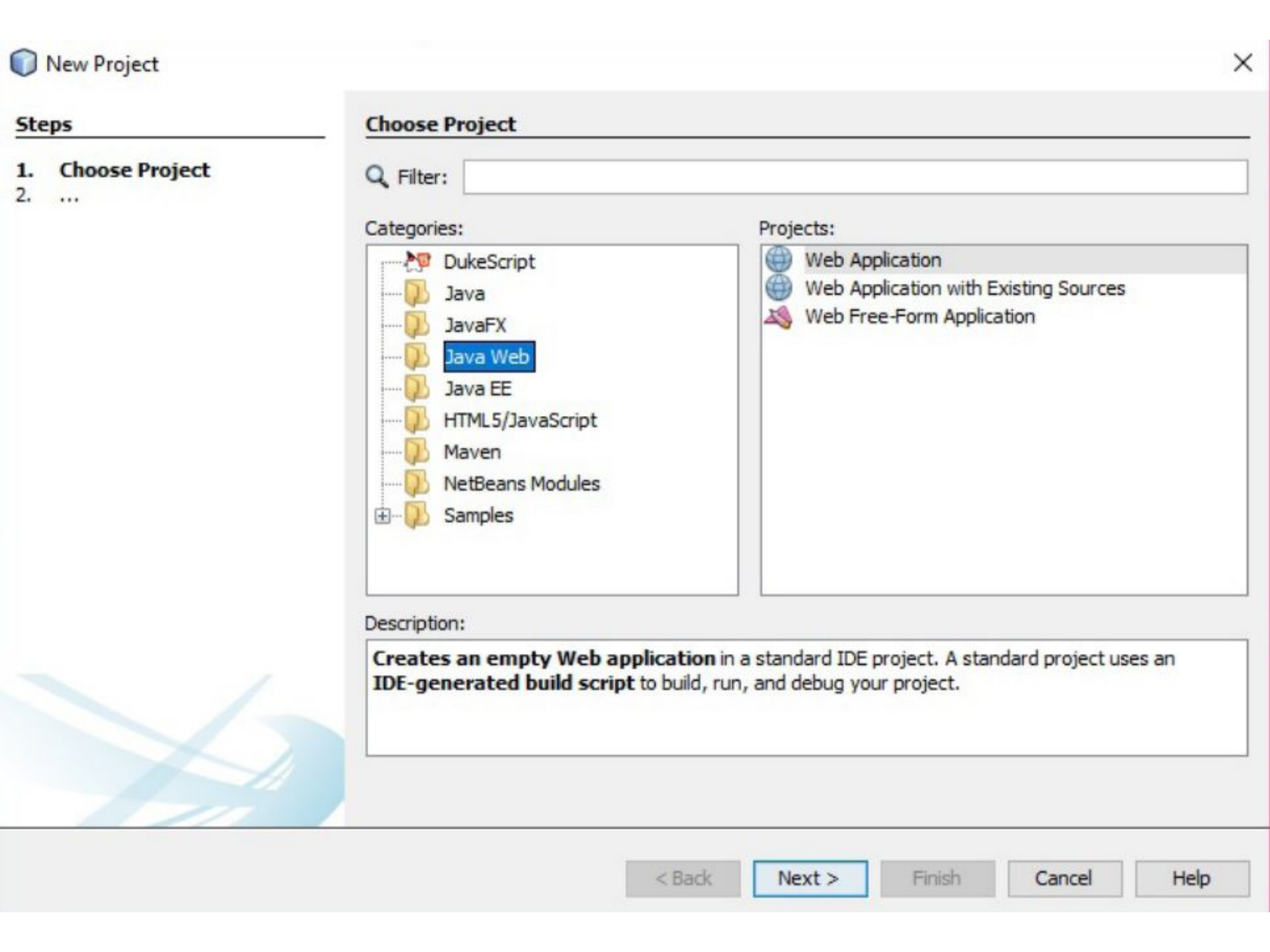

WebStore - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Files
Services
WebStore
Apply Diff Patch...
Diff
Add to Favorites
Internationalization
Java Platforms
NetBeans Platforms
Ant Variables
Libraries
Servers
Cloud Providers
Templates
DTDs and XML Schemas
Palette
Plugins
Options

- К этому моменту у вас уже установлена среда разработки NetBeans (желательно, версии 8.2), но, наверное, сейчас она настроена на использование платформы Java SE, а Java EE пока является недоступной. Как узнать о том, активирована ли платформа Java EE? Проверьте, есть ли у вас в списке создаваемых проектов категории проектов Java Web и Java EE. Если их нет, значит, вам надо установить платформу Java EE. Сделать это несложно. Сначала активируйте меню Tools – Plugins, как показано на рисунке 8.

- В появившемся окне перейдите на вкладку Available plugins и в окне поиска, в правом верхнем углу, введите то, что надо найти, в нашем случае — “web” или “Java EE base”. Затем выделите в левой панели компонеты «Java EE base» и «EJB and EAR», как показано на рисунке, и, наконец, нажмите кнопку Install (рис. 9).

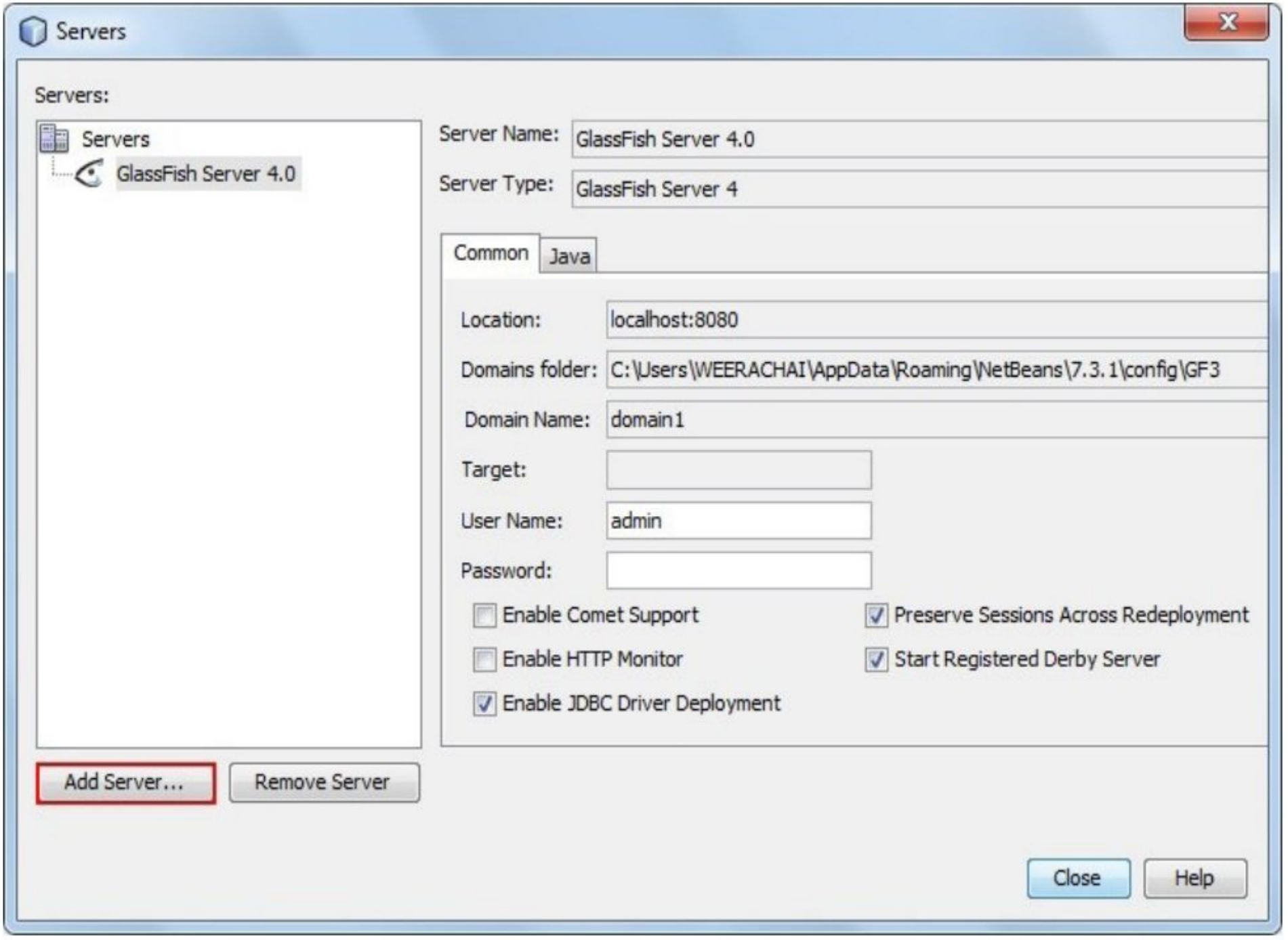

Plugins
Updates (15
Available Plugins (244)
Downloaded
Installed (40)
Settings
Check for Newest
Search:
Install
Name
Category
Source
Java ME Project
Java ME
NodeJS
JavaScript
JUnit
Java SE
TikiOne JaCoCoverage
Java SE
Swing Application Framework Library
Java SE
ImportantFiles 4 JavaSE
Java SE
Struts
Java Web and EE
JSF
Java Web and EE
RichFaces
Java Web and EE
SCXML Editor Module
Java Web and EE
PrimeFaces
Java Web and EE
SOAP Web Services
Java Web and EE
ICEfaces
Java Web and EE
EJB and EAR
Java Web and EE
Java EE Base
Java Web and EE
Velocity Editor Support
Languages Supp...
Perl On NetBeans
Languages Supp...
Java EE Base
Certified Plugin
Version: 1.27.1
Date: 11/18/14
Source: NetBeans Distribution
Homepage: http://www.netbeans.org/
Plugin Description
Provides baselevel support for Java EE, i.e web application project and basic wizards, plus support for deployment, debugging and profiling web applications on GlassFish, WebLogic, Tomcat and JBoss. Support for additional Java EE technologies such as Java Persistence, JSF and JSF component suites, REST and SOAP web services, EJB or EAR projects needs to be installed and enabled separately, as well as support for more web frameworks such as Spring or Hibernate.
Install
plugin selected, 7MB
Close
Help

- После установки плагина, возможно, понадобится перегрузка NetBeans. Чтобы убедиться в том, что платформа Java EE установлена, снова перейдите в меню File–New Project и проверьте, появился ли там тип проектов Java Web.

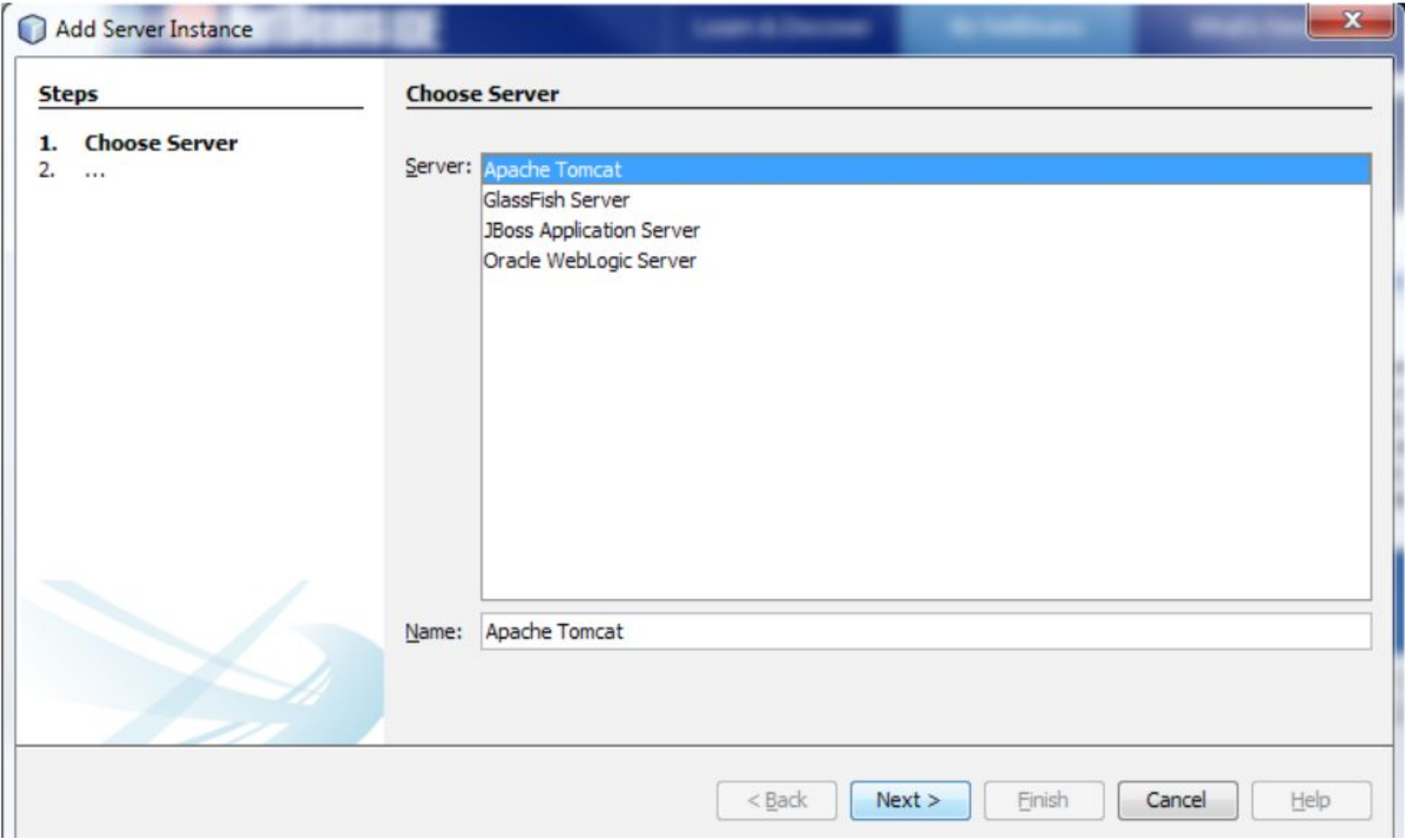

New Project
Steps
1. Choose Project
2. ...
Choose Project
Filter:
Categories:
DukeScript
Java
JavaFX
Java Web
Java EE
HTML5/JavaScript
Maven
NetBeans Modules
Samples
Projects:
Web Application
Web Application with Existing Sources
Web Free-Form Application
Description:
Creates an empty Web application in a standard IDE project. A standard project uses an IDE-generated build script to build, run, and debug your project.
< Back
Next >
Finish
Cancel
Help

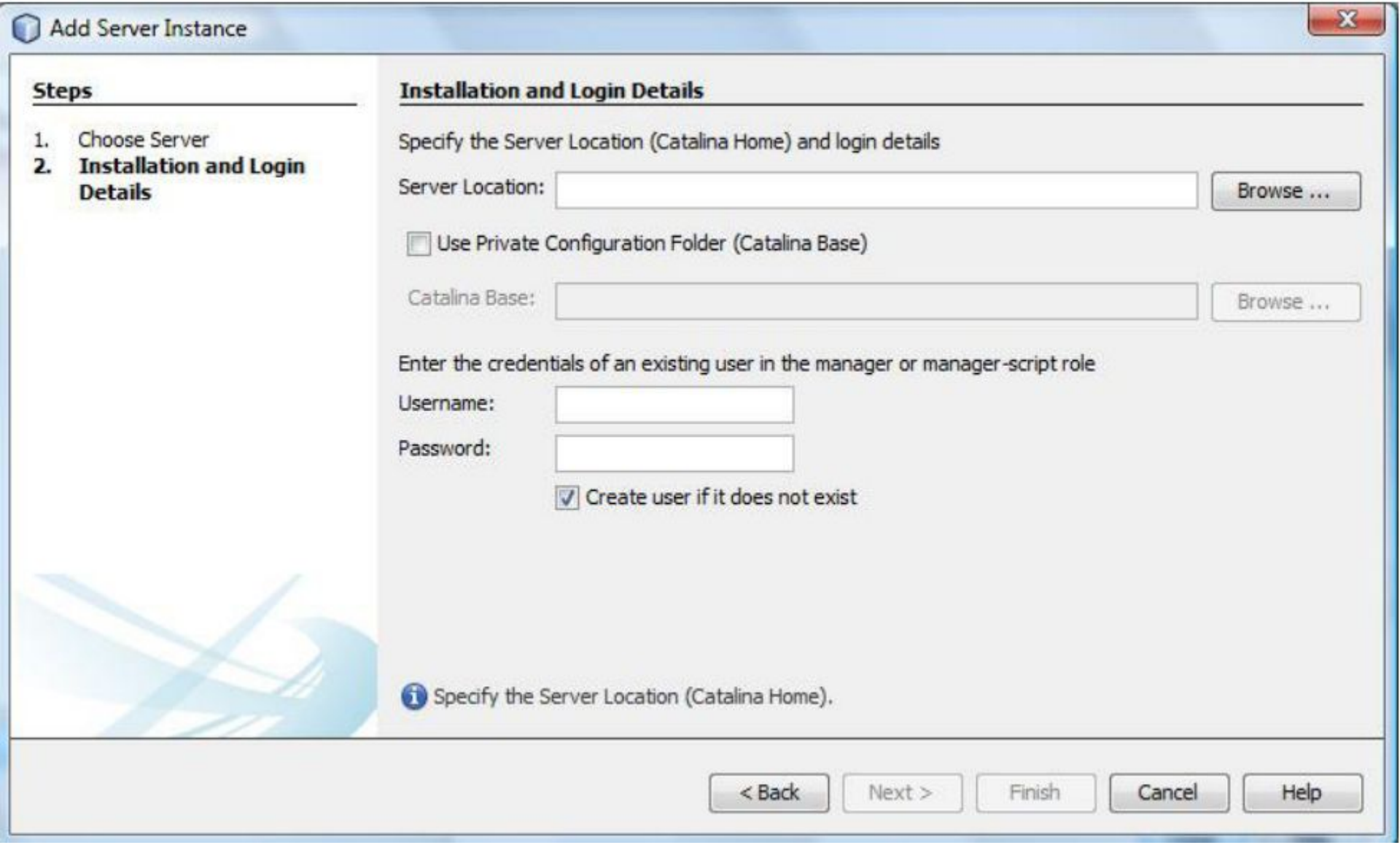
- Обратите внимание, пока что мы с вами лишь установили платформу Java EE. Эта платформа позволяет создавать веб-приложения. Но любое веб-приложение требует наличия сервера приложений. Такого, как Tomcat, который ожидает нас в папке, где мы его развернули. Сейчас мы рассмотрим процесс установки самого Tomcat

- Снова активируйте меню Tools — Servers. Вы увидите окно, в котором отображены уже установленные в вашем NetBeans серверы приложений. Возможно, это окно будет еще пустым, возможно, там уже будет отображен сервер GlassFish — это не имеет значения для того, что мы делаем. Поэтому просто нажмите в левом нижнем углу окна кнопку Add Server, как показано на рисунке 11.

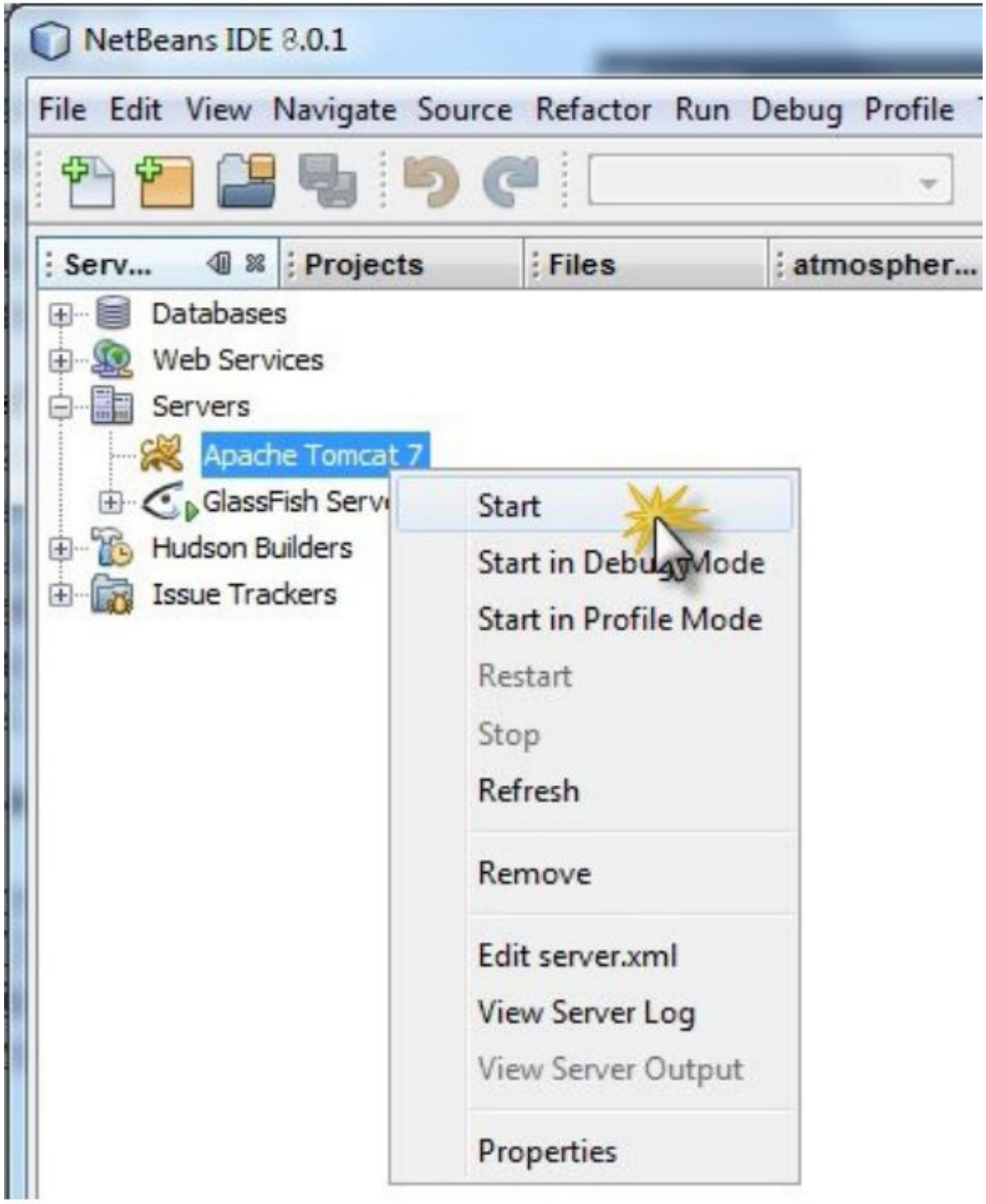

Servers
Servers:
Servers
GlassFish Server 4.0
Server Name:
GlassFish Server 4.0
Server Type:
GlassFish Server 4
Common
Java
Location:
localhost:8080
Domains folder:
C:\Users\WEERACHAI\AppData\Roaming\NetBeans\7.3.1\config\GF3
Domain Name:
domain1
Target:
User Name:
admin
Password:
Enable Comet Support
Preserve Sessions Across Redeployment
Enable HTTP Monitor
Start Registered Derby Server
Enable JDBC Driver Deployment
Add Server...
Remove Server
Close
Help

- Теперь вы увидите новое окно, в котором сможете установить требуемый Tomcat. Сначала выберите опцию Apache Tomcat и нажмите кнопку Next (Далее) (Рис. 12).
- В следующем окне укажите путь к папке, в которой вы развернули дистрибутив Tomcat и другие требуемые данные, и нажмите кнопку Finish (Готово) (Рис. 13).


Add Server Instance
Steps
1. Choose Server
2. ...
Choose Server
Server:
Apache Tomcat
GlassFish Server
JBoss Application Server
Oracle WebLogic Server
Name:
Apache Tomcat
< Back
Next >
Finish
Cancel
Help


Add Server Instance
Steps
1. Choose Server
2. Installation and Login Details
Installation and Login Details
Specify the Server Location (Catalina Home) and login details
Server Location:
Browse ...
Use Private Configuration Folder (Catalina Base)
Catalina Base:
Browse ...
Enter the credentials of an existing user in the manager or manager-script role
Username:
Password:
Create user if it does not exist
Specify the Server Location (Catalina Home).
< Back
Next >
Finish
Cancel
Help

- После успешного завершения этих действий в вашем NetBeans будет установлен сервер приложений Tomcat, который будет использоваться при разработке веб-приложений. Чтобы проверить установку Tomcat, перейдите в окне инспектора на вкладку Services, разверните узел Servers, выделите Tomcat и нажмите правую кнопку мышки. Из появившегося контекстного меню выберите и активируйте команду Start.


NetBeans IDE 8.0.1
File Edit View Navigate Source Refactor Run Debug Profile
Serv...
Projects
Files
atmospher...
Databases
Web Services
Servers
Apache Tomcat 7
GlassFish Serv
Hudson Builders
Issue Trackers
Start
Start in Debug Mode
Start in Profile Mode
Restart
Stop
Refresh
Remove
Edit server.xml
View Server Log
View Server Output
Properties

- Теперь Tomcat у вас установлен и готов к использованию. Очень скоро мы начнем с ним работать, а пока рассмотрим еще некоторые важные компоненты Java EE платформы.

JBoss

- JBoss — еще один популярный сервер приложений для платформы Java EE. Этот продукт представляет собой отличную платформу для выполнения средних и больших распределенных Java EE приложений. JBoss включает в себя набор уже сконфигурированных служб, необходимых для их обслуживания. Использование JBoss обеспечивает для распределенного приложения такие опции:

- кластеризацию (Clustering) — объединение группы связанных компьютеров вместе настолько плотно, что во многих отношениях такие компьютеры работают как один;

- балансировку загрузки приложения (Load Balancing) — оптимизацию при распределении входящих запросов;

- кеширование часто используемых данных (Caching) — сохранение таких данных во временном хранилище для организации более быстрого доступа к ним;

- Enterprise Java Beans — допускает использование этих компонентов, получивших широкую популярность у разработчиков Java EE. Кроме перечисленных опций JBoss предоставляет еще ряд менее значимых. Понятно, что, будучи сервером приложений, JBoss является контейнером для развертывания и выполнения таких Java EE компонент, как Servlet и JSP.

Spring

- Хотя Spring предназначен для разработки Java EE приложений, отдельные его модули могут использоваться и в приложениях других типов. Вот основные черты этого фреймворка.

- Spring имеет модульную структуру, и все его модули практически независимы друг от друга. Все что вам надо знать при использовании Spring — это то, какой модуль вам необходим, и как его использовать. При этом вы можете совсем ничего не знать о других модулях, и это не помешает вам успешно работать со Spring, потому что все его модули работают независимо.

- Spring имеет собственный MVC модуль, который позволяет вам обходиться без других MVC-фреймворков.
- Spring содержит собственный API для работы с JDBC, что избавляет разработчика от необходимости выполнять огромную часть рутинной работы при использовании источников данных

- Spring имеет API для обработки возникающих исключений.
- Spring управляет созданием объектов классов, входящих в состав вашего приложения.
- Spring содержит контейнеры для управления жизненным циклом объектов вашего приложения.

- Работа со Spring выполняется декларативно
- Приложения, созданные с помощью Spring, имеют размер всего около 2MB, что также является достоинством этого фреймворка

- При создании приложения вы должны стремиться, чтобы ваши классы были максимально независимыми друг от друга. Это позволит вам использовать их в других приложениях, а также облегчит выполнение unit тестирования. Но, с другой стороны, классы в приложении должны взаимодействовать друг с другом. Каким образом можно примирить эти два взаимоисключающих требования? Для этой цели Spring предлагает использовать паттерн проектирования Dependency Injection (DI), являющийся разновидностью концепции Inversion of Control (IoC). Контейнеры IoC являются центральной частью Spring.

- Эти контейнеры управляют всеми аспектами жизненного цикла объектов: их созданием, конфигурацией, установлением связей между ними. Контейнеры выполняют все свои действия на основании сценария, записанного либо в XML формате, либо с помощью Java аннотаций, либо же в Java коде.

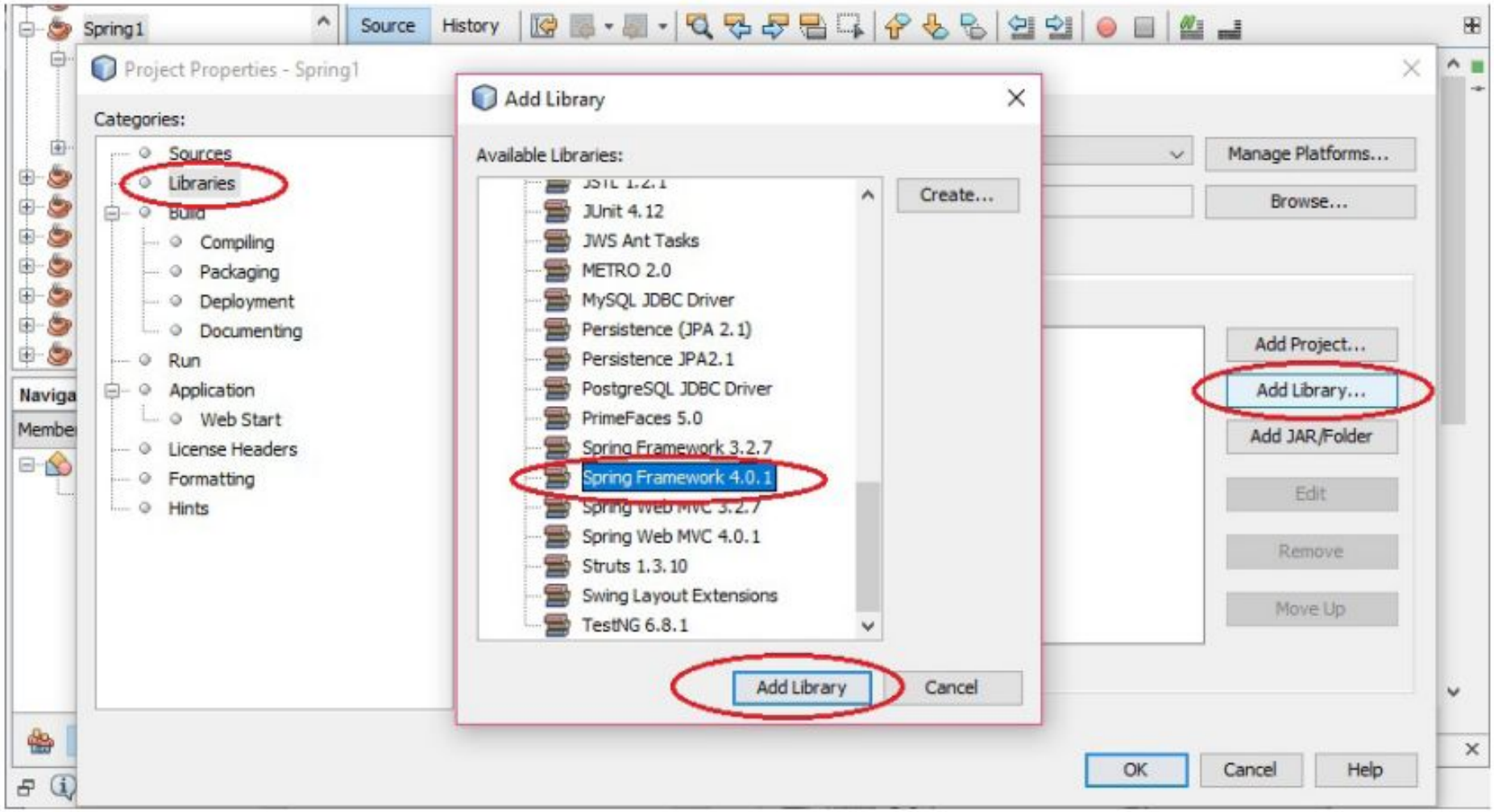
- В Spring существуют два вида контейнеров: BeanFactory и ApplicationContext. Контейнер BeanFactory является более простым и предлагает основные действия, необходимые для выполнения DI. Его рекомендуется использовать в самых простых приложениях. Контейнер ApplicationContext включает в себя все возможности BeanFactory и, кроме этого, еще может применяться в Java EE приложениях. Его надо использовать, если вы создаете веб-приложение.

- Вы уже знакомы с понятием Java bean. Это набор формальных требований к объявлению Java класса, выполнение которых делает такой класс «правильным» с точки зрения Java, а именно — превращает класс в компоненту, которую можно использовать многократно в различных библиотеках и API. Использование Java bean повышает эффективность использования класса. Если вы забыли набор формальных требований к Java bean, повторим их еще раз. Чтобы класс стал Java bean, он должен:

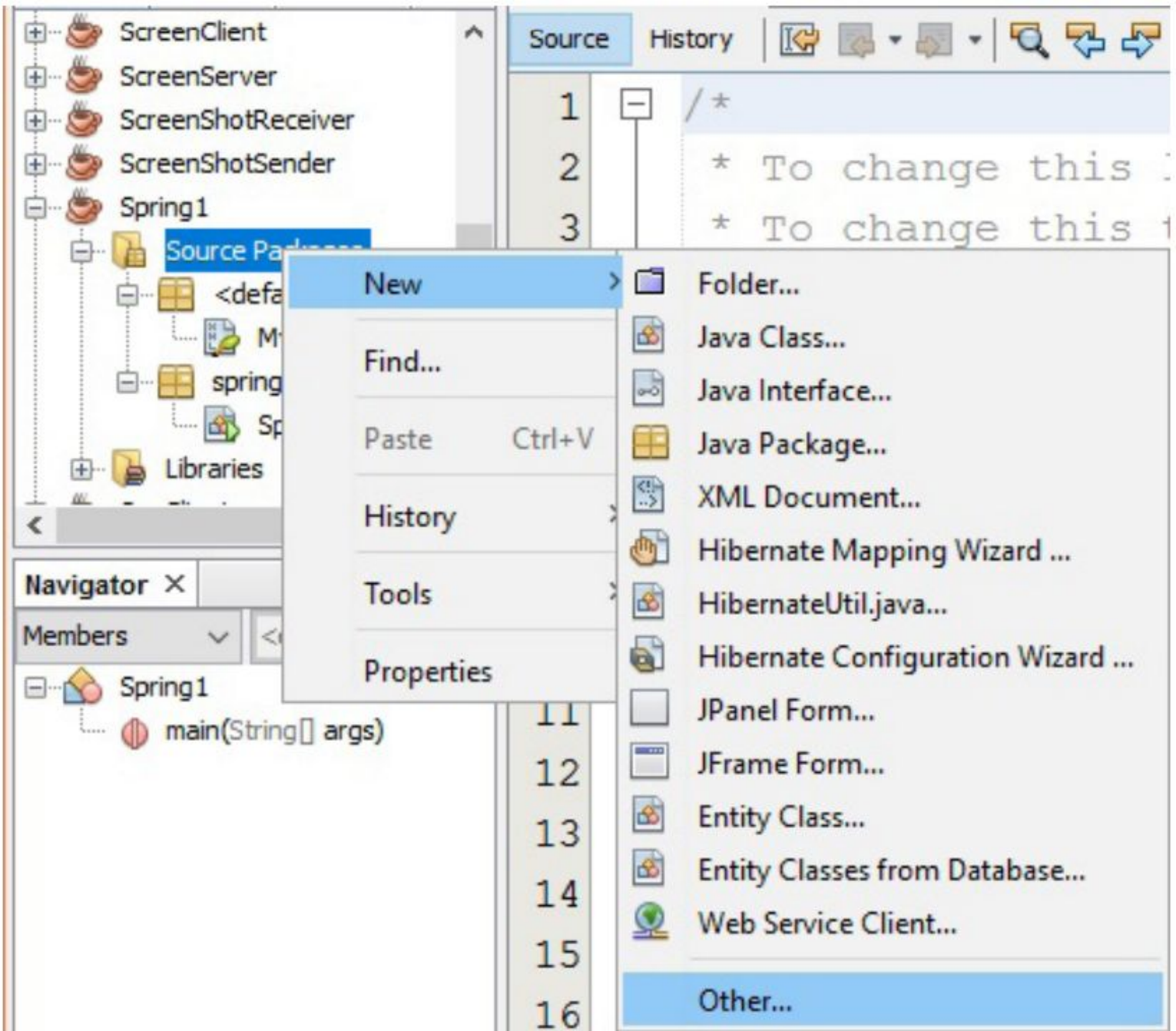
- иметь public конструктор без параметров;
- быть сериализуемым;
- содержать сеттеры и геттеры для своих свойств.
- Иногда еще требуется, чтобы класс содержал методы equals(), hashCode() и toString().ы

Первое Spring приложение

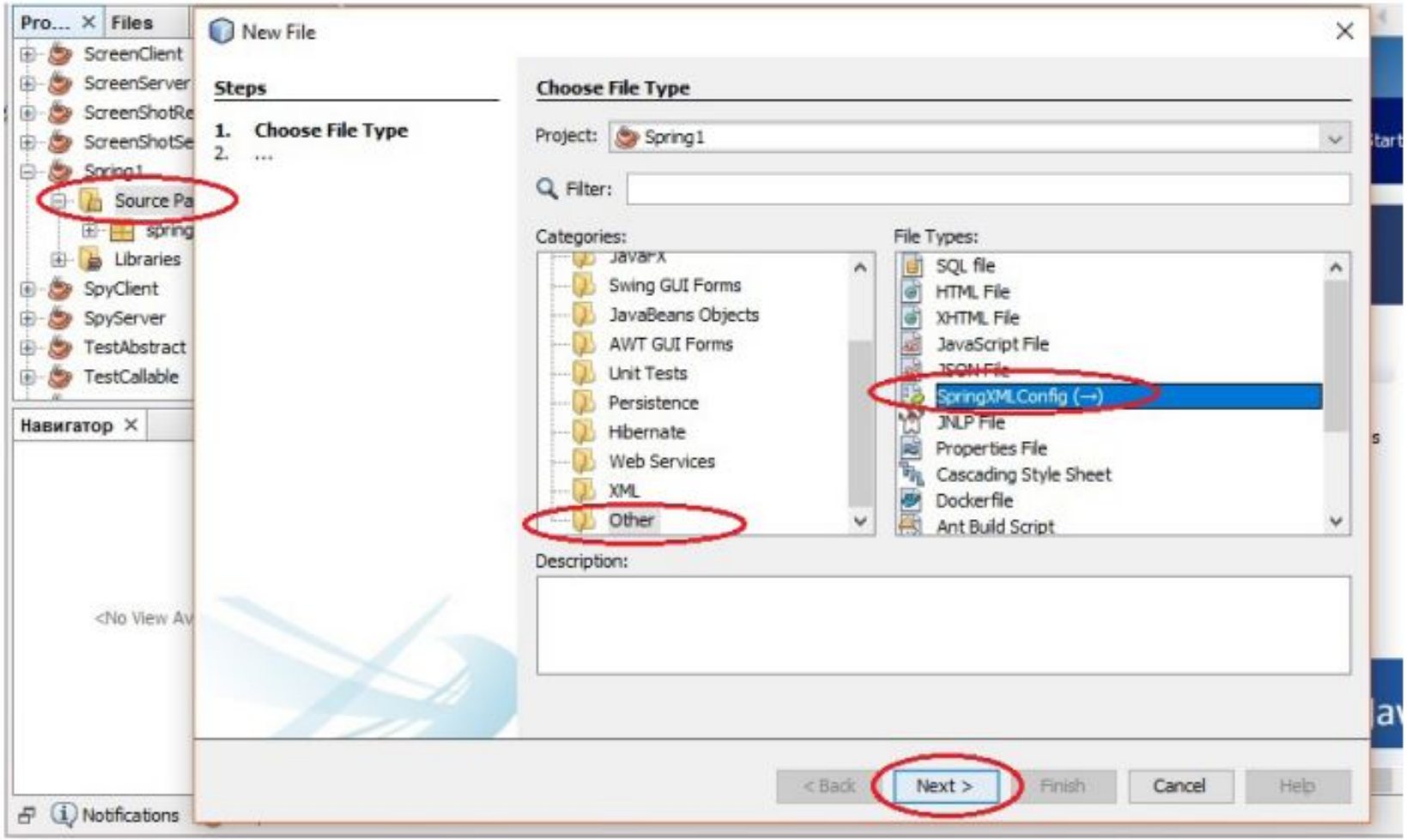
- Создайте в NetBeans новый проект с именем, например, Spring1. Когда проект будет создан, выделите его в окне инспектора и активируйте команду Свойства. В правой части появившегося окна выделите опцию Libraries, а затем нажмите справа кнопку «Add Library …»


Spring1
Source
History
Project Properties - Spring1
Categories:
Sources
Libraries
Build
Compiling
Packaging
Deployment
Documenting
Run
Application
Web Start
License Headers
Formatting
Hints
Add Library
Available Libraries:
JUnit 4.12
JWS Ant Tasks
METRO 2.0
MySQL JDBC Driver
Persistence (JPA 2.1)
Persistence JPA2.1
PostgreSQL JDBC Driver
PrimeFaces 5.0
Spring Framework 3.2.7
Spring Framework 4.0.1
Spring Web MVC 4.0.1
Struts 1.3.10
Swing Layout Extensions
TestNG 6.8.1
Create...
Add Library
Cancel
Manage Platforms...
Browse...
Add Project...
Add Library...
Add JAR/Folder
Edit
Remove
Move Up
OK
Cancel
Help

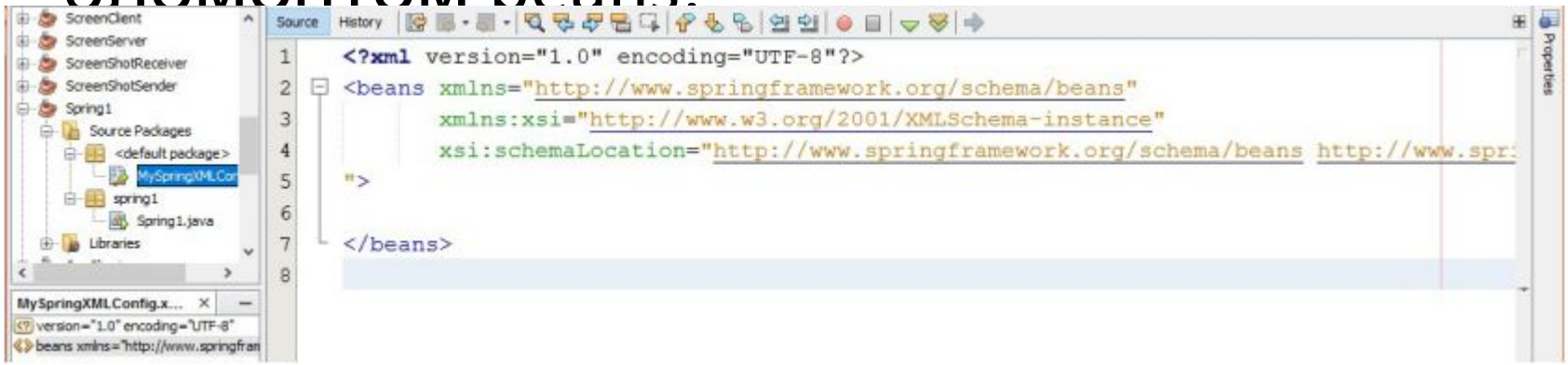
- В появившемся окне выберите Spring Framework последней версии (Рис. 15). В нашем первом Spring приложении мы будем описывать свои классы в специальном конфигурационном XML файле. Он должен располагаться в папке проекта src, поэтому для его добавления выделите узел SourcePackages, активируйте контекстное меню и кликните по опции New — Other (рис. 16).


ScreenClient
ScreenServer
ScreenShotReceiver
ScreenShotSender
Spring1
Source Packages
Libraries
Navigator
Members
Spring1
main(String[] args)
Source
History
1
2
3
/*
* To change this
* To change this
11
12
13
14
15
16
New
Find...
Paste Ctrl+V
History
Tools
Properties
Folder...
Java Class...
Java Interface...
Java Package...
XML Document...
Hibernate Mapping Wizard ...
HibernateUtil.java...
Hibernate Configuration Wizard ...
JPanel Form...
JFrame Form...
Entity Class...
Entity Classes from Database...
Web Service Client...
Other...

В появившемся окне выберите Other — SpringXMLConfig(->):



- Выберите для добавляемого конфигурационного файла имя и нажмите кнопку Finish, не отмечая никакие другие опции. Выделите добавленный файл — и увидите заготовку пустого XML файла с корневым элементом beans:



- В этом файле надо описывать классы, из которых мы хотим создать приложение. Приведем сначала код класса Student, а затем опишем этот класс в конфигурационном файле. Создайте в проекте пакет с именем myclass и добавьте в него такой класс:

```
public class Student implements Serializable
{
    private String name;
    private double rate;

    public Student() {
    }

    @Override
    public String toString() {
        return "Student{" + "name=" + name + ",
                rate=" + rate + '}';
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public double getRate() {
        return rate;
    }

    public void setRate(double rate) {
        this.rate = rate;
    }
}
```

- Обратите внимание, что класс оформлен как Java bean — это обязательное условие в случае использования Spring. Класс Student очень простой, он содержит два поля и все необходимое для их обслуживания. Теперь давайте посмотрим, как этот класс надо описывать в конфигурационном файле

- Каждый класс приложения должен описываться в конфигурационном файле парным элементом bean. В этом элементе необходимо указать атрибут id и присвоить ему уникальное значение. Обычно это имя класса, но с маленькой буквы:

```
<bean id="student">
</bean>
```

- Затем с помощью атрибута class надо указать, какой именно класс соответствует этому bean. При указании имени класса надо обязательно указывать имя пакета, в котором определен класс:

```
<bean id="student" class="myclass.Student">
</bean>
```

- Далее надо указать атрибут scope, который определяет, как будет вести себя Spring всякий раз, когда вы будет создавать bean с именем student. У атрибута scope есть пять значений, три последние из которых используются только в веб-приложениях:

scope="singleton"	будет создан только один объект такого bean, и при каждой попытке создать новй объект будет возвращаться этот объект;
scope="prototype"	при каждой попытке создать новый объект будет создаваться новый объект;
scope="request"	указывает, что bean существует только в пределах запроса;
scope="session"	указывает, что bean существует в пределах сессии;
scope="global-session"	указывает, что bean существует в пределах глобальной сессии.

Мы планируем создавать много объектов класса Student, поэтому добавляем атрибут scope со значением prototype:

```
<bean id="student" class="myclass.Student"
      scope="prototype">
</bean>
```

- В элементе bean можно описывать поля класса и присваивать им значения по умолчанию. Полное описание нашего класса может выглядеть так:

```
<bean id="student" class="myclass.Student"
scope="prototype">
    <property name="name" value="No Name"></property>
    <property name="rate" value="0"></property>
</bean>
```

- В этом описании вам все должно быть понятно. Добавьте его в наш конфигурационный файл. Теперь перейдем к методу main() в главном классе нашего приложения и добавим туд

```
public static void main(String[] args) {
    ApplicationContext context = new
        ClassPathXmlApplicationContext(
        "MySpringXMLConfig.xml");
    Student s1 = (Student) context.getBean("student");
    s1.setName("Robert");
    s1.setRate(145.5);
    System.out.println(s1);

    Student s2 = context.getBean(Student.class);
    System.out.println(s2);
}
```

- Сначала создается контекст приложения — на основании конфигурационного файла, в котором описаны все его классы. В нашем приложении там пока описан только один класс Student. В дальнейшем для создания объекта какого-нибудь класса надо будет обращаться к этому контексту и вызывать метод getBean(), передавая ему в качестве параметра, строковый идентификатор (значение

```
Student s1 = (Student) context.getBean("student");
```

создать.

- Запустите наше приложение, и увидите в консольном окне такой вывод:

```
Student{name=Robert, rate=145.5}
Student{name=No Name, rate=0.0}
```

Понятие сервлета

- Сервлет представляет собой приложение, выполняющееся на веб сервере или, гораздо чаще, на сервере приложений (Tomcat, GlassFish и др.). Сервлет располагается между клиентской и серверной частью приложения и предназначен для обработки клиентских запросов. Обрабатывая клиентские запросы, сервлет может обращаться к БД, к веб службам или же формировать Response самостоятельно.

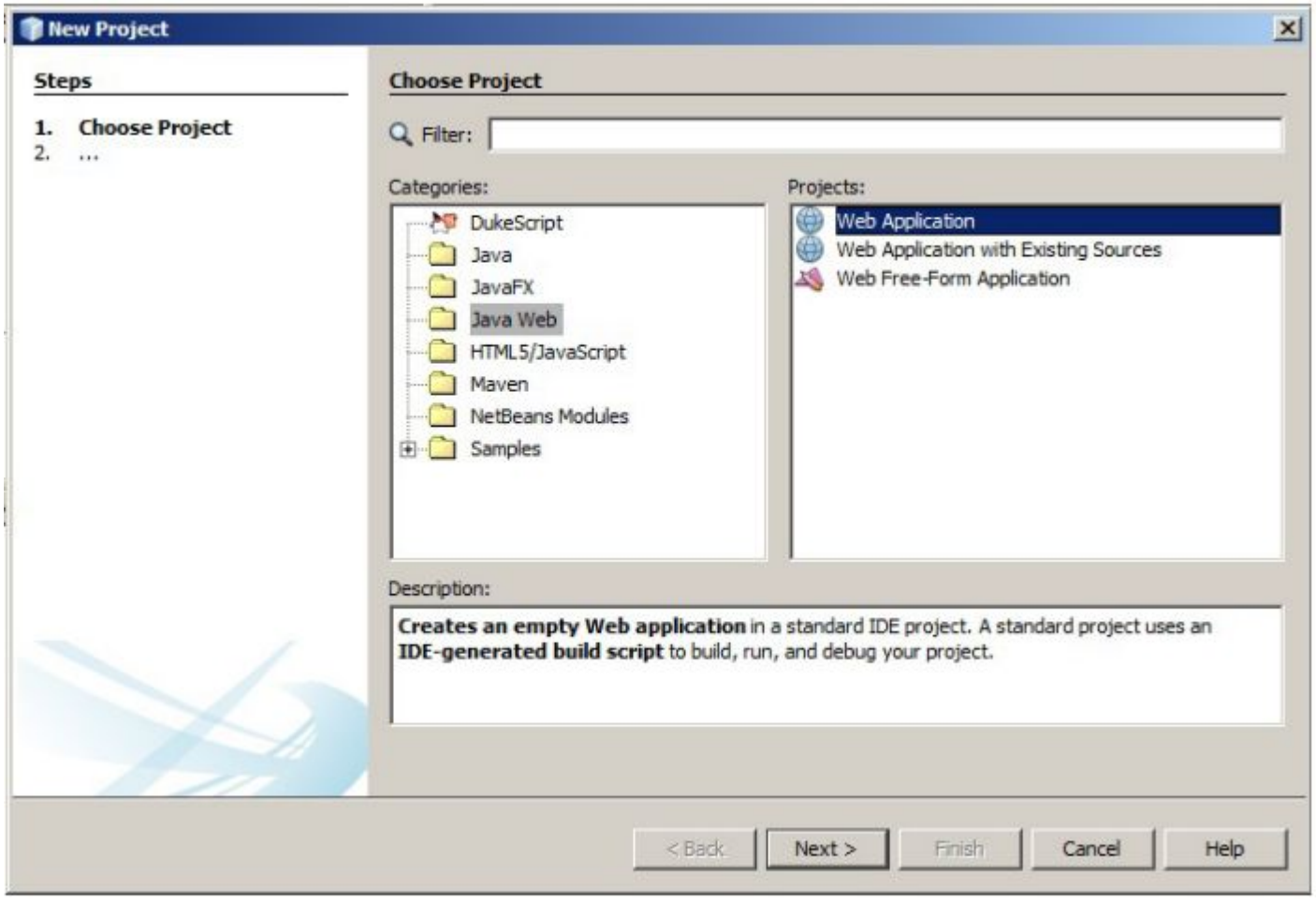
- По своей природе сервлеты являются Java классами, наследующими класс HttpServlet. Реализация этих классов базируется на пакетах javax.servlet и javax.servlet.http, входящими в состав Java EE. В классе сервлета определены методы, управляющие его жизненным циклом. Это такие методы, как:

- init() — вызывается единожды и инициализирует сервлет. Сервлет обычно создается, когда клиент обращается к Url, соответствующему сервлету. Однако можно сделать и так, чтобы сервлет инициализировался сразу при запуске сервера;

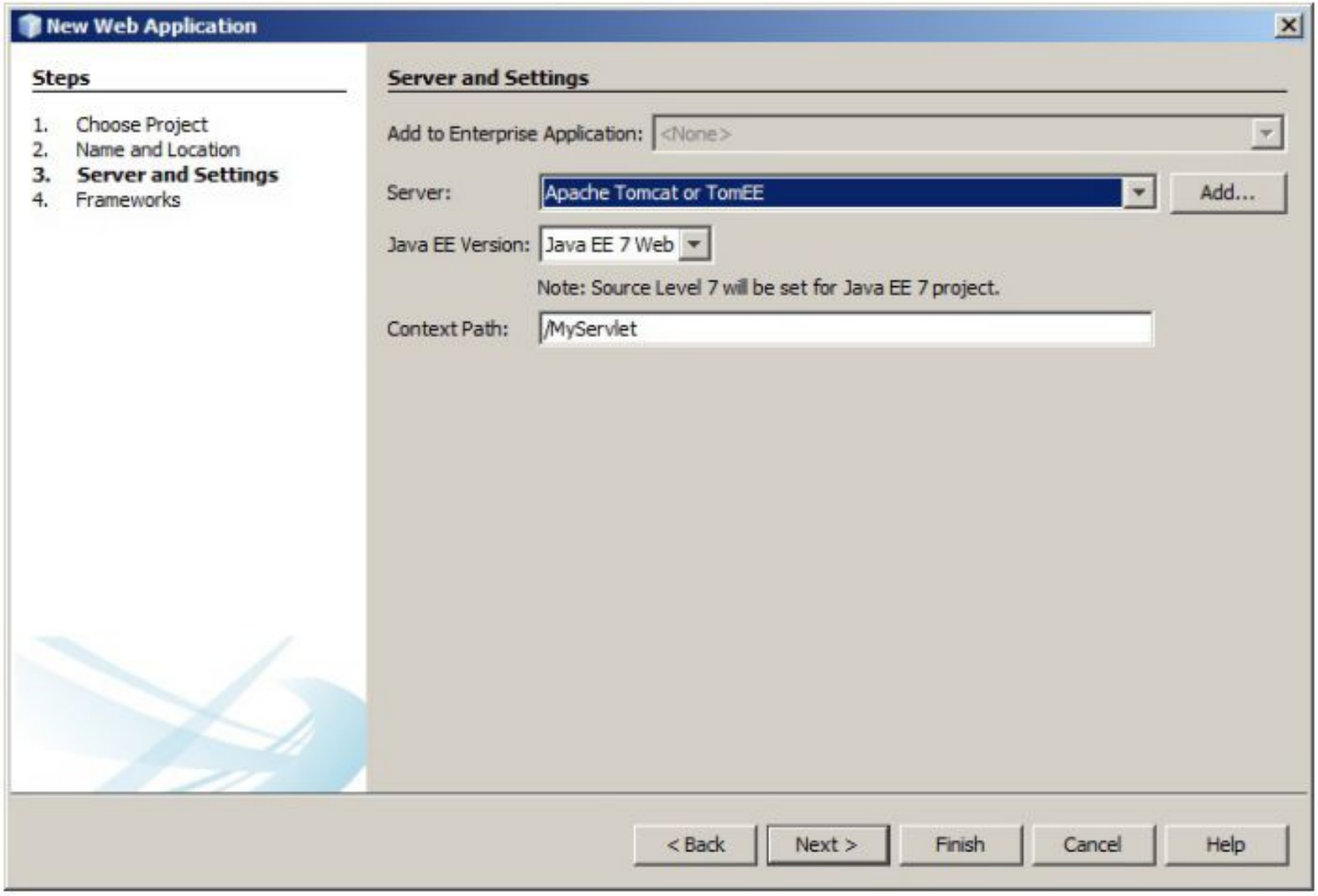
- service() — вызывается всякий раз, когда сервлет получает клиентский запрос. Это основной метод сервлета. В нем выполняется работа, для которой сервлет создан. Каждый раз при получении нового запроса от клиента сервер вызывает в новом потоке метод service(), далее уже сам этот метод проверяет, по какому HTTP методу пришел запрос (GET, POSTDELETE, PUT), и вызывает соответствующий метод doGet(), doPost(), doDelete() и т.п. Обратите внимание, что метод service() вызывается контейнером, в котором размещен сервлет, и программисту не надо самостоятельно вызывать его. Задача программиста — переопределить методы doGet(), doPost(), doDelete() и т.п.;

- destroy() — вызывается единственный раз и завершает работу сервлета, после чего сервлет удаляется сборщиком мусора. В этом методе надо выполнять такие действия, как отключение от серверов БД, завершение потоков-демонов, сохранение куки файлов и т.п.

- После этой начальной информации рассмотрим примеры использования сервлетов. Для работы с ними надо иметь установленной платформу Java EE и какой-нибудь сервер приложений. У нас к этому моменту оба условия выполнены: установлены Java EE и Tomcat. Поэтому перейдем к созданию нового приложения.


New Project
Steps
1. Choose Project
2. ...
Choose Project
Filter:
Categories:
DukeScript
Java
JavaFX
Java Web
HTML5/JavaScript
Maven
NetBeans Modules
Samples
Projects:
Web Application
Web Application with Existing Sources
Web Free-Form Application
Description:
Creates an empty Web application in a standard IDE project. A standard project uses an IDE-generated build script to build, run, and debug your project.
< Back
Next >
Finish
Cancel
Help

- Запустите NetBeans и создайте новый проект по шаблону Java Web (см. рис. 19). Назовите проект, например MyServlet1, и перейдите к следующему окну, где вам предложат выбрать сервер для хостинга создаваемого приложения. Выберите опцию Apache Tomcat:

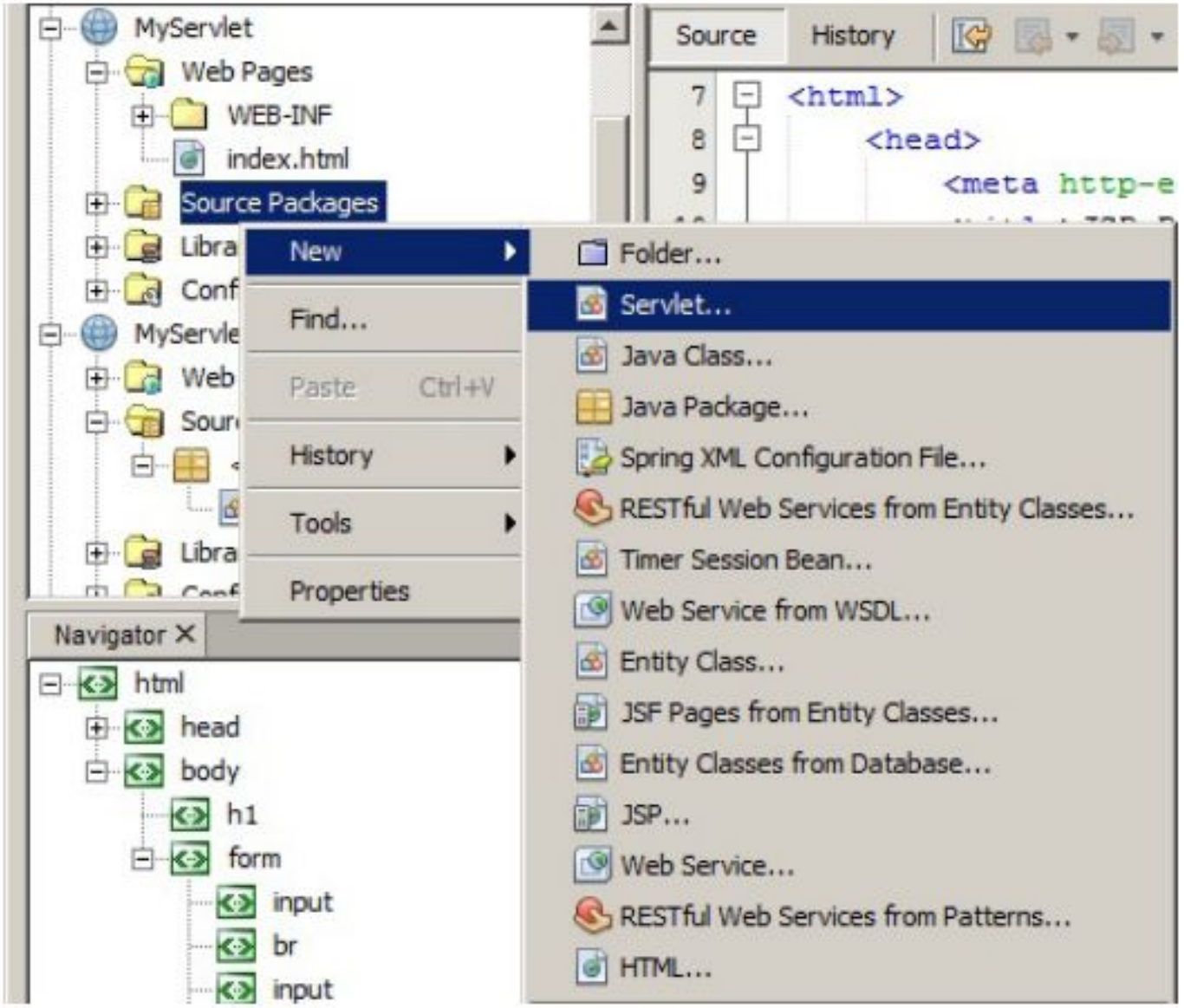

New Web Application
Steps
1. Choose Project
2. Name and Location
3. Server and Settings
4. Frameworks
Server and Settings
Add to Enterprise Application:
<None>
Server:
Apache Tomcat or TomEE
Add...
Java EE Version:
Java EE 7 Web
Note: Source Level 7 will be set for Java EE 7 project.
Context Path:
/MyServlet
< Back
Next >
Finish
Cancel
Help

- В корневой папке созданного приложения вы увидите файл index.html, это страница входа созданного приложения. Приведите разметку элемента body этого файла к такому виду:

```
<body>
    <h1>Hello World!</h1>
    <form action="GreetingServlet" method="POST">
        Login: <input type="text" name="login"
               size="20"> <br/>
        Password: 
        <input type="password" name="pass" size="20">
       <br /><br />
       <input type="submit" value="Submit">
    </form>
</body>
```

- Мы создали простую форму для ввода имени и пароля. Обработчиком данных запроса, который будет создан при нажатии на кнопку submit в этой форме, будет сервлет.

- Чтобы добавить в проект сервлет, можно поступить таким образом. Выделите в проекте узел Source Packages, нажмите правую кнопку мышки, затем — New, затем — Servlet (рис. 21).


MyServlet
Web Pages
WEB-INF
index.html
Source Packages
Libra
Conf
MyServle
Web
Sour
Libra
Source
History
7 <html>
8 <head>
9 <meta http-e
New
Find...
Paste Ctrl+V
History
Tools
Properties
Folder...
Servlet...
Java Class...
Java Package...
Spring XML Configuration File...
RESTful Web Services from Entity Classes...
Timer Session Bean...
Web Service from WSDL...
Entity Class...
JSF Pages from Entity Classes...
Entity Classes from Database...
JSP...
Web Service...
RESTful Web Services from Patterns...
HTML...
Navigator
html
head
body
h1
form
input
br
input

- В качестве имени сервлета укажите GreetingServlet. Вы увидите шаблон класса сервлета, производного от HttpServlet. В этом классе нас будет интересовать метод processRequest() с двумя параметрами — HttpServletRequest request и HttpServletResponse response. Первый параметр содержит входящий запрос от клиента. Во второй надо записывать сформированный response на полученный request. Приведите этот метод к такому виду:

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    response.setContentType("text/html;
        charset=UTF-8");
    PrintWriter out = response.getWriter();
    String login = request.getParameter("login").
        toString();
    String pass = request.getParameter("pass").
        toString();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet GreetingServlet
        </title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Servlet GreetingServlet at" +
    request.getContextPath () + "</h1>");
    out.println("<p>Welcome, " + login + ",
            your pass is " + pass + "</p>");
    out.println("</body>");
    out.println("</html>");
    out.close();
}
```

- В этом методе выполняются очень простые действия. Сначала мы задаем требуемую кодировку и связываем с объектом response выходной поток, чтобы иметь возможность писать разметку в response, который будет получен клиентом как результат обработки формы. Затем из объекта request с помощью метода getParameter() мы по имени элементов управления получаем значения, занесенные в форму. Дальше идет формирование разметки для возвращаемой клиенту страницы.

- Запустите приложение, и в браузере откроется созданная форма. Занесите в нее какие-нибудь данные и нажмите кнопку submit. Тем самым вы создадите запрос, который будет отправлен нашему сервлету. Там его обработает метод processRequest() и вернет страницу ответа. У меня это выглядит так:

Источники данных

- Вообще говоря, источником данных не обязательно должна быть база данных. Это может быть и файл, и коллекция. Но именно базы данных практически стали стандартным источником данных. Для работы с базами данных в Java SE включен пакет java.sql, содержащий компоненту JDBC (Java DataBase Connectivity). JDBC основывается на понятии драйвера. Для подключения к конкретной БД надо динамически загрузить соответствующий драйвер и передать ему аналог строки подключения, который в Java называется URL. Основную роль в JDBC играют такие интерфейсы, как Connection, Statement, PreparedStatement, CallableStatement и ResultSet. Драйвер каждой конкретной БД реализует эти интерфейсы, как ему необходимо.

Алгоритм работы с БД выглядит таким образом:

- регистрация драйвера для конкретного сервера БД;
- выполнение соединения с сервером БД (Connection);
- создание и выполнение запроса к БД (Statement или PreparedStatement);
- извлечение результата выполненного запроса (ResultSet).

Загрузка и инициализация драйвера происходит при выполнении строки кода:

```
Class.forName(driver);
```

- здесь driver — специальный строковый описатель драйвера.

- Выполнение подключения к серверу БД выглядит так:

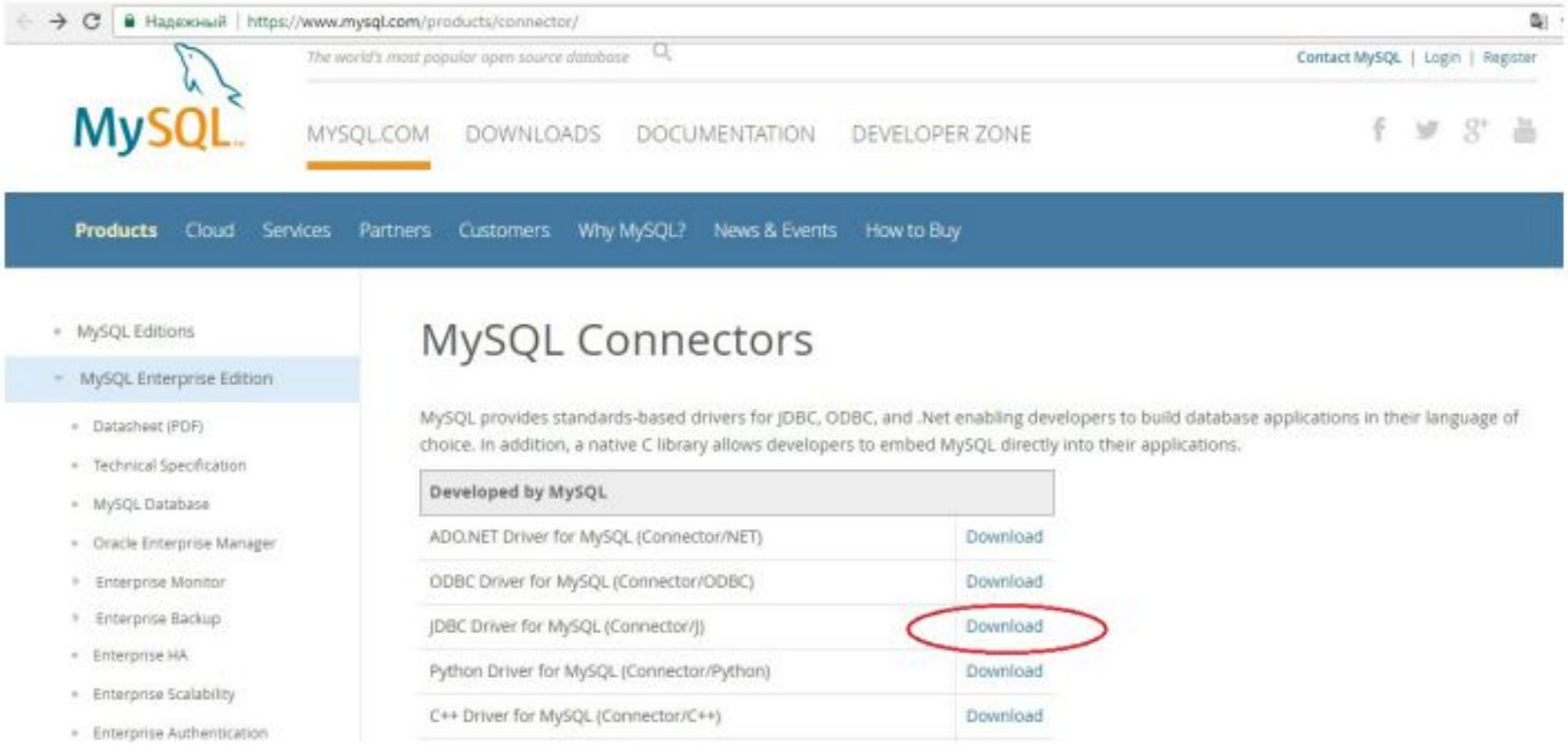
```
Connection connection = DriverManager.
                        getConnection(driverUrl);
```

- здесь driverUrl — уникальная для каждого сервера БД строка специального вида.

- Дальнейшие действия специфичны для конкретных выполняемых запросов.
- Работа с JDBC практически закрывает от пользователя реализацию конкретной БД. Другими словами, код, использующий JDBC, очень мало зависит от специфики БД. В случае изменения БД, в Java коде надо будет делать очень мало правок, а возможно, не понадобится их делать вовсе, за исключением нескольких строк подключения драйвера

- Рассмотрим пример использования БД. Напишем приложение, которое выполнит подключение к MySQL и продемонстрирует выполнение основных действий с БД. У вас уже должен быть установлен экземпляр MySQL и PhpMyAdmin. Запустите PhpMyAdmin и создайте базу данных с именем jtest. Затем запустите NetBeans и создайте новый консольный проект. Мой проект будет называться TryJDBC.

- Поскольку мы будем работать с сервером MySQL, нам надо добавить в состав приложения JDBC драйвер для работы с этим сервером. Этот драйвер можно загрузить со страницы https://www.mysql.com/products/connector/. Перейдите на эту страницу и активируйте выделенную ссылку:


Надежный | https://www.mysql.com/products/connector/
The world's most popular open source database
Contact MySQL | Login | Register
MySQL
MYSQL.COM DOWNLOADS DOCUMENTATION DEVELOPER ZONE
Products Cloud Services Partners Customers Why MySQL? News & Events How to Buy
MySQL Editions
MySQL Enterprise Edition
Datasheet (PDF)
Technical Specification
MySQL Database
Oracle Enterprise Manager
Enterprise Monitor
Enterprise Backup
Enterprise HA
Enterprise Scalability
Enterprise Authentication
MySQL Connectors
MySQL provides standards-based drivers for JDBC, ODBC, and .Net enabling developers to build database applications in their language of choice. In addition, a native C library allows developers to embed MySQL directly into their applications.
Developed by MySQL
ADO.NET Driver for MySQL (Connector/NET) Download
ODBC Driver for MySQL (Connector/ODBC) Download
JDBC Driver for MySQL (Connector/J) Download
Python Driver for MySQL (Connector/Python) Download
C++ Driver for MySQL (Connector/C++) Download

- На следующей странице выберите дистрибутив, соответствующий вашей платформе:

- Наконец, на следующей странице загрузки нажмите внизу ссылку, позволяющую загрузить выбранный дистрибутив без регистрации (рис. 26). Распакуйте скачанный архив в в какую-либо папку на своем диске. Теперь вы можете добавить в созданный новый проект скачанный драйвер. Запомните, это надо делать для каждого проекта, в котором вы хотите работать с JDBC.

- Выделите в окне инспектора проект, активируйте его свойства и выделите в левой части окна опцию libraries. В правой части следующего окна нажмите кнопку Add JAR/Folder, перейдите в папку, куда вы распаковали скачанный драйвер, выберите файл mysql-connectorjava–5.1.42–bin.jar и нажмите кнопку OK. Возможно, вы скачаете другую версию драйвера, тогда имя файла будет отличаться от приведенного (рис. 27).

- MySQL SUSE Repository
- MySQL Community Server
- MySQL Cluster
- MySQL Router
- MySQL Utilities
- MySQL Shell
- MySQL Workbench
- MySQL Connectors
- Other Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

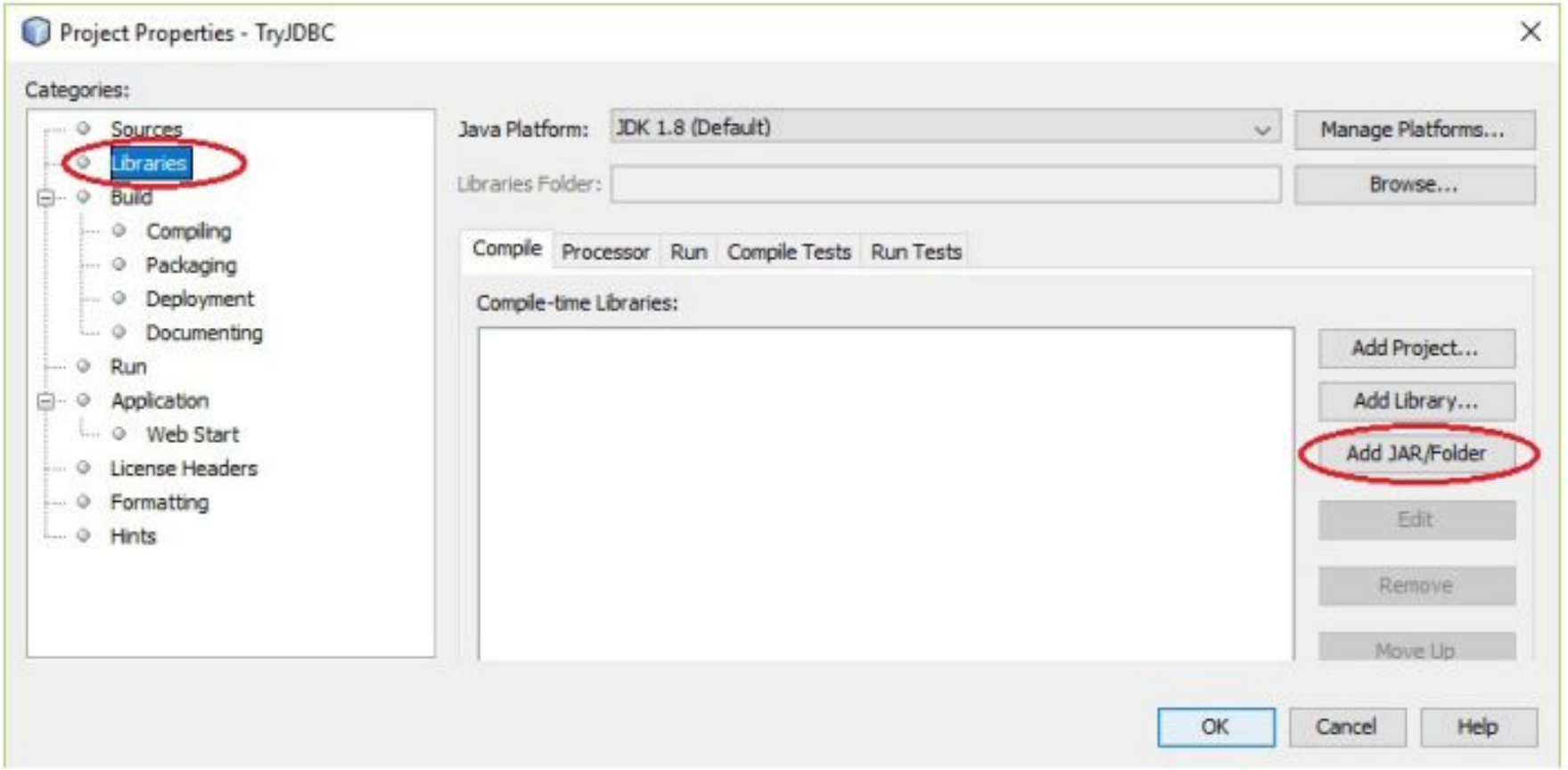
- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system
- Comment in the MySQL Documentation

Login »
using my Oracle Web account

Sign Up »
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.


Project Properties - TryJDBC
Categories:
Sources
Libraries
Build
Compiling
Packaging
Deployment
Documenting
Run
Application
Web Start
License Headers
Formatting
Hints
Java Platform:
JDK 1.8 (Default)
Manage Platforms...
Libraries Folder:
Browse...
Compile
Processor
Run
Compile Tests
Run Tests
Compile-time Libraries:
Add Project...
Add Library...
Add JAR/Folder
Edit
Remove
Move Up
OK
Cancel
Help

- Теперь мы можем перейти к написанию кода. Наше приложение выполнит подключение к созданной БД jtest, создаст в ней таблицу, занесет в нее несколько записей, а затем выведет данные из таблицы в консольное окно.

- Вынесем работу с БД в отдельный класс. В этом классе создадим методы для выполнения соединения с БД и метод для выполнения запросов select. Запросы, которые не возвращают данные из БД, будем выполнять другим способом, вне этого класса. При создании объекта этого класса будем передавать конструктору данные, идентифицирующие адрес сервера, пользователя и имя конкретной БД. Добавьте в состав проекта такой класс:

```
public class DbManager {
    private String host;     //server address
    private String user;     //user name
    private String pass;     //user password
    private String dbName;   //DB name
    private Connection conn; //connection object

    public DbManager(String host, String user,
                     String pass, String dbName)
    {
        this.conn=null;
        this.host=host;
        this.user=user;
        this.pass=pass;
        this.dbName=dbName;

        try
        {
            //driver registration
```

```
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException ex)
        {
            Logger.getLogger(DbManager.class.getName()).
                  log(Level.SEVERE, null, ex);
        }
    }

    public Connection connect()
    {
        String url="jdbc:mysql:
               //"+this.host+"/"+this.dbName+
               "?useUnicode=true&characterEncoding=
               utf-8";
        try
        {
            this.conn=DriverManager.
                getConnection(url,this.user,this.pass);
            return this.conn;
        } catch (SQLException ex)
        {
            Logger.getLogger(DbManager.class.getName()).
                  log(Level.SEVERE, null, ex);
            return null;
        }
    }
```

```
    public ResultSet getSelectQuery(String sql,
        Connection conn)
    {
        Statement comm = null;
        ResultSet set = null;
        try
        {
            comm = (Statement) conn.createStatement();
            set = comm.executeQuery(sql);
        }
         catch (SQLException ex)
         {
             Logger.getLogger(DbManager.class.getName()).
                   log(Level.SEVERE, null, ex);
         }
         return set;
     }
}
```

В конструкторе класса выполняется загрузка MySQL драйвера. Чтобы загрузить драйвер именно для MySQL, методу Class.forName() надо передать в качестве параметра строку с именем драйвера вида: "com.mysql.jdbc. Driver". Это предопределенная строка. Для подключения к другим СУБД она будет другой. Например:

- для MS SQL Server — "com.microsoft.jdbc.sqlserver. SQLServerDriver";
- для Oracle — "oracle.jdbc.driver.OracleDriver";
- для PostgreSQL — "postgresql.Driver".

- Вы всегда можете узнать, как должно выглядеть имя JDBC драйвера на сайте разработчика требуемой вам СУБД.
- Метод connect() выполняет подключение к указанной в строке url базе данных. Обратите внимание, что формат строки url тоже строго фиксирован и уникален для каждого сервера. Для случая MySQL формат этой строки должен быть таким:

```
"jdbc:mysql://host/dbName?useUnicode=
true&characterEncoding=utf-8"
```

- где:
- host — адрес сервера;
- dbName — имя БД;

- Далее в этом методе инициализируется объект Connection, через который потом будут выполняться запросы к БД.
- Теперь перейдем в метод main() и будем вставлять туда фрагменты кода, объясняя их работу, где это необходимо. Добавьте две следующие строки:

```
DbManager db = new DbManager("localhost:3307",
               "root", "123456", "jtest");
Connection conn = db.connect();
```

- Создаем объект класса DbManager, указав ему данные, необходимые для подключения к нашей БД. Затем вызываем метод connect() созданного нами класса DbManager, чтобы получить объект типа Connection. Это один из центральных типов при работе с БД. Создав этот объект, мы сможем с его помощью (вызовом метода createStatement()) создать объект типа Statement. А последний позволит нам

```
Statement statement1 = conn.createStatement();
```

- У объекта Statement есть метод execute(), который принимает строку с SQL запросом и выполняет этот запрос. Мы выполняем запрос, создающий таблицу Student с полями id, name и rate. Чтобы запрос не выполнялся при каждом запуске приложения, мы включили в него фразу if not exists.

```
statement1.execute("create table if not exists
                    Student(" + "id integer primary
                    key auto_increment, " + "name
                    varchar(100), " + "rate double);");
```

- На этом этапе вы можете запустить приложение, и если вы набрали код без ошибок, то приложение успешно выполнится, и в базе данных будет создана таблица Student. Убедитесь, что таблица создана.
- Теперь рассмотрим выполнение запросов insert, чтобы добавить в созданную таблицу несколько записей. Э

```
String s1_name="Corwin of Amber";
    double s1_rate=190;
    String ins1="insert into Student(name, rate)
        values('"+s1_name+"',"+s1_rate+")";
    statement.execute(ins1);
```

- Готовим переменные с требуемыми данными, вставляем эти данные в строку запроса, а затем снова выполняем запрос с помощью метода execute(), как и при создании таблицы. Такой способ выполнения insert будет работать, однако пользоваться им не стоит. Во-первых, вшивать данные в тело запроса, следить при этом за кавычками — работа рутинная. Во-вторых, данные из переменных при этом никак не проверяются, что является брешью для SQL инъекции.

- Более предпочтительным является использование типа PreparedStatement, позволяющего создавать параметризированные запросы, в которых вместо параметров можно указывать плейсхолдеры, а затем связывать с каждым плейсхолдером требуемые данные.

- Объект создается вызовом метода prepareStatement() от имени того же объекта Connection. В качестве параметра методу prepareStatement() надо передать строку запроса, в которой вместо каждого вводимого значения можно указать плейсхолдер «?». Затем, учитывая тип значения, вызовами методов setBoolean(), setString(), setDouble(), setInt() и другими подобными надо привязать значение для каждого плейсхолдера, указывая первым параметром номер плейсхолдера (начиная с 1).

```
String s2_name="Han Solo";
double s2_rate=180;
PreparedStatement statement2 = conn.prepareStatement
    ("insert into Student(name,rate) values(?,?)");
statement2.setString(1, s2_name); //first value is of
                                  //type String
statement2.setDouble(2, s2_rate); //second value is
                                  //of type Double
statement2.executeUpdate();
```

У вас может возникнуть вопрос, чем PreparedStatement отличается от Statement. Отличия есть и они существенны:

- PreparedStatement выполняются значительно быстрее, чем Statement, потому, что такие запросы компилируются;
- данные для Statement являются статическими, в то время как для PreparedStatement их можно добавлять динамически;
- PreparedStatement автоматически выполняют защиту от SQL инъекций;
- запросы, использующие PreparedStatement, выглядят значительно проще.

- Поэтому рекомендуется всегда использовать PreparedStatement.
- Если вы выполните сейчас наше приложение, то обнаружите в таблице вставленные строки. Чтобы эти строки не добавлялись каждый раз при запуске приложения, закомментируйте вызовы метода execute(), выполняющие запросы insert. Теперь перед нами стоит задача прочитать данные из таблицы и вывести их в консольное окно. Для этого мы будем работать с типом ResultSet. Это аналог SqlDataReader в ADO.NET

- ResultSet надо использовать при выполнении SQL запросов select. Данные, возвращаемые запросом select, вставляются в тип ResultSet, откуда их можно затем извлекать в цикле. Обратите внимание на методы getString(), getDouble() и другие аналогичные, позволяющие извлекать прочитанные данные согласно их типу. Мы передаем этим методам строковые имена полей из таблицы, однако мы могли бы указывать порядковые номера требуемых полей (нумерация начинается с 1):

```
String sel="select * from Student";
ResultSet set=db.getSelectQuery(sel, conn);
while(set.next()){
    System.out.println(set.getInt("id")+" "+
        set.getString("name")+" "+
            set.getDouble("rate"));
}
```

- Добавьте этот код и запустите приложение. У меня вывод получился

```
1 Corwin of Amber 190.0
2 Han Solo 180.0
```

- Как видите, данные из нашей формы прочитаны и выведены в консольное окно. Обратите внимание на блок finally, в котором выполняется закрытие созданного соединения.