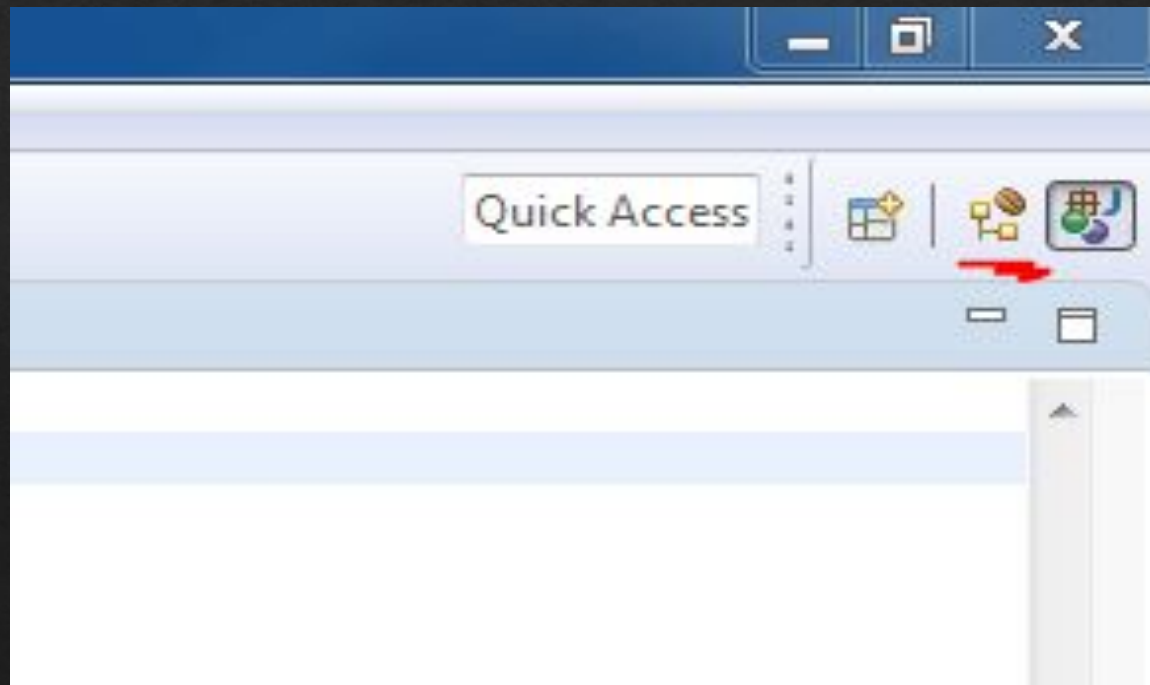


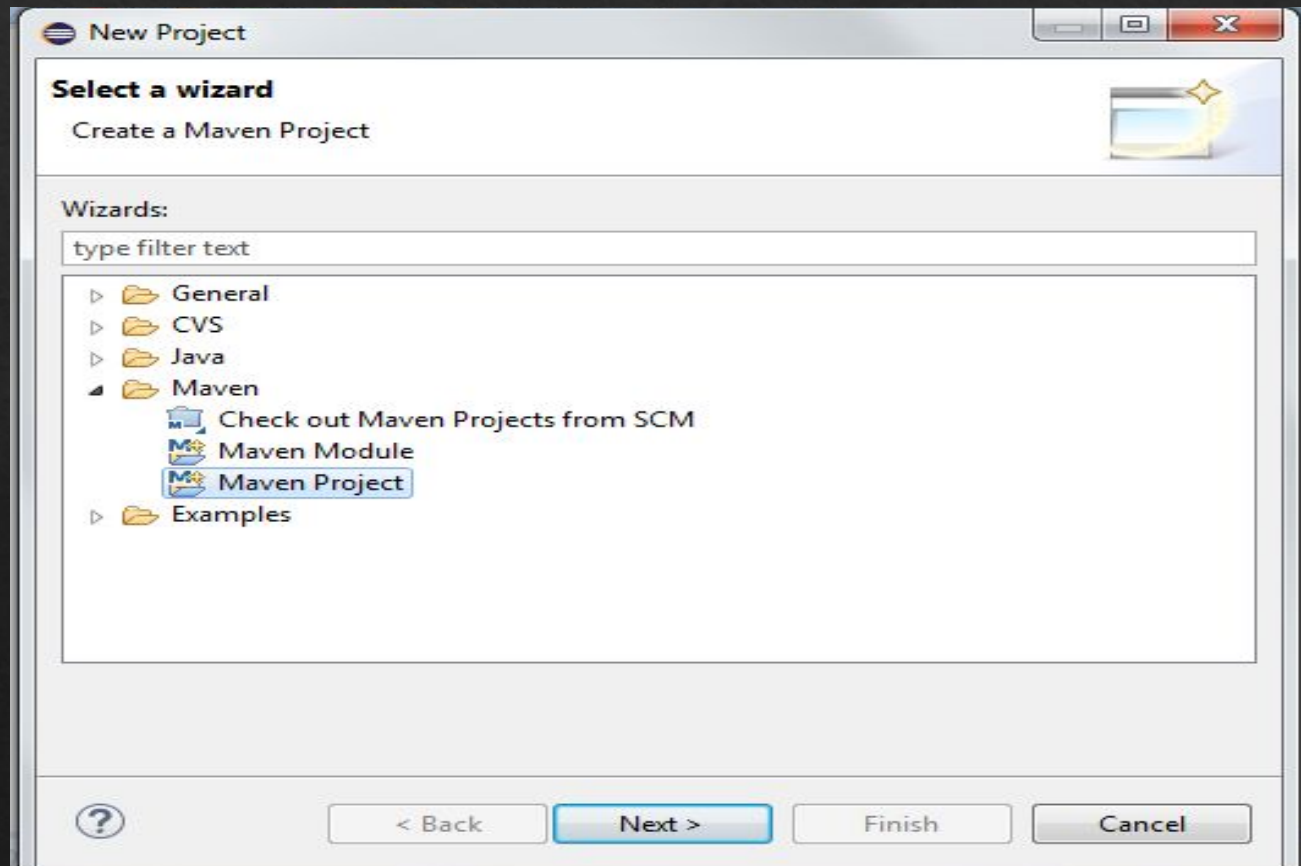
# WEB java



В Eclipse нужно выбрать с **Java** на **Java EE** (серверная платформа) в правом верхнем углу. (проверить, чтобы так же было)



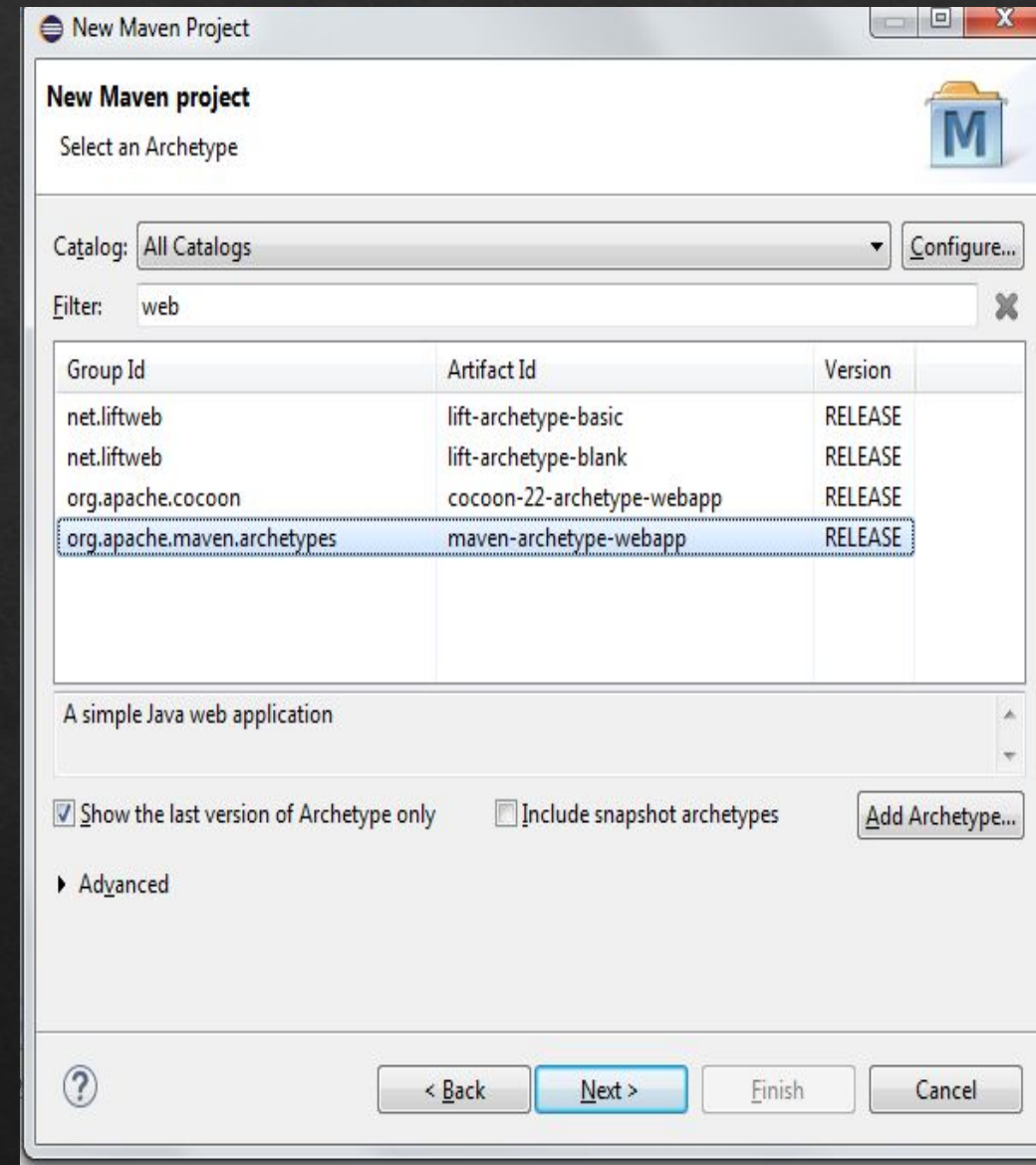
Теперь нужно создать новый **Maven** проект. **File** -> **new** -> **other...** -> **maven** -> **Maven Project**



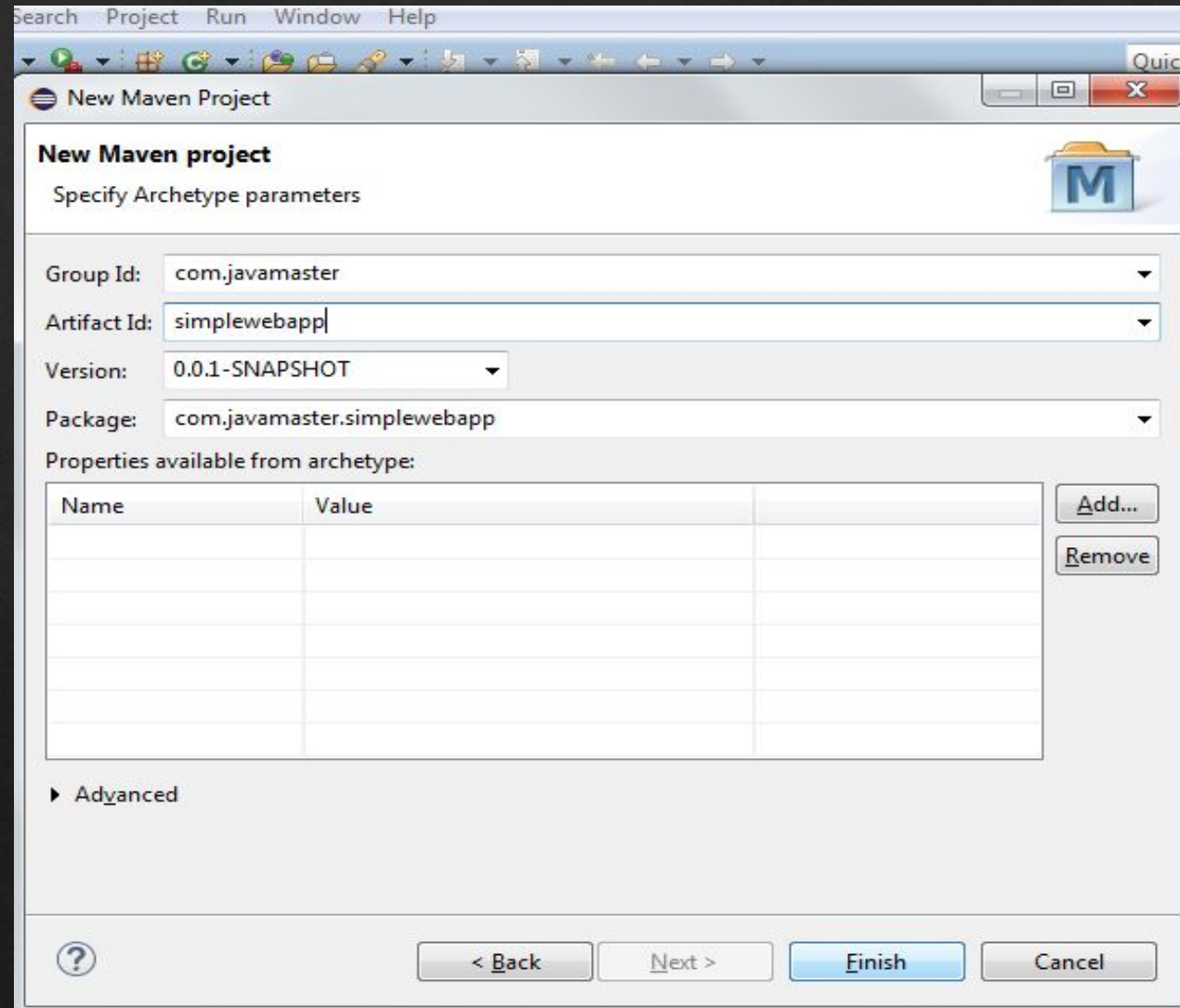


Нажимаем **Next** и далее нас перебрасывает в окно выбора так называемого архетипа (**archetype**). Мы используем **Maven** как инструмент сборки потому, он практичен, лёгок в усвоение. Нам не нужно больше искать дополнительные **jar** библиотеки в Интернете (как с **mp3** или **jdbc**), качать их и подключать. **Maven** позволяет нам подключать дополнительные библиотеки в специальном файле настроек, который называется **pom.xml**.

При выборе архетипа нужно выбрать **maven-archetype-webapp**:



Далее появится окно ввода **Group id**, **Artifact id**. В строку **Group id** введите **com.javamaster**, а в строку **Artifact id** - например, **simplewebapp**.



**New Maven Project**

Specify Archetype parameters

Group Id:

Artifact Id:

Version:

Package:

Properties available from archetype:

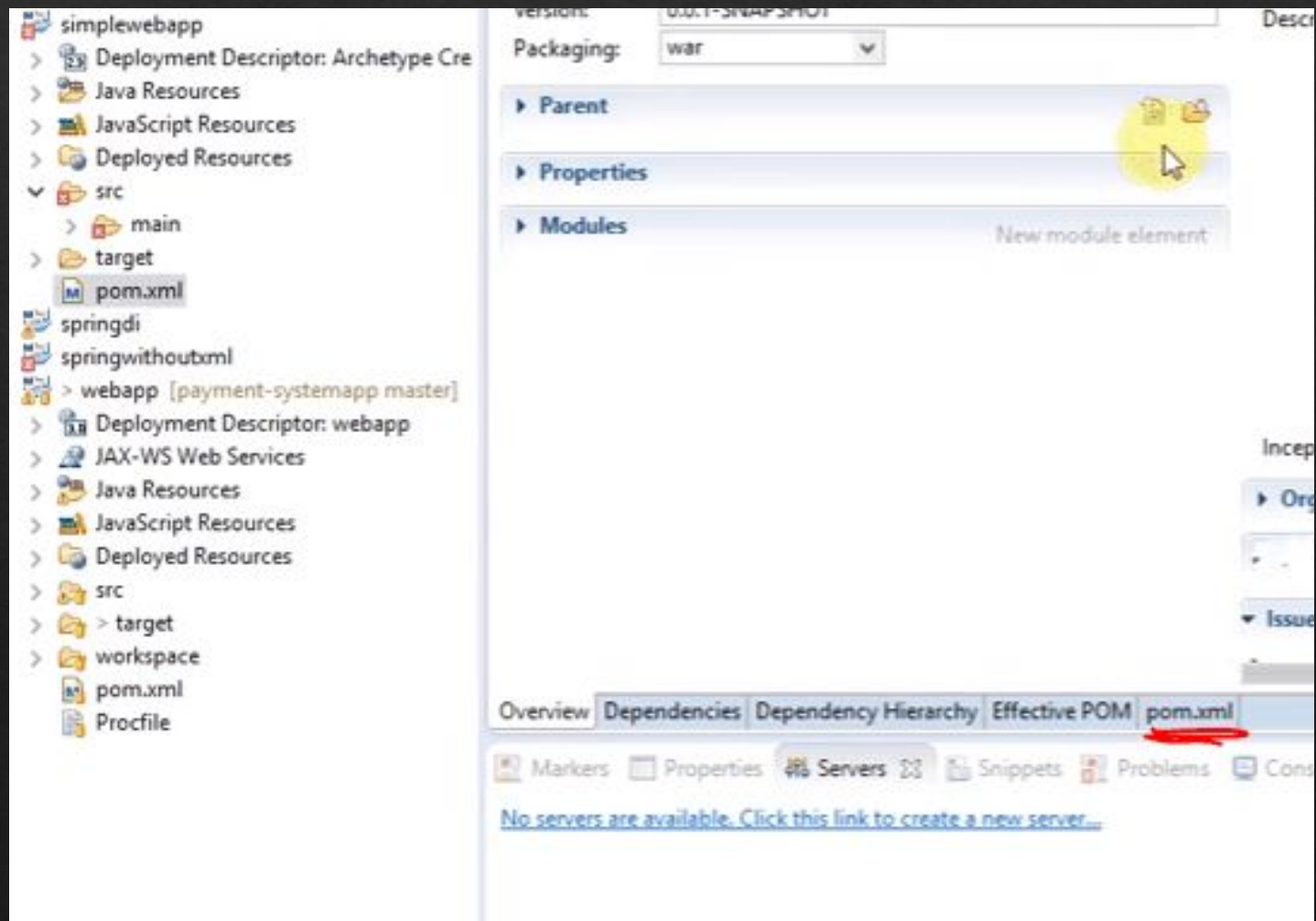
Name	Value

► Advanced

Нажимаем **Finish** и видим, что в панели проектов нам создало новый проект с названием нашего **Artifact id**. В проекте много файлов и папок по умолчанию.



Далее (если присутствует красный крестик, теоретически приложение должно запускаться, но если по каким-то причинам ошибка, то тогда) —  
Откройте файл **pom.xml**, перейдите на вкладку кода:



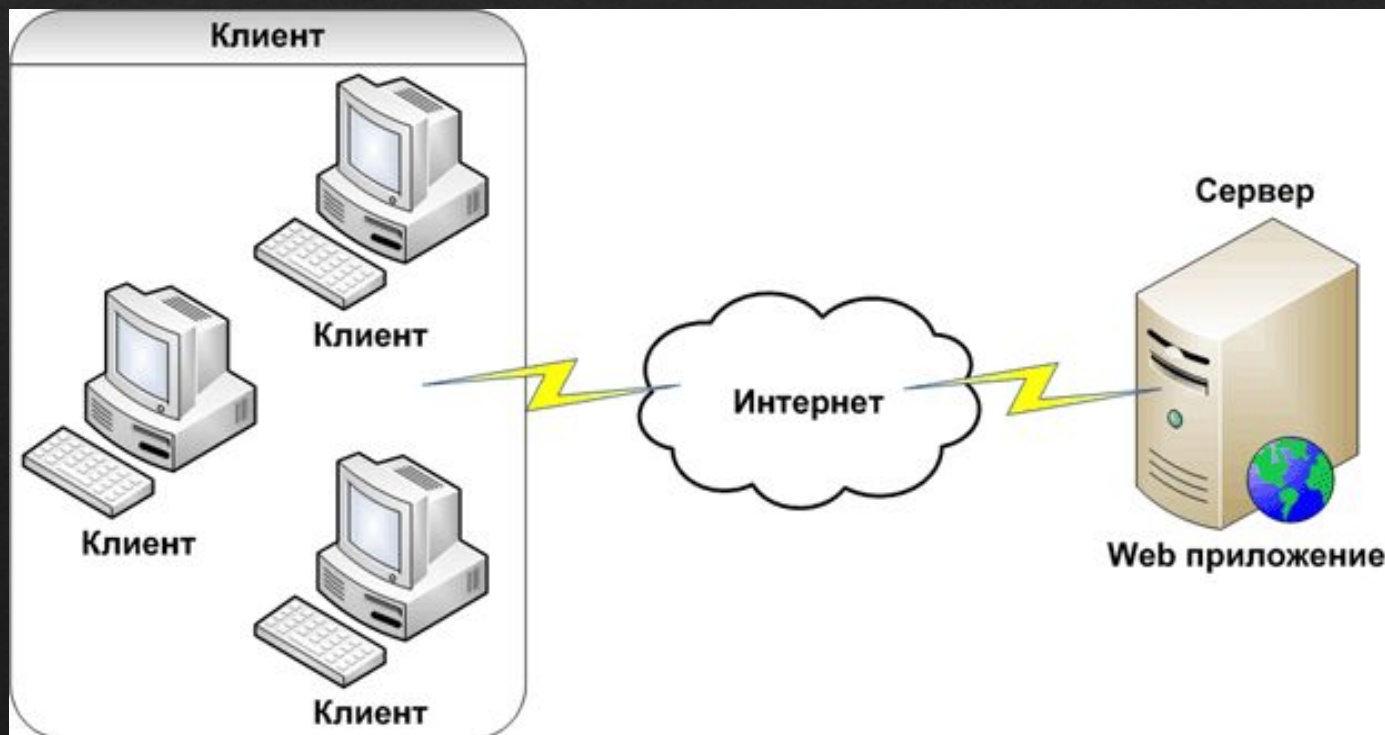
Далее в раздел **dependencies** (зависимости) нужно добавить (ниже основного dependency. То есть должно получиться **dependency ... dependency** и **dependency... dependency**):

```
<dependency>  
    <groupId>javax</groupId>  
    <artifactId>javaee-web-api</artifactId>  
    <version>6.0</version>  
    <scope>provided</scope>  
</dependency>
```

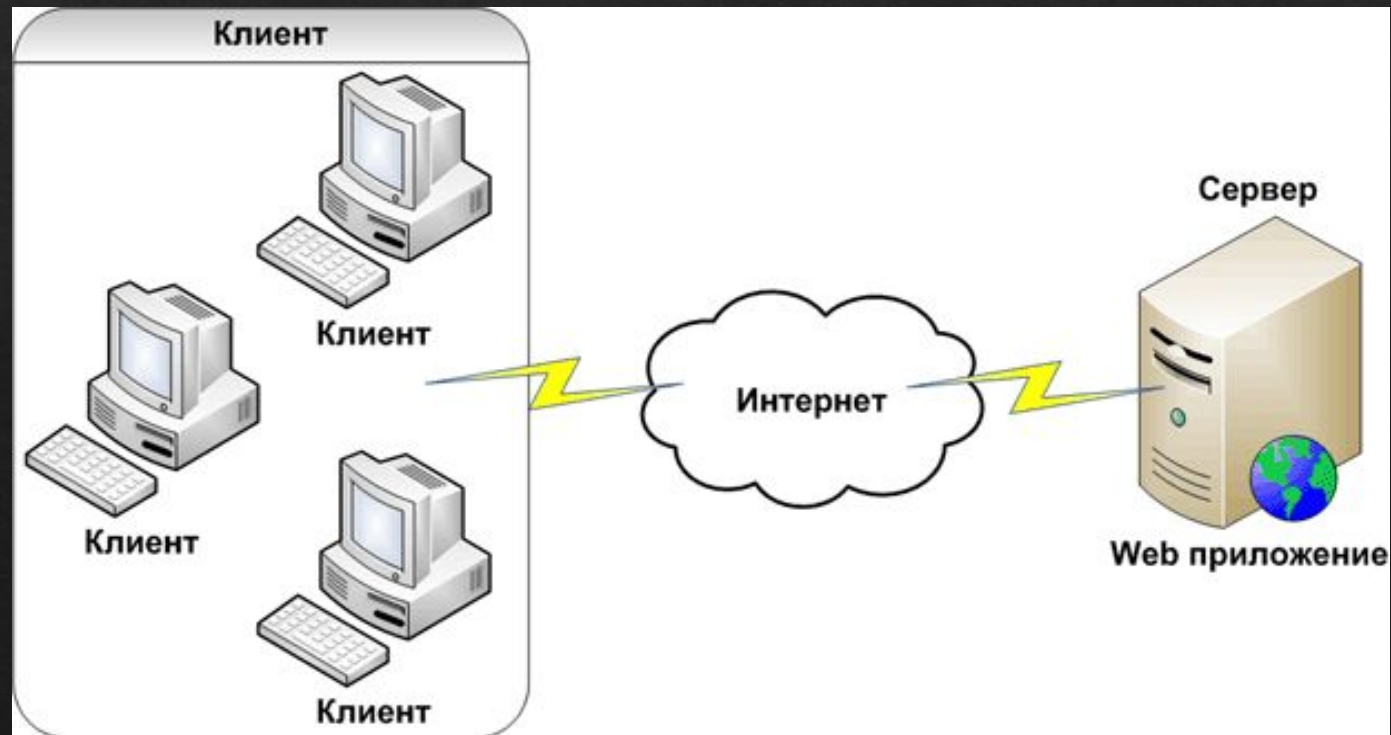


Этим действием мы добавили новую библиотеку к нашему проекту. Ничего не нужно скачивать: добавление нескольких строчек в **pom.xml** укажет программе автоматически загрузить данные библиотеки. Сохраните файл и проблема с ошибкой проекта должна исчезнуть.

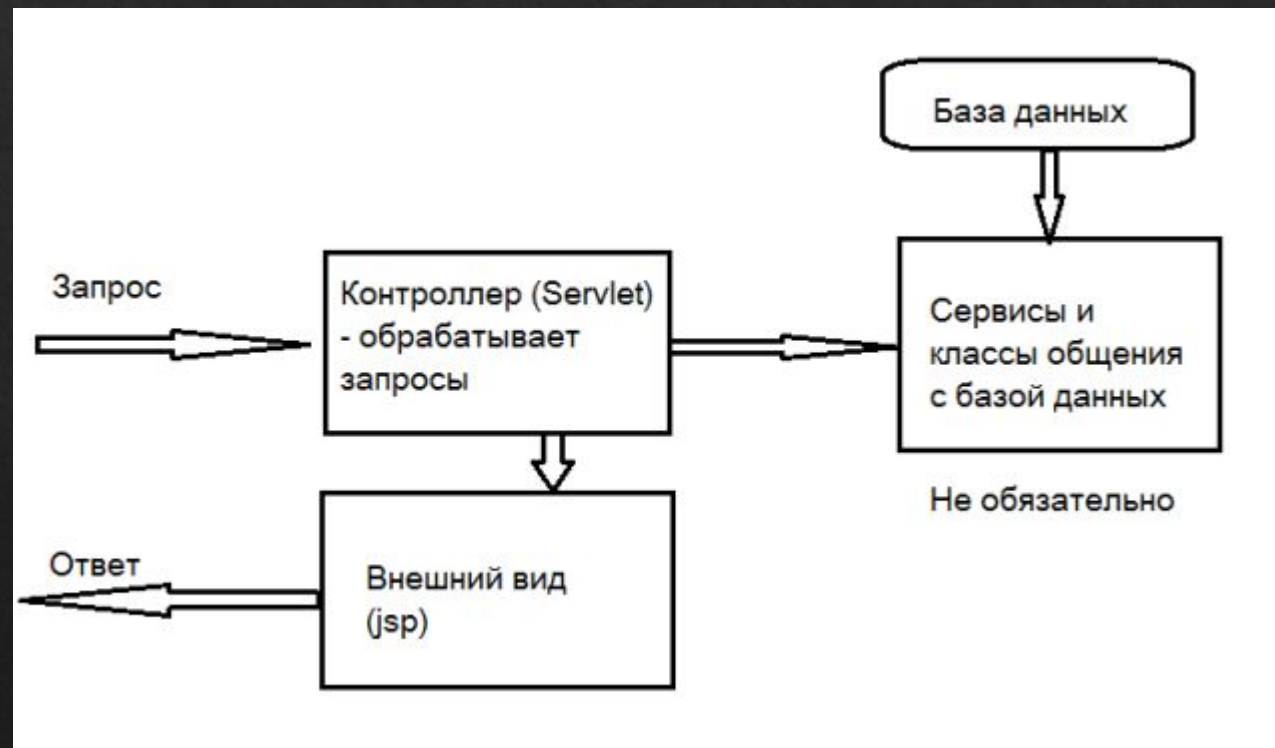
Теперь необходимо рассмотреть, как работает веб и в частности, веб приложения.



На рисунке представлена схема работы интернет приложений: клиент, то есть Ваш браузер, посылает запрос по определенному адресу и порту; через Интернет запрос доходит до Вашего сервера, на котором находится приложение; обработанный запрос возвращается клиенту в виде ответа; браузер интерпретирует ответ с сервера в понятный для человека вид картинок или текст.



## Схема веб программы:

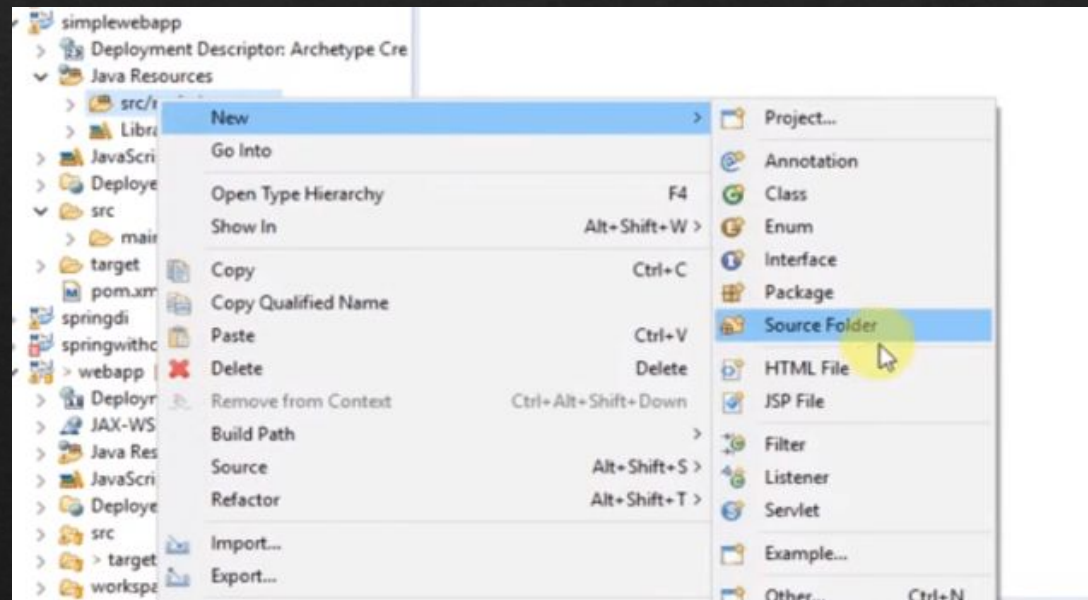




Для распознавания запросов в **Java** есть такой механизм как сервлет (**Servlet**) — он может определить строку запроса и перенаправить его на **jsp.jsp** — технология, которая позволяет динамически генерировать веб страницы. По сути **jsp** очень похож на **HTML** с тем отличием, что в нём можно запускать Java код.

**Сервлет** — это класс, который унаследован от **HttpServlet**. В классе **HttpServlet** есть несколько методов по обработке запросов. Нас пока будут интересовать **doGet** и **doPost**, который обрабатывают соответственно гет и пост запросы.

В проекте нужно найти папку **src/main/java**. Если таковой нет, нужно её создать: правой кнопкой на **src** -> **new** -> **source folder**.



Далее нужно добавить в эту папку новый пакет (package). например - **com.javamaster.controller**

Теперь создаём **Servlet** (с кодом ниже), например - **HomeServlet**.

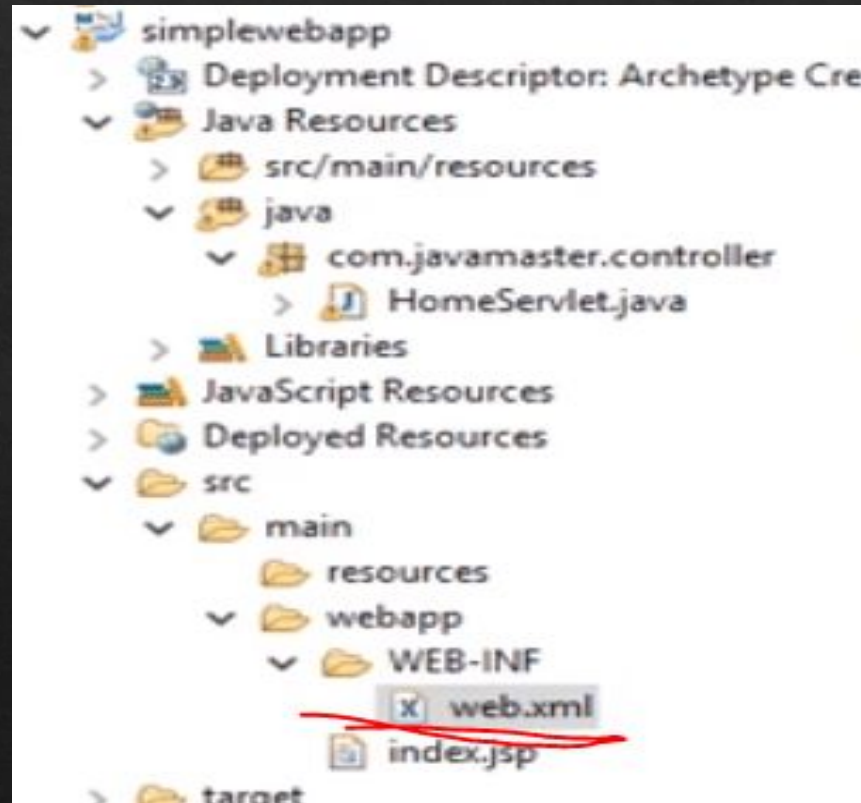
```
package com.javamaster.controller;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class HomeServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
    }}
```

В коде только один метод: **doGet** на данном этапе этого достаточно.

Теперь нужно дать нашей программе понять, что класс **HomeServlet** является контроллером и что все запросы нужно отправлять на него.

Теперь нужно дать нашей программе понять, что класс `HomeController` является контроллером и что все запросы нужно отправлять на него.

Для этого у нас есть `web.xml` — файл настроек нашей программы. Его можно найти в папке `WEB-INF`.



И заменим код:

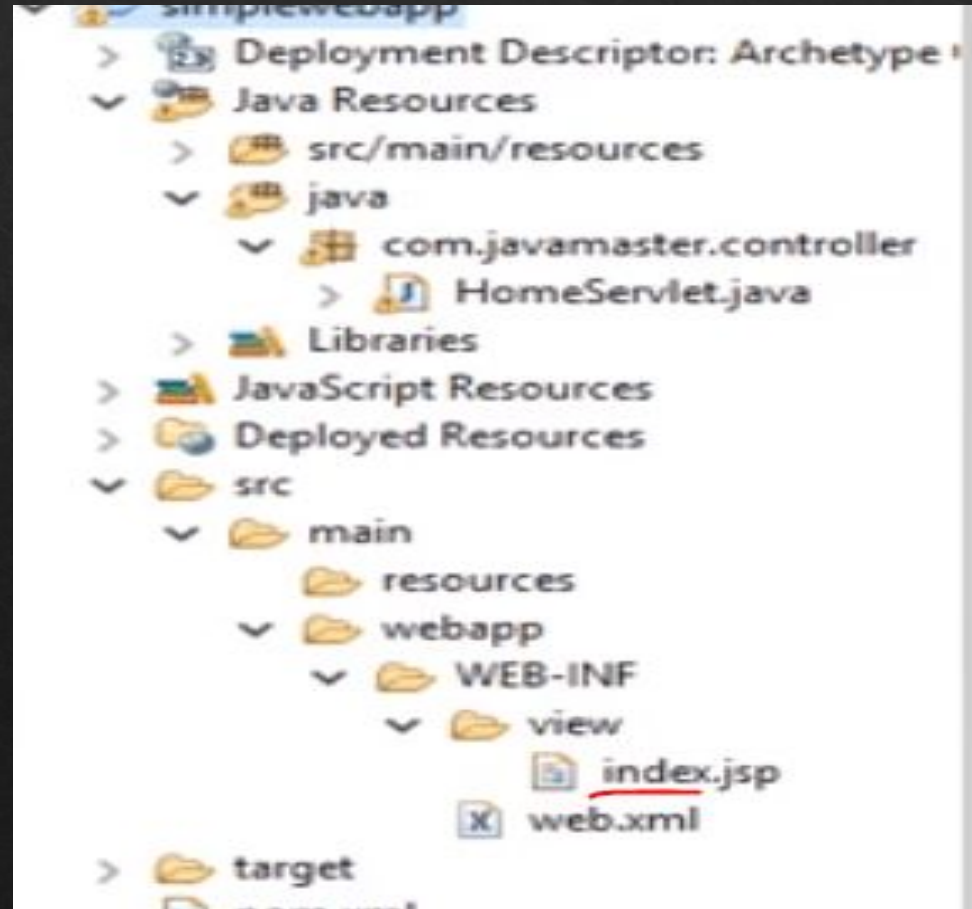


```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0">
  <display-name>Archetype Created Web Application</display-name>
  <servlet>
    <description></description>
    <display-name>HomeServlet</display-name>
    <servlet-name>HomeServlet</servlet-name>
    <servlet-class>com.javamaster.controller.HomeServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HomeServlet</servlet-name>
    <url-pattern>/</url-pattern></servlet-mapping></web-app>
```

В коде, мы указали что передаём управление урл путями нашему классу **HomeServlet**.

Теперь осталось добавить файл, который будет отвечать за внешний вид. Система уже создала нам файл **index.jsp** и проделала всю работу за нас.

Добавьте новую папку **view** в папку **WEB-INF**. Когда появится много файлов, чтобы не запутаться мы поместим все файлы, которые отвечают за внешний вид в папку **view**. Переместите файл **index.jsp** в только что созданную папку.





Теперь осталось отловить  
запрос и перенаправить его на  
нашу страницу **index.jsp**. Для этого  
в сервлете метод **doGet** нужно  
немного изменить.

```
public class HomeServlet extends HttpServlet {  
    @Override  
    protected void doGet(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        String path = request.getServletPath();  
        if (path.equals("/")) {  
  
            request.getRequestDispatcher("/WEB-INF/view/index.jsp").forward(request, response);  
        }  
        else if (path.equals("/welcome")) {  
  
            request.getRequestDispatcher("/WEB-INF/view/welcome.jsp").forward(request, response);  
        }  
    }  
}
```

если запрос равен /, тогда указываем путь, куда перенаправить запрос: на страницу индекс, если же запрос равен /welcome тогда, перенаправить на страницу приветствия.

Теперь осталось только запустить наше приложение и поднять сервер.

Нам нужен контейнер сервлетов, который сможет управлять нашей программой и который реализует Java Servlets и Java Server Pages (jsp) спецификации. Для наших нужд отлично подойдет **Apache Tomcat** — он бесплатный и очень удобный в использовании.

Apache Tomcat нужно скачать с официального сайта, распаковать и указать в настройках Eclipse путь к распакованному архиву. Сделаем все по порядку.

# Качаем архив с сайта:

Other mirrors:

**7.0.78**

Please see the [README](#) file for packaging information. It explains what every distribution contains.

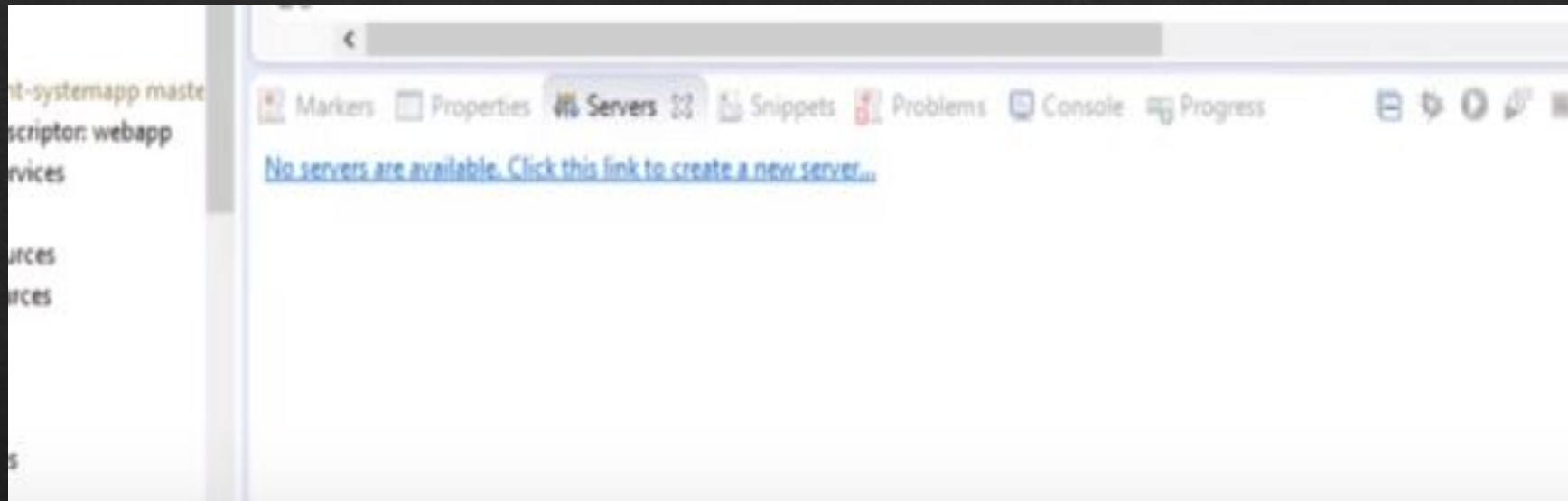
**Binary Distributions**

- Core:
  - [zip \(pgp, md5, sha1\)](#)
  - [tar.gz \(pgp, md5, sha1\)](#)
  - [32-bit Windows zip \(pgp, md5, sha1\)](#)
  - [64-bit Windows zip \(pgp, md5, sha1\)](#)
  - [32-bit/64-bit Windows Service Installer \(pgp, md5, sha1\)](#)
- Full documentation:
  - [tar.gz \(pgp, md5, sha1\)](#)
- Deployer:
  - [zip \(pgp, md5, sha1\)](#)



Распаковываем его в любое место папку на диске.

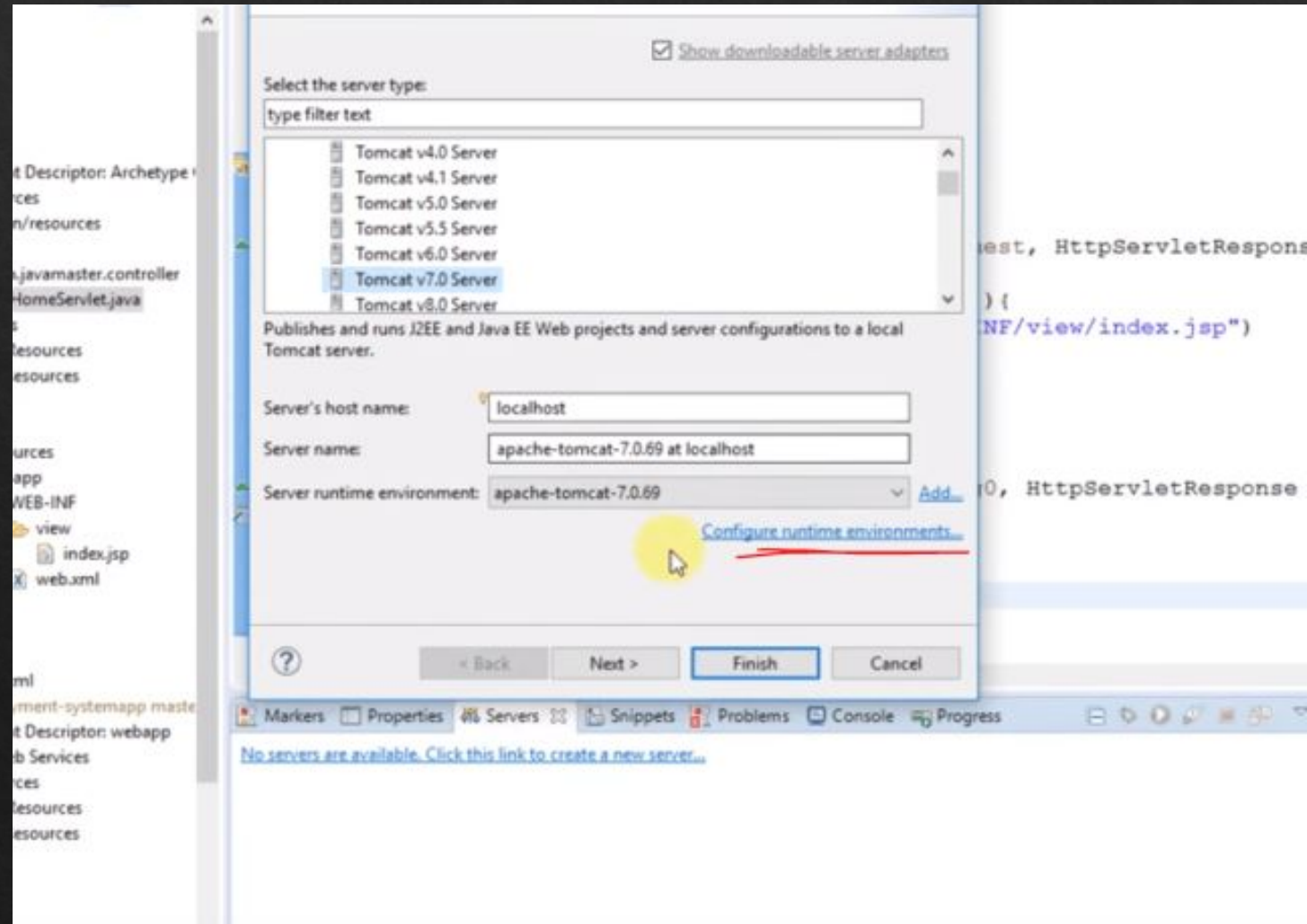
Переходим в Eclipse и во вкладке сервера нажимаем создать новый:



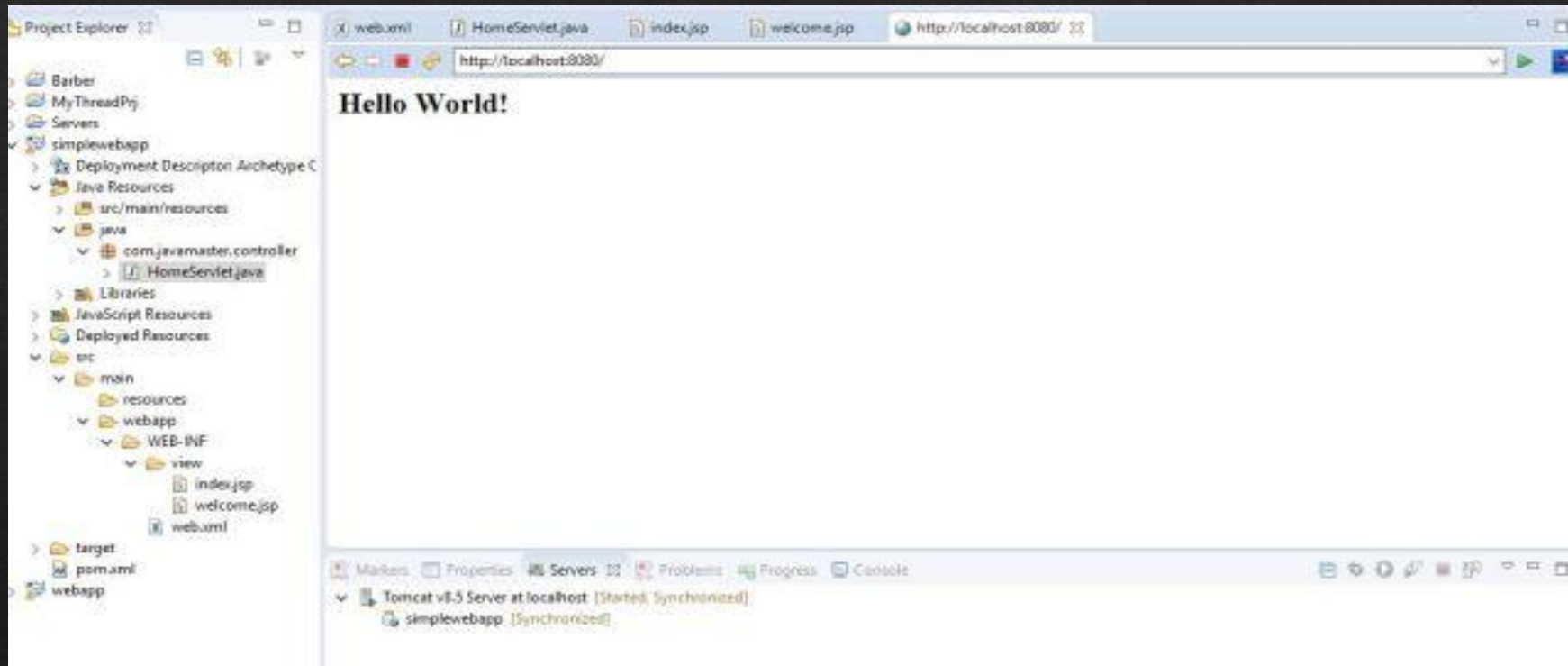
Если такой вкладки у Вас нет, её нужно добавить: **Window -> View ->**

## **Servers**

Теперь нужно выбрать нужный нам вариант из предложенных серверов, указать путь к нашему и готово.



Переходим к нашему проекту. В разделе **Project explorer** нужно нажать правой клавишей мыши и выбрать: **run as** -> **run on server**. Далее в окошке выбрать добавленный сервер подождать, пока проект запустится. Если у Вас нет опции **run on server**, её нужно добавить: **run as** -> **run configuration** -> **run on server**.












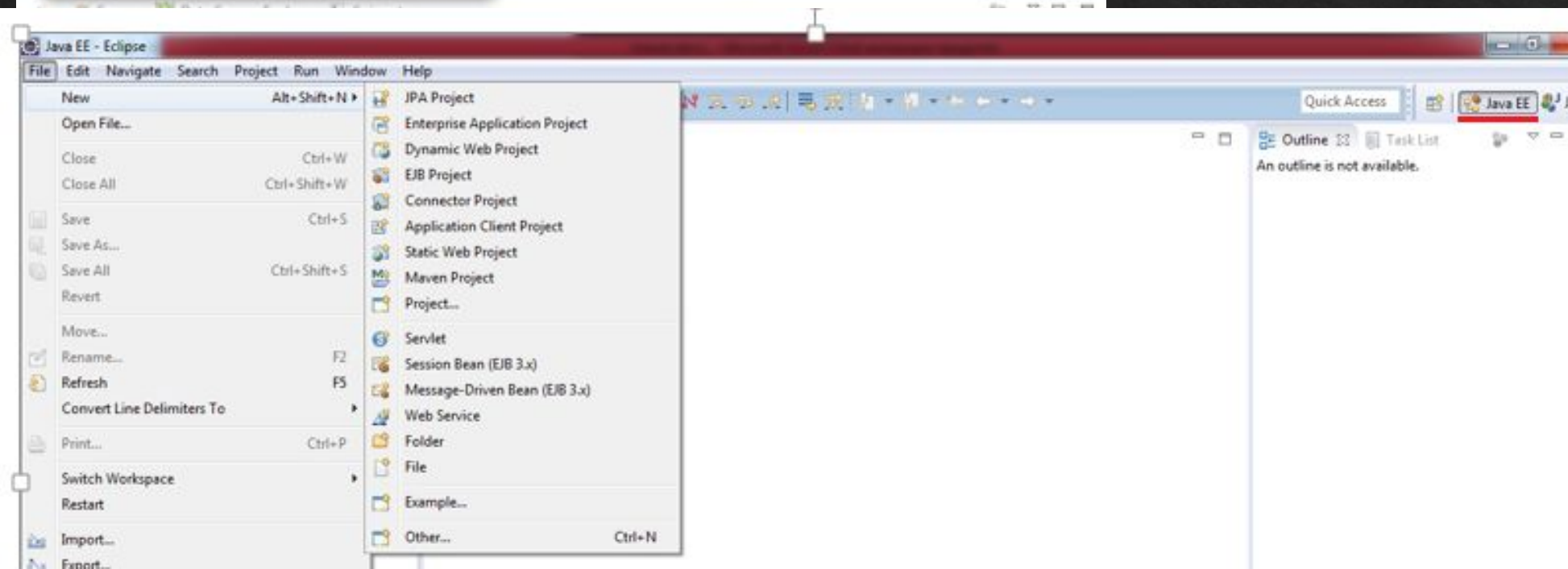
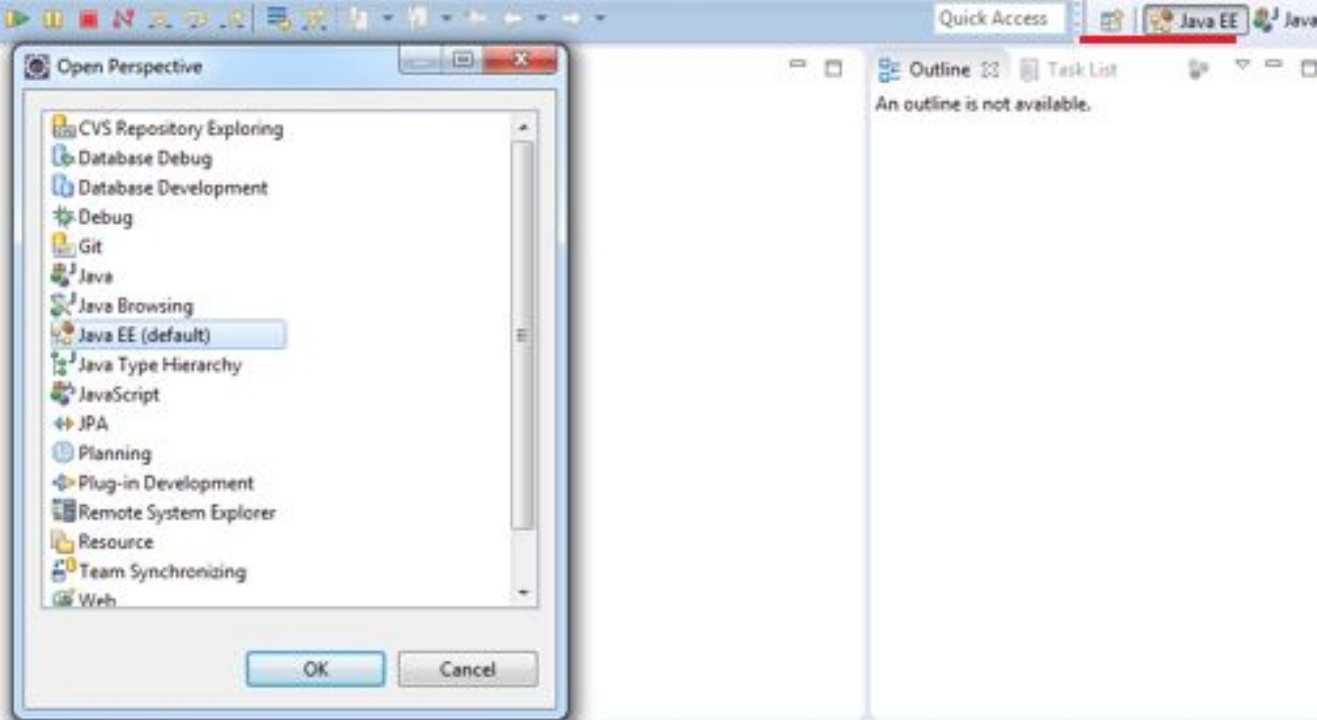
# Информация для размышления

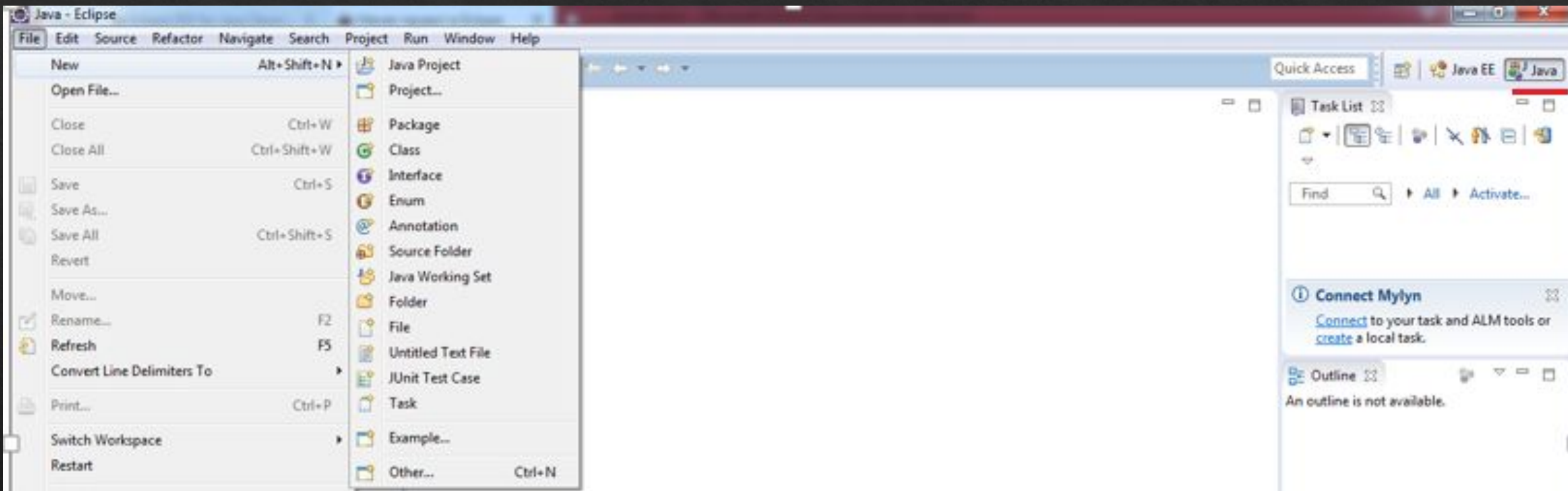
## Нужен Java EE Developers

### Eclipse IDE Luna SR2 Packages

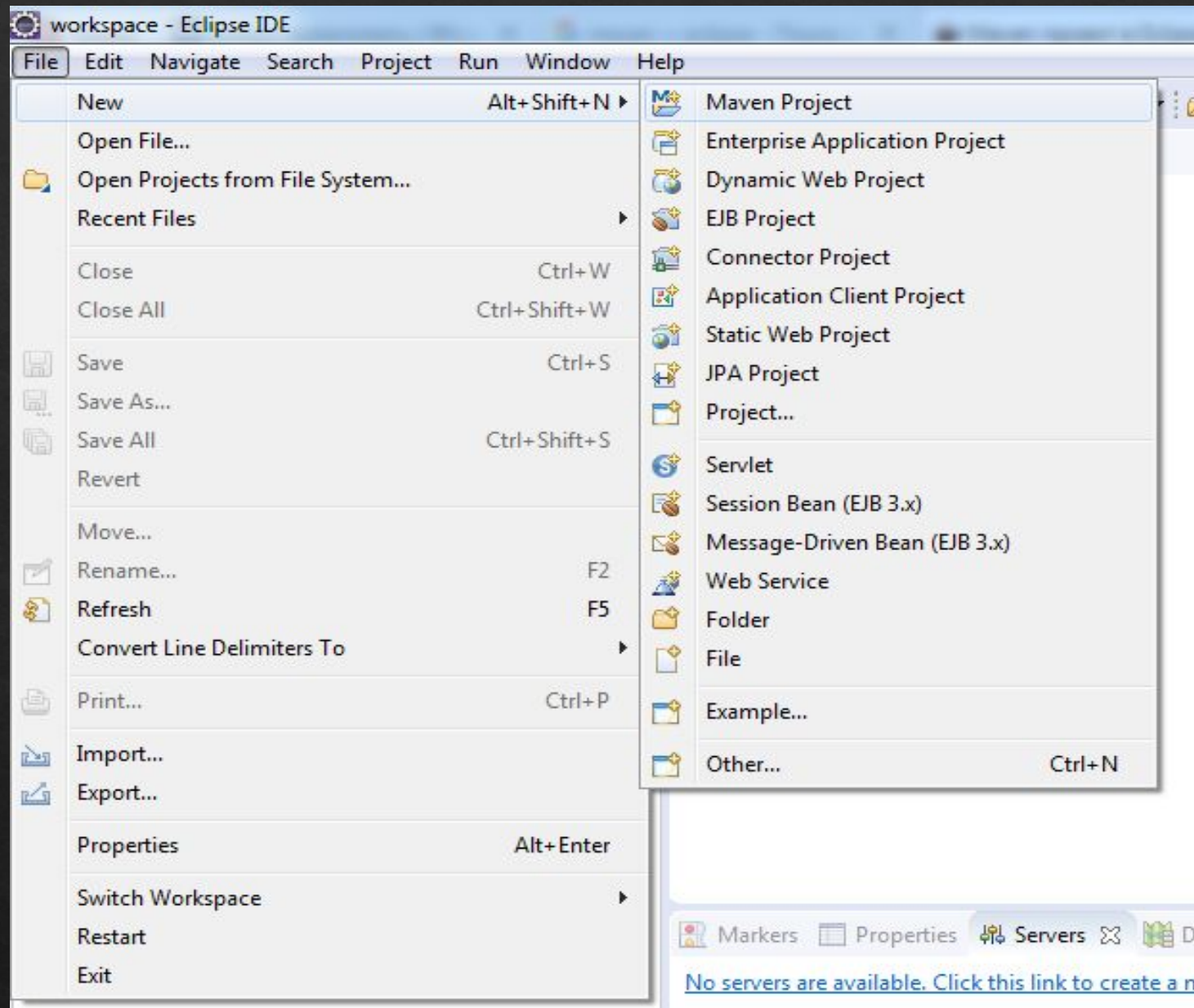
 <b>Eclipse IDE for Java Developers</b> 155 MB 4,159,396 DOWNLOADS The essential tools for any Java developer, including a Java IDE, a CVS client, Git client, XML Editor, Mylyn, Maven Integration and WindowBuilder	 Windows 32-bit 64-bit Mac Cocoa 32-bit 64-bit Linux 32-bit 64-bit
 <b>Eclipse IDE for Java EE Developers</b> 254 MB 2,276,213 DOWNLOADS Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn, EGit and others.	 Windows 32-bit 64-bit Mac Cocoa 32-bit 64-bit Linux 32-bit 64-bit
 <b>Eclipse IDE for C/C++ Developers</b> 365 MB 644,495 DOWNLOADS An IDE for C/C++ developers with Mylyn integration.	 Windows 32-bit 64-bit Mac Cocoa 32-bit 64-bit Linux 32-bit 64-bit

  
**Get Eclipse IDE 2019-12**  
Install your favorite desktop IDE packages.  
[Download 64 bit](#)



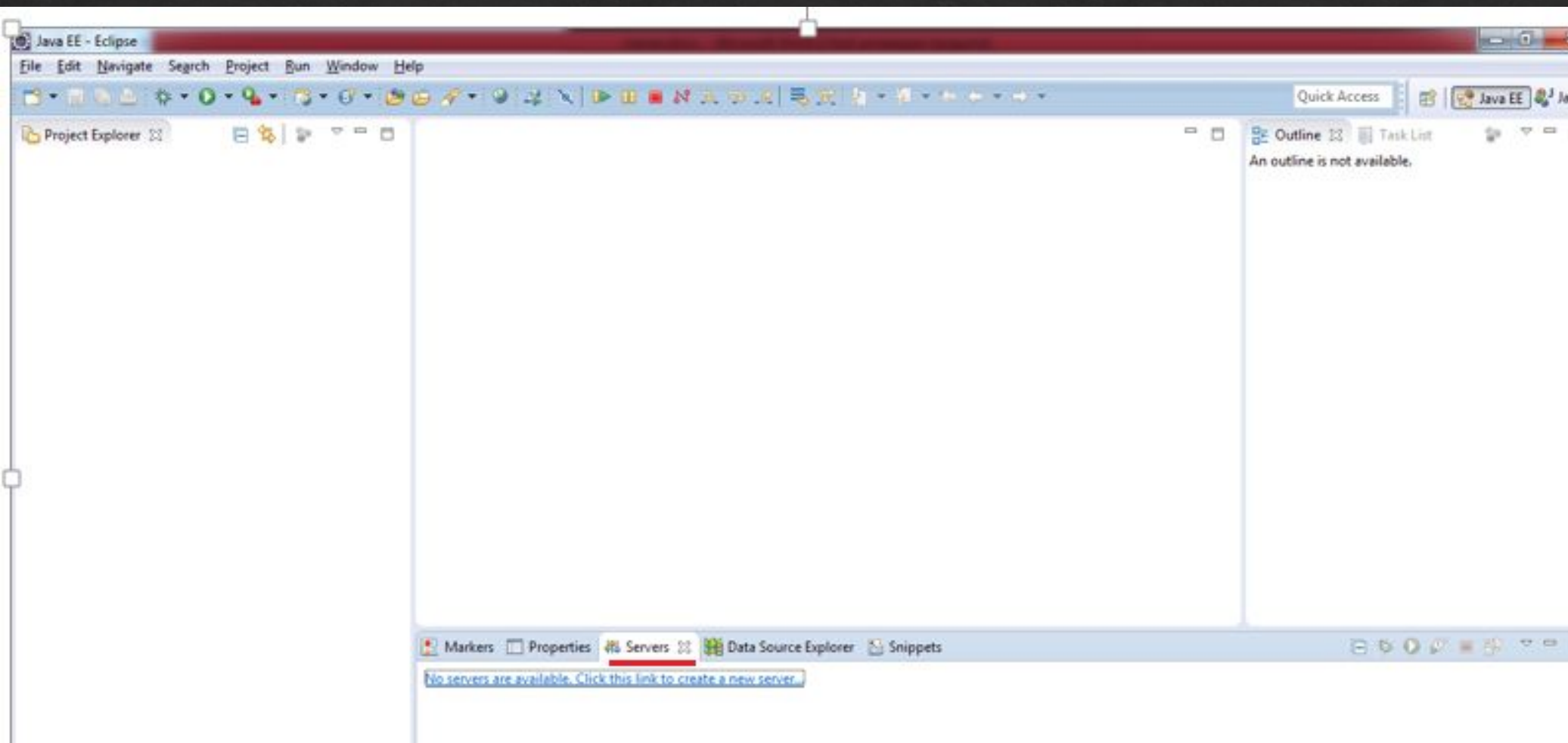


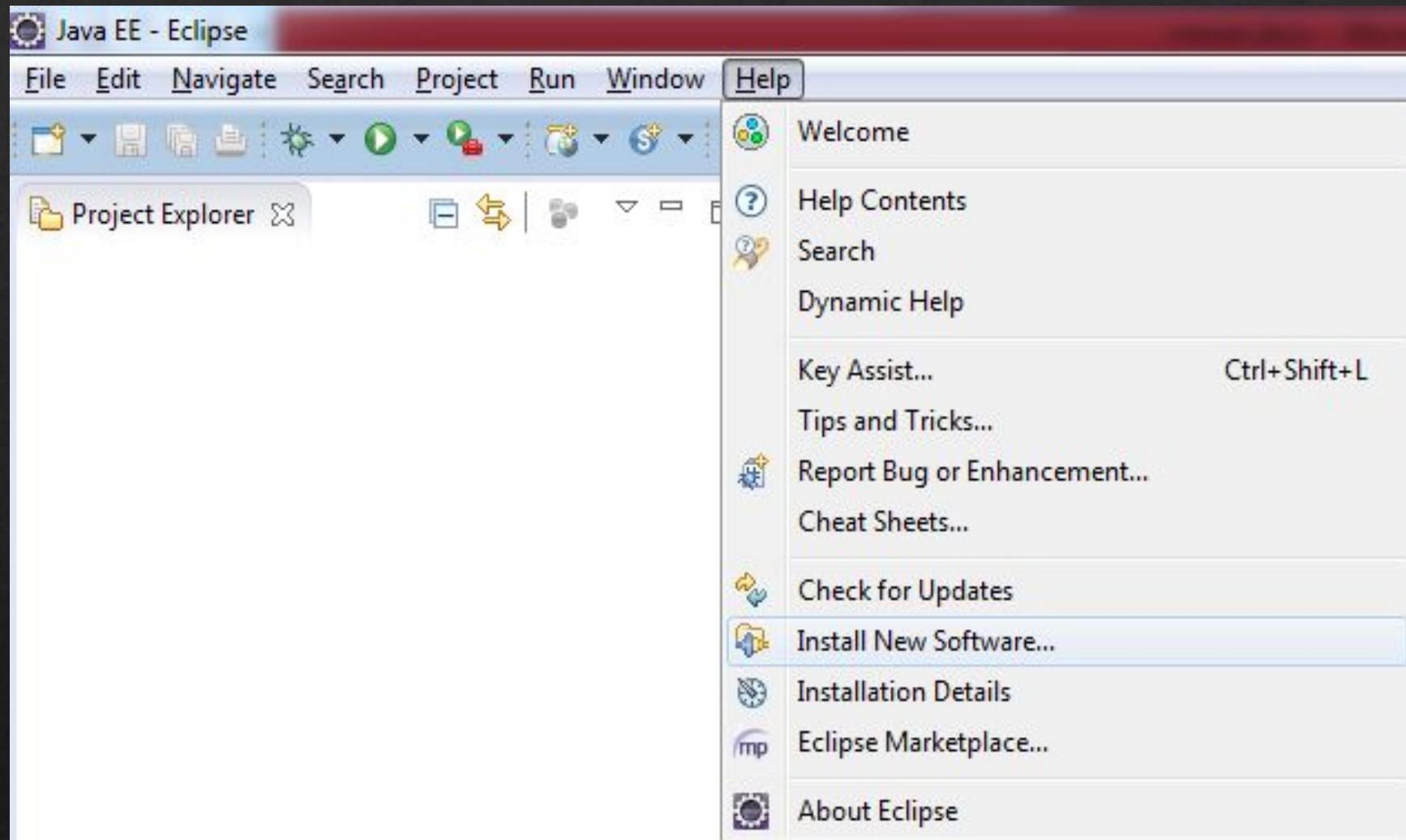




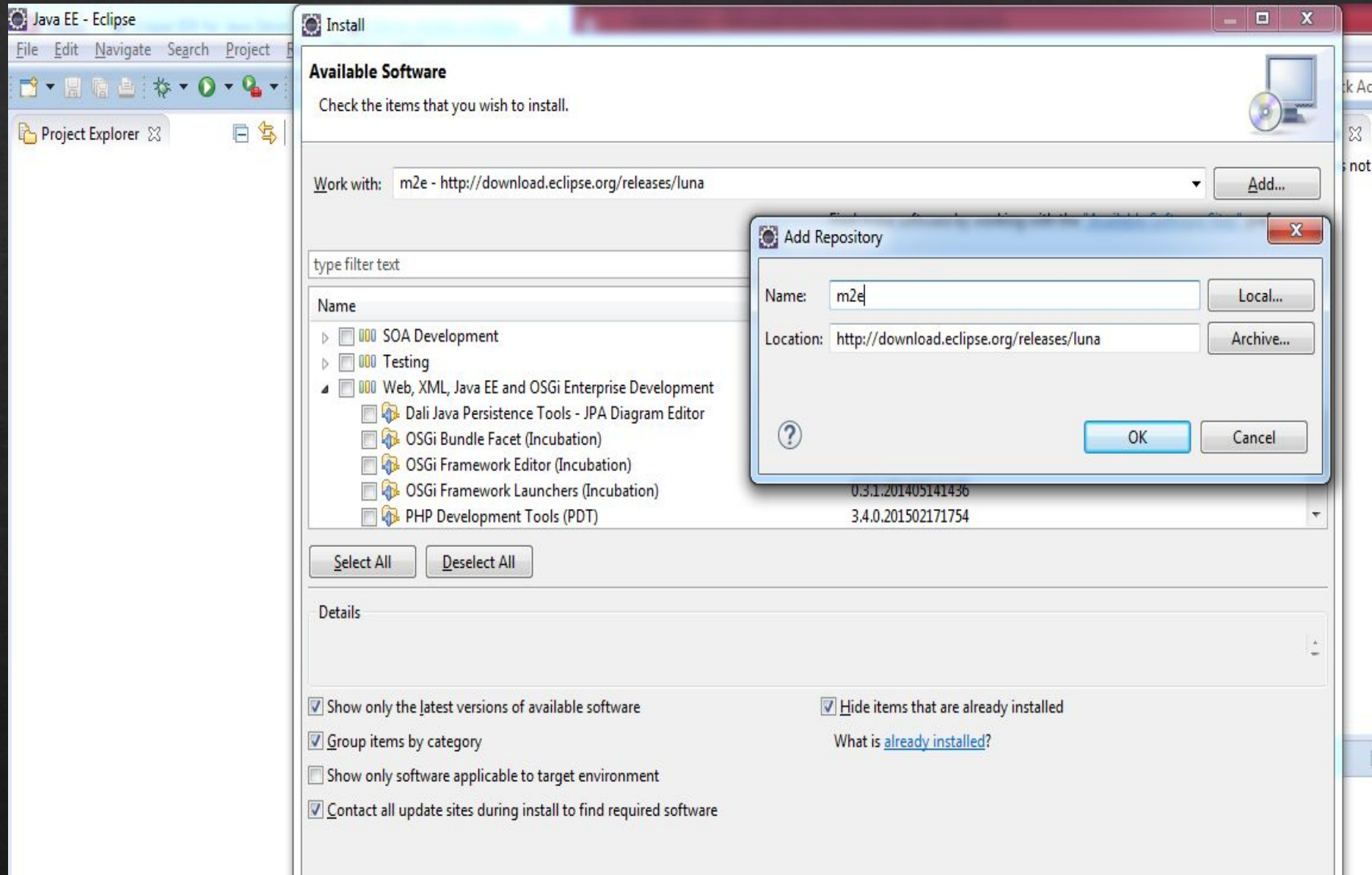
+есть Servers.

Если нет, то открыть его в  
настройках












У меня нет то что  
надо

**Available Software**

Check the items that you wish to install.

Work with:

Find more software by working with the ["Available Software Sites"](#) preferences.

Name	Version
<input checked="" type="checkbox"/>  Web, XML, Java EE and OSGi Enterprise Development	
<input checked="" type="checkbox"/>  m2e connector for mavenarchiver pom properties	0.16.0.20140121145...

1 item selected


Details

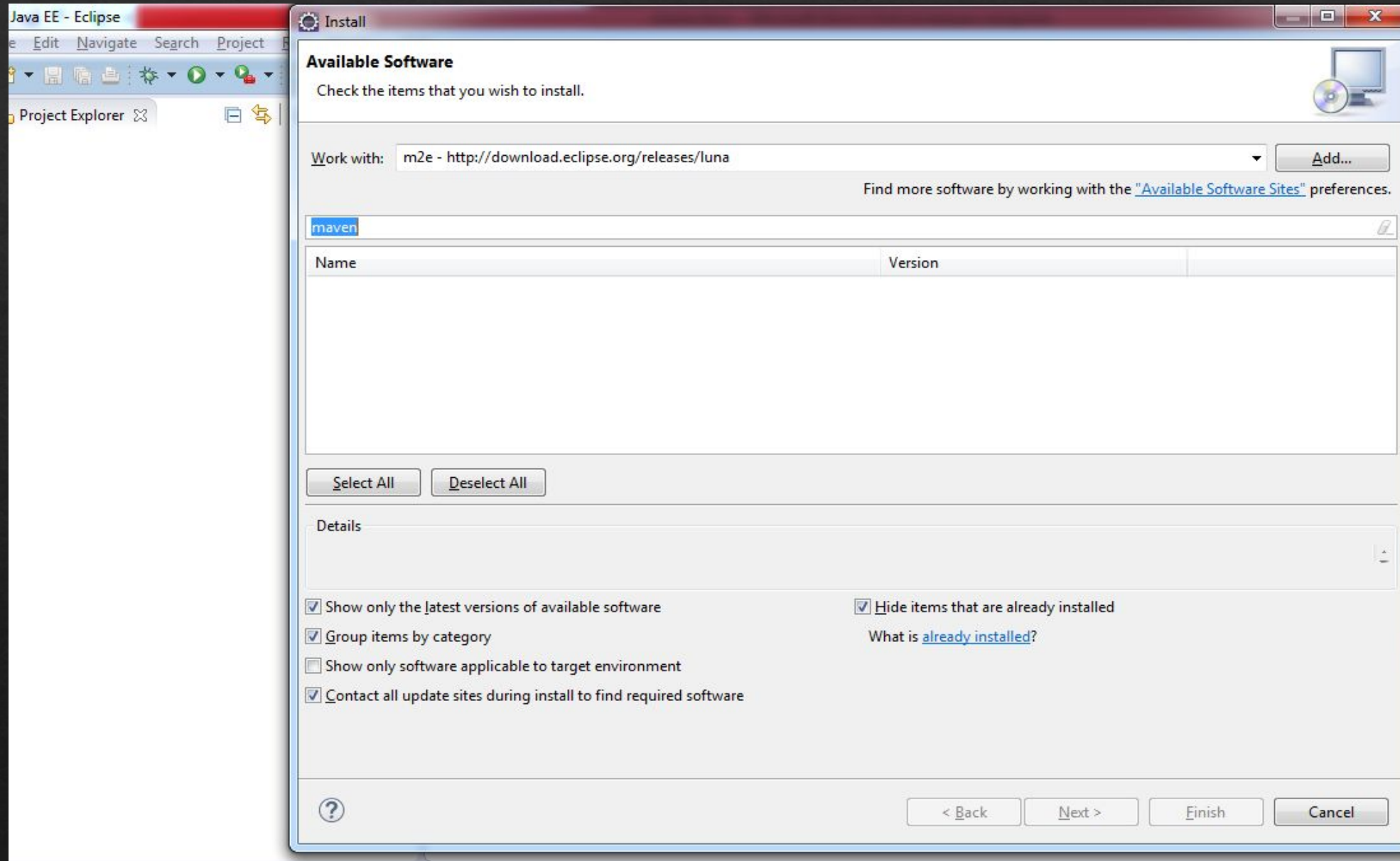
☒ Show only the latest versions of available software ☒ Hide items that are already installed

☒ Group items by category What is [already installed](#)?

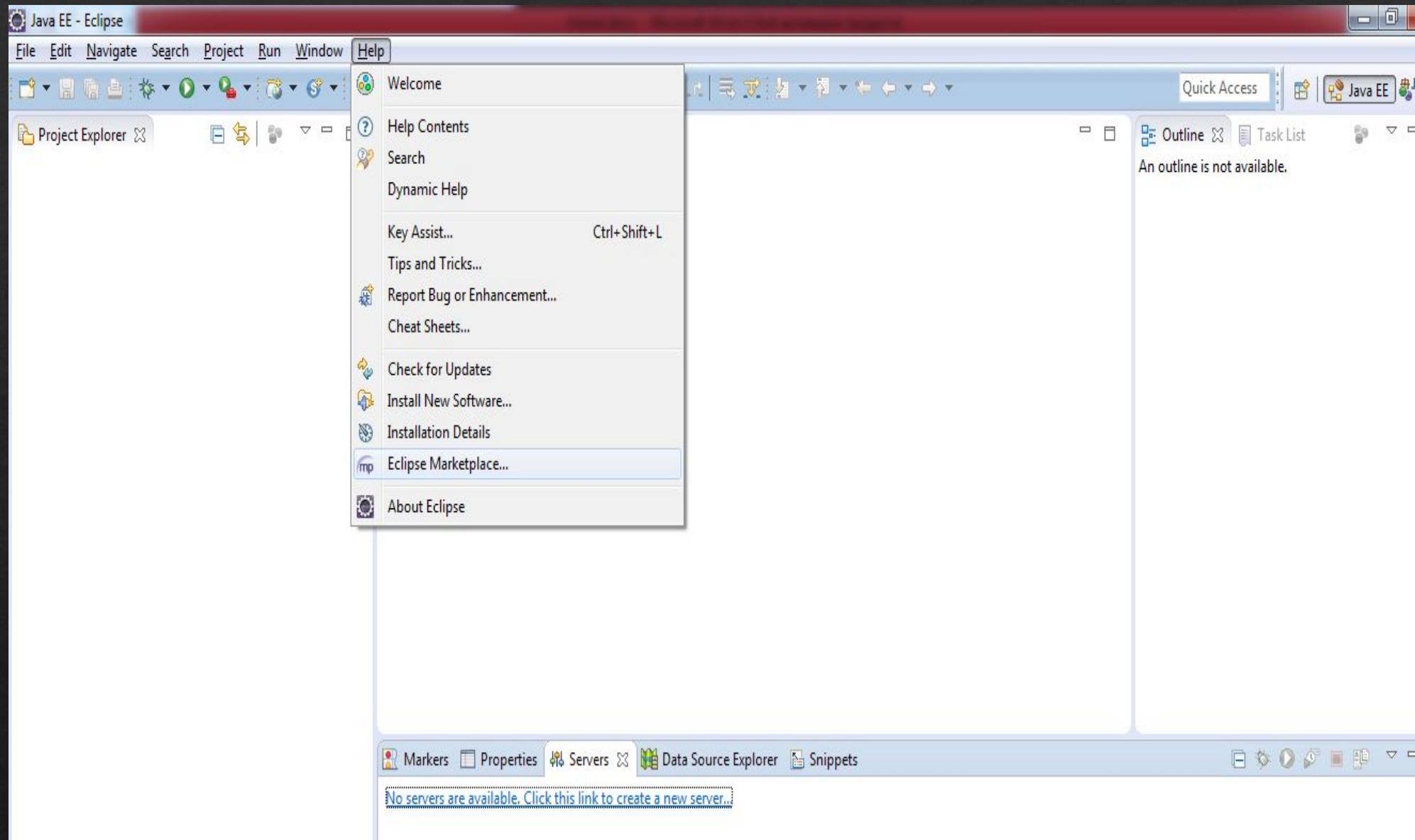
☐ Show only software applicable to target environment

☒ Contact all update sites during install to find required software

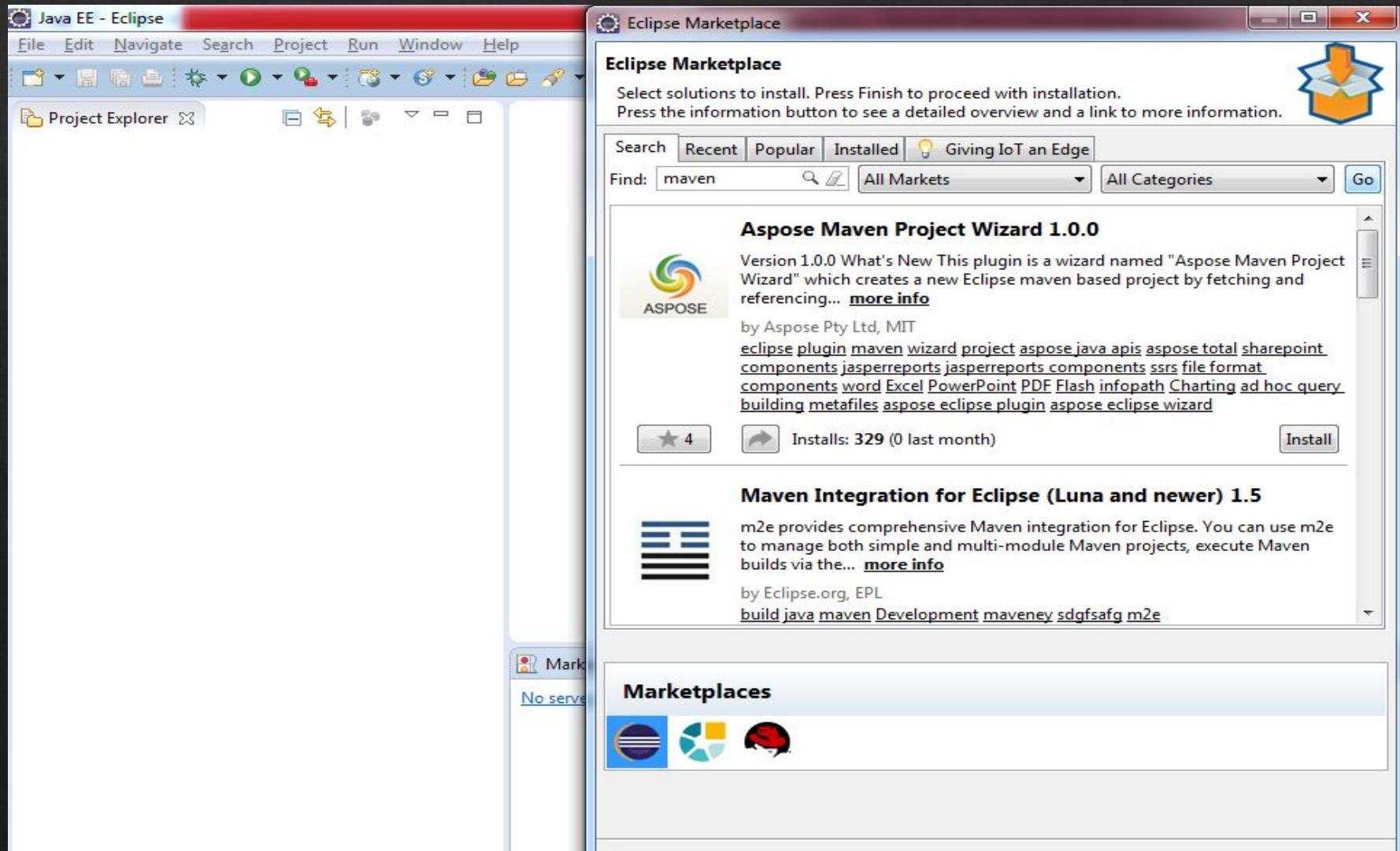




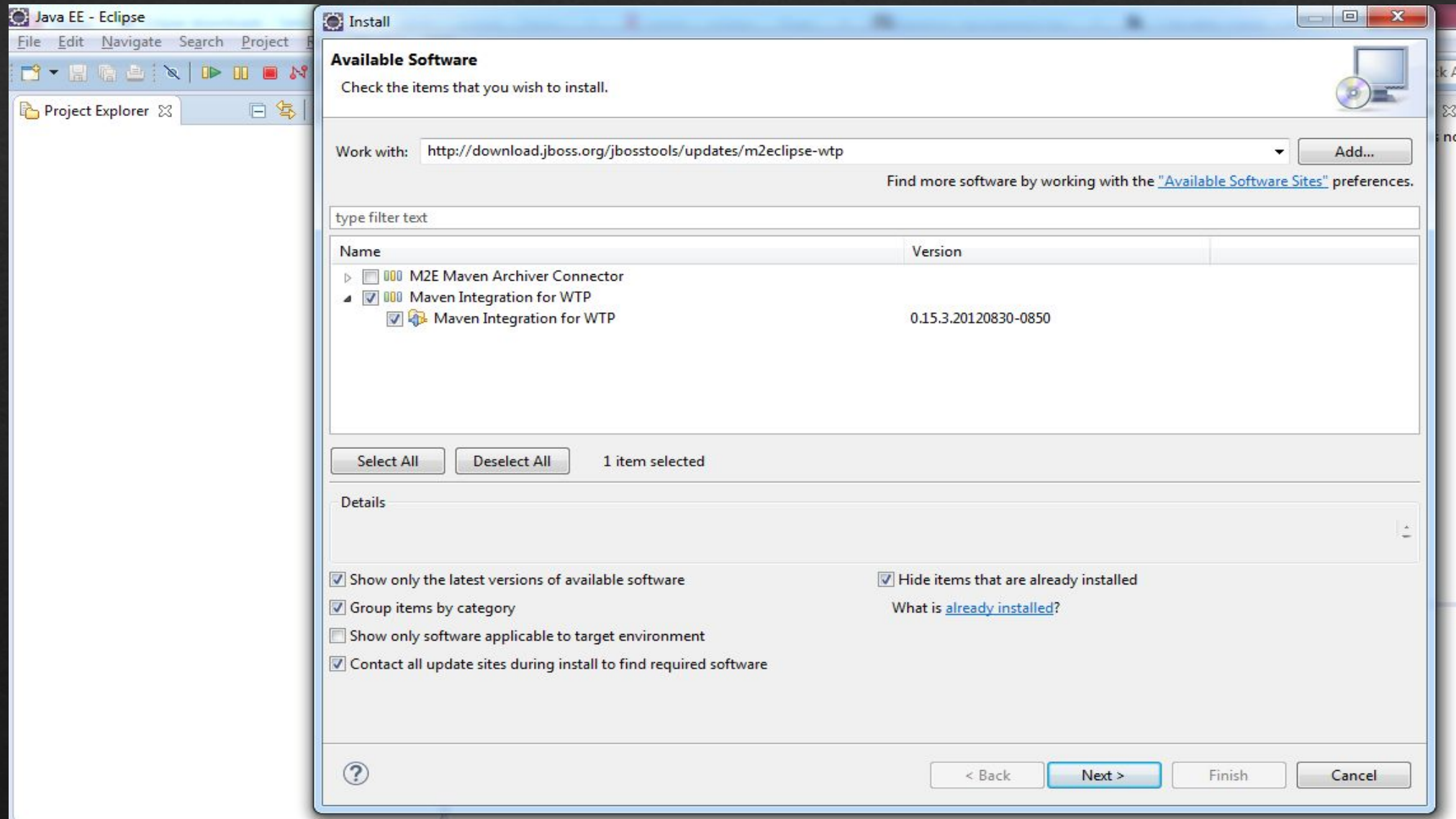




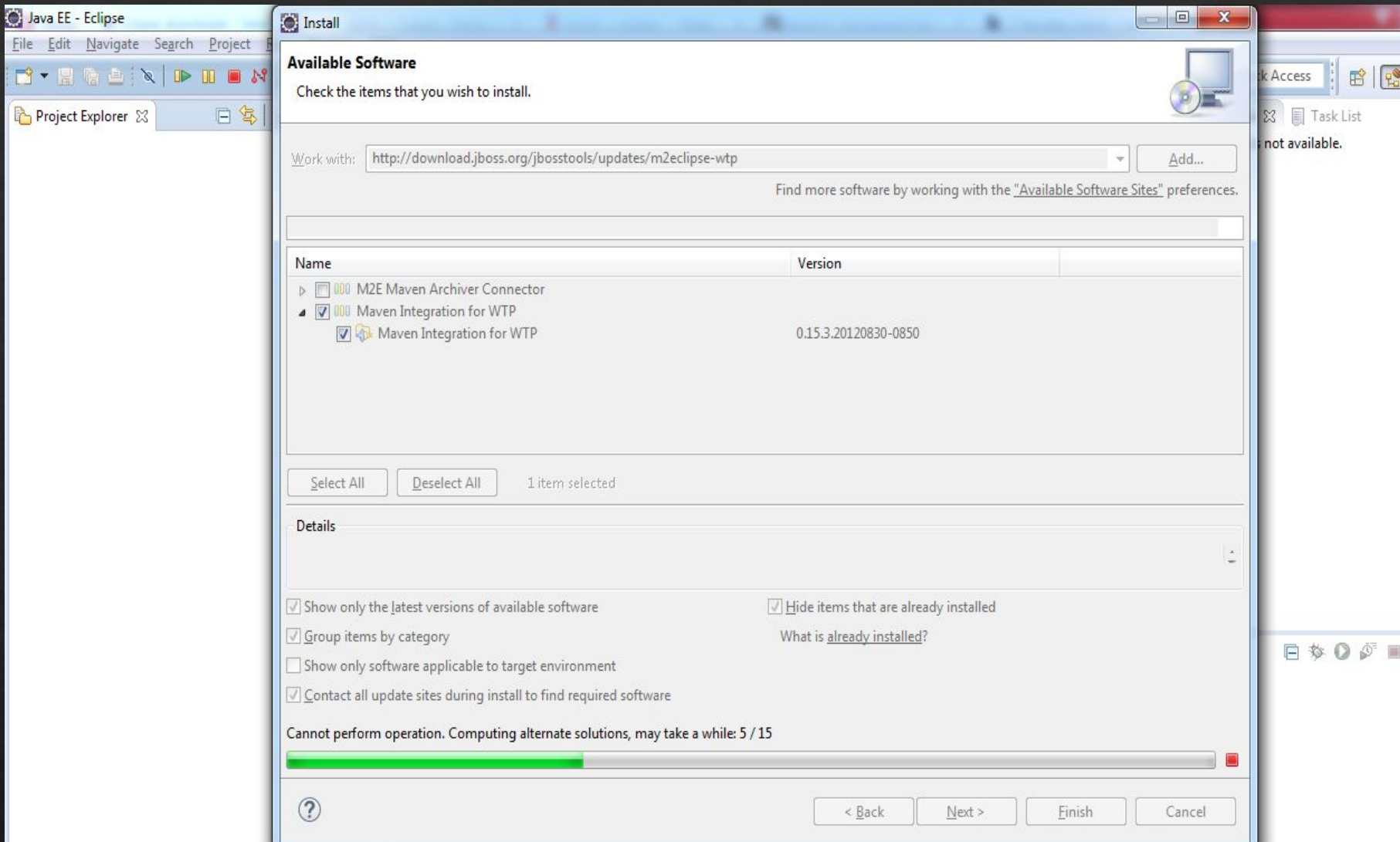
# Maven integration



Если Maven проект не создаётся, то  
ВОЗМОЖНО:

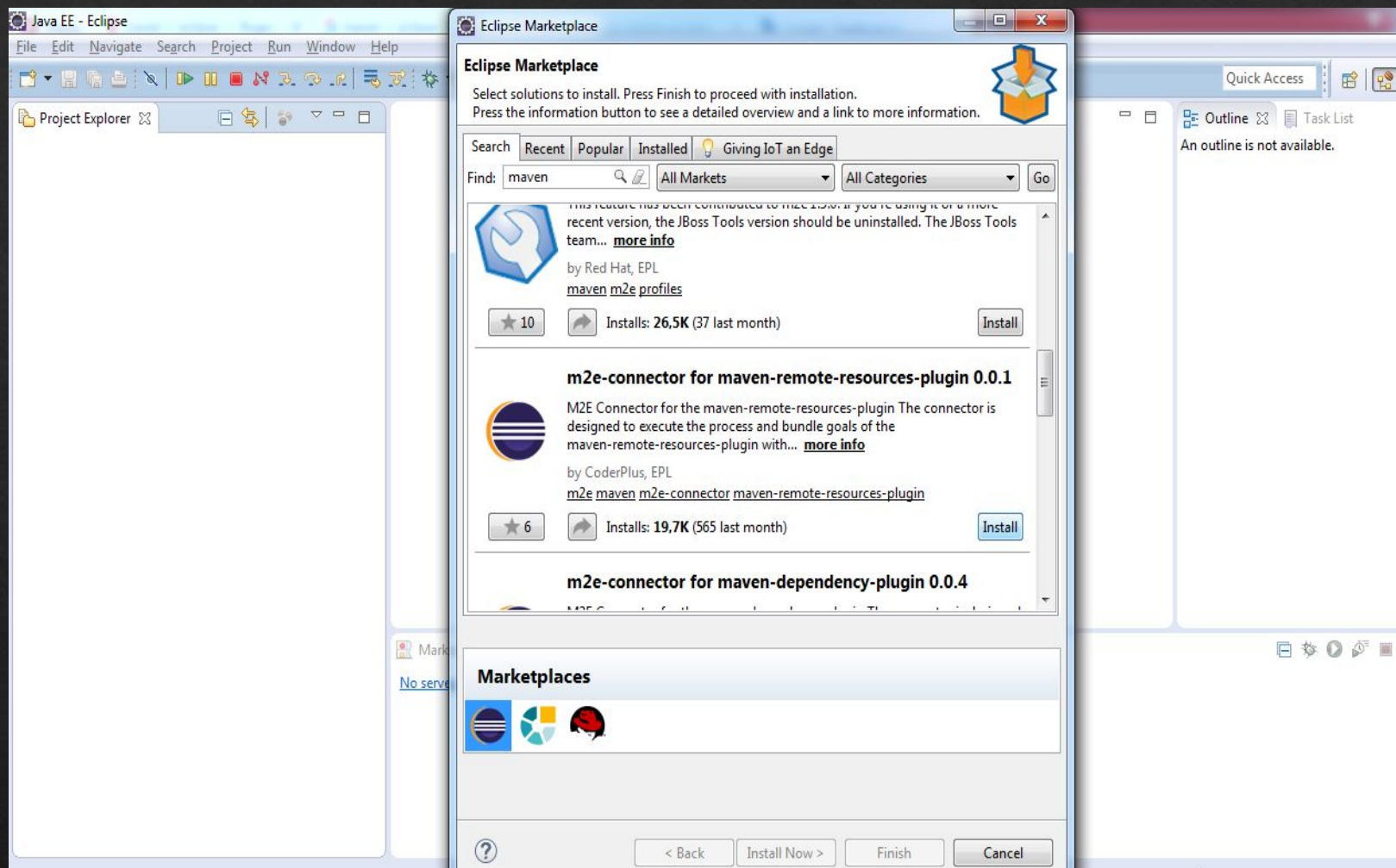






Reboot Eclipse

Если ещё косяк, то help – eclipse Marketplace. Из списка установить нужно m2e-connector и тд



Если **Eclipse** ругается на директорию, сервер и прочее (вся что связано с сетью)

То скорее всего нужно добавить **Location** и **Name**, чтобы скачать пакет

Preferences

type filter text

- > General
- > Ant
- > Cloud Foundry
- > Data Management
- Gradle
- > Help
- ▼ Install/Update
  - Automatic Updates
  - Available Software Site
- > Java
- > Java EE
- > Java Persistence
- > JavaScript
- > JSON
- > Maven
- > Mylyn

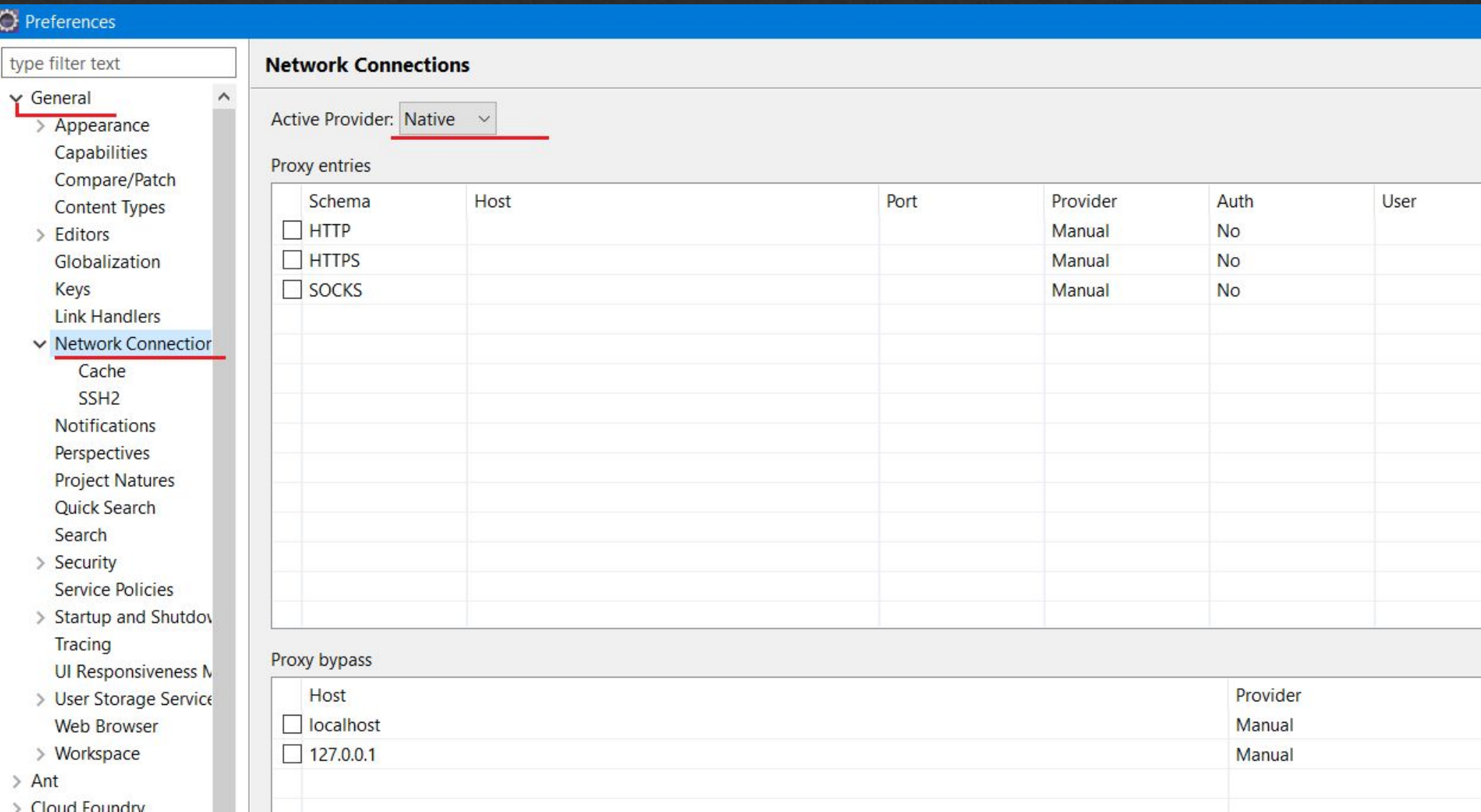
### Available Software Sites

type filter text

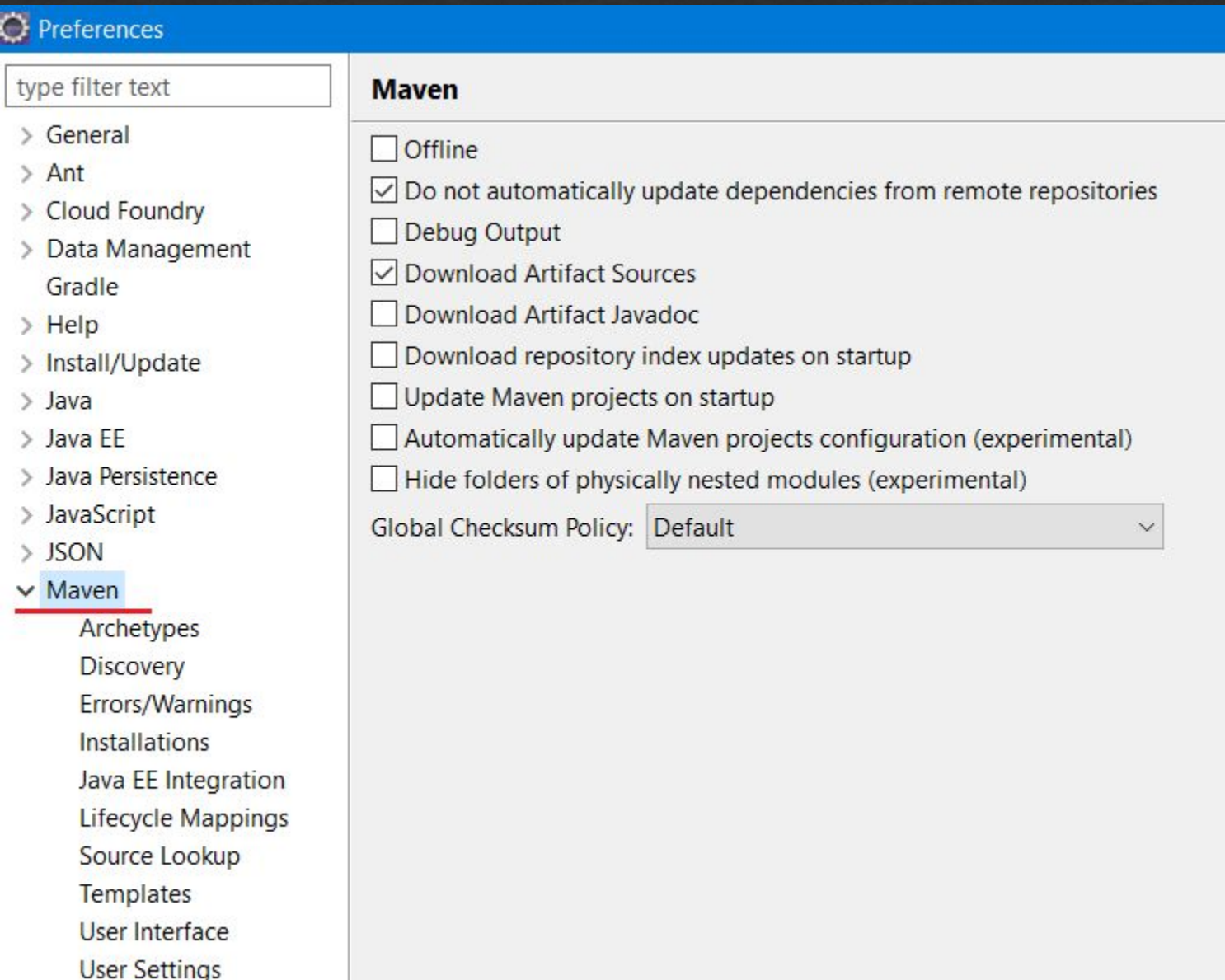
Name	Location	Enabled
<input checked="" type="checkbox"/> 2019-12	<a href="http://download.eclipse.org/releases/2019-12">http://download.eclipse.org/releases/2019-12</a>	Enabled
<input checked="" type="checkbox"/> EPP Marketplace Client	<a href="http://download.eclipse.org/mpc/releases/1.8.1">http://download.eclipse.org/mpc/releases/1.8.1</a>	Enabled
<input checked="" type="checkbox"/> Latest Eclipse Release	<a href="http://download.eclipse.org/releases/latest">http://download.eclipse.org/releases/latest</a>	Enabled
<input checked="" type="checkbox"/> m2e	<a href="http://download.eclipse.org/technology/m2e/releases">http://download.eclipse.org/technology/m2e/releases</a>	Enabled
<input checked="" type="checkbox"/> The Eclipse Project Updates	<a href="http://download.eclipse.org/eclipse/updates/4.14">http://download.eclipse.org/eclipse/updates/4.14</a>	Enabled
<input checked="" type="checkbox"/> The Eclipse Web Tools Platform (WTP) software repository	<a href="http://download.eclipse.org/webtools/repository/latest">http://download.eclipse.org/webtools/repository/latest</a>	Enabled
<input checked="" type="checkbox"/>	<a href="http://download.eclipse.org/releases">http://download.eclipse.org/releases</a>	Enabled
<input checked="" type="checkbox"/>	<a href="http://download.eclipse.org/ussdk/updates/release/latest">http://download.eclipse.org/ussdk/updates/release/latest</a>	Enabled
<input checked="" type="checkbox"/>	<a href="http://download.oracle.com/otn_software/oepe/12.2.1.8/oxygen/r...">http://download.oracle.com/otn_software/oepe/12.2.1.8/oxygen/r...</a>	Enabled
<input type="checkbox"/>	<a href="http://master.dl.sourceforge.net/project/sapnweclipse/updates/">http://master.dl.sourceforge.net/project/sapnweclipse/updates/</a>	Disabl...



Так же проверить **Network Connection**  
Возможно поменять **Active Provider**



Так же проверить **Maven** директорию

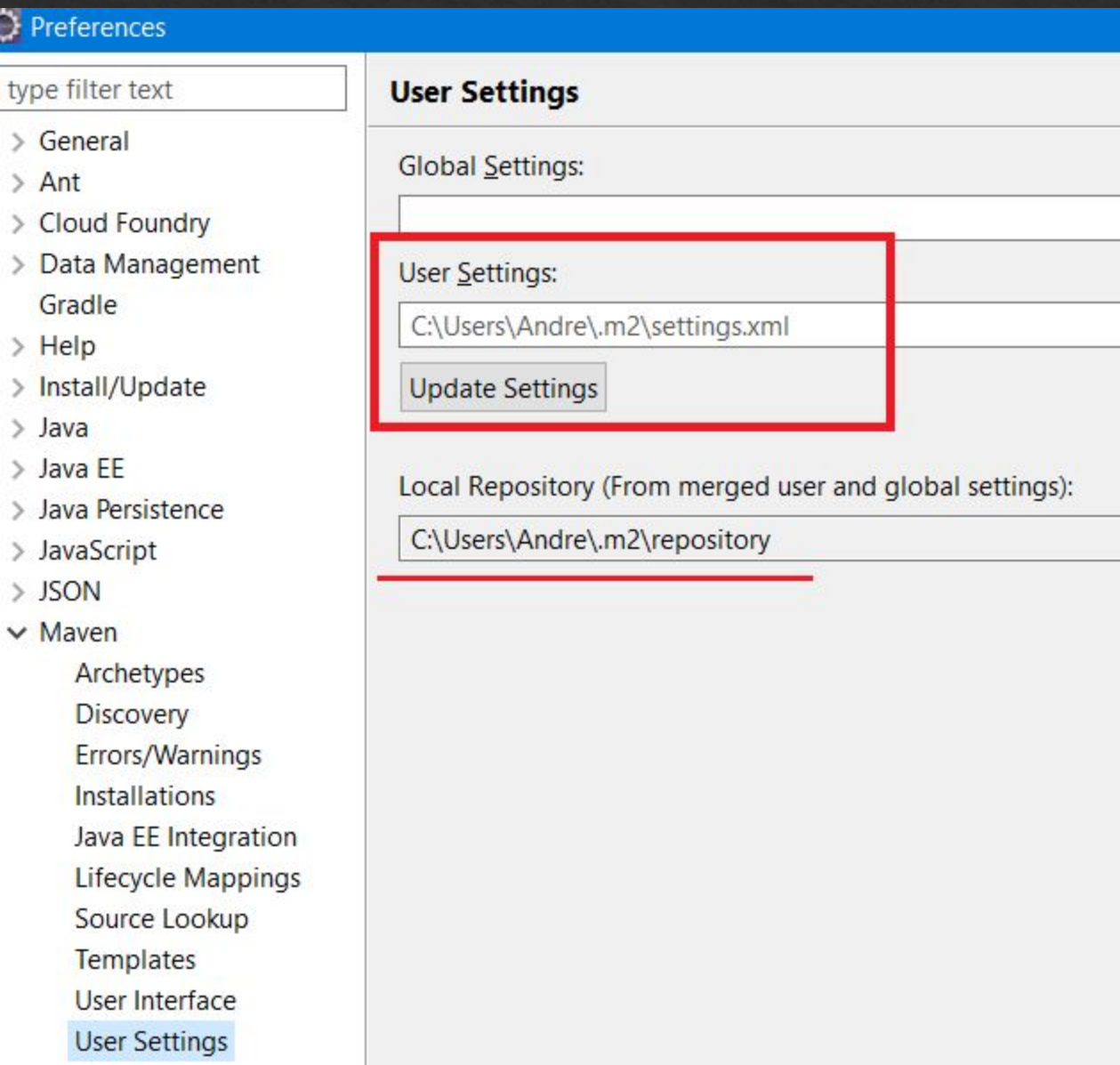


The screenshot shows the Eclipse IDE's 'Preferences' dialog box. The left sidebar contains a tree view of preference categories, with 'Maven' selected and highlighted in blue. The main area on the right is titled 'Maven' and contains several configuration options. A search bar at the top left of the main area contains the text 'type filter text'. The options in the 'Maven' section are:

- ☐ Offline
- ☒ Do not automatically update dependencies from remote repositories
- ☐ Debug Output
- ☒ Download Artifact Sources
- ☐ Download Artifact Javadoc
- ☐ Download repository index updates on startup
- ☐ Update Maven projects on startup
- ☐ Automatically update Maven projects configuration (experimental)
- ☐ Hide folders of physically nested modules (experimental)

At the bottom of the 'Maven' section, there is a label 'Global Checksum Policy:' followed by a dropdown menu currently set to 'Default'.

В **User Settings** пишут настройки прокси (если таковые имеются) в формате **XML**





# Проверить Templates

Preferences

type filter text

> General

> Ant

> Cloud Foundry

> Data Management

Gradle

> Help

> Install/Update

> Java

> Java EE

> Java Persistence

> JavaScript

> JSON

> Maven

Archetypes

Discovery

Errors/Warnings

Installations

Java EE Integration

Lifecycle Mappings

Source Lookup

Templates

Templates

Create, edit or remove templates:

Name	Context	Description	Auto Ins
<input checked="" type="checkbox"/> ear plugin	Plugins	EAR plugin configuration	on
<input checked="" type="checkbox"/> ejb plugin	Plugins	EJB plugin configuration	on
<input checked="" type="checkbox"/> exclusion	Exclusions	New exclusion element	on
<input checked="" type="checkbox"/> execution	Executions	New execution element	on
<input checked="" type="checkbox"/> javac plugin	Plugins	Java compiler plugin configuration	on
<input checked="" type="checkbox"/> m2e profile	Profiles	m2e activated profile	on
<input checked="" type="checkbox"/> m2e-wtp activation	Properties	Enable or disable m2e-wtp configuration	on
<input checked="" type="checkbox"/> m2e-wtp JAX-RS activation	Properties	Enable or disable automatic JAX-RS configuration	on
<input checked="" type="checkbox"/> m2e-wtp JPA activation	Properties	Enable or disable automatic JPA configuration	on
<input checked="" type="checkbox"/> m2e-wtp JSF activation	Properties	Enable or disable automatic JSF configuration	on
<input checked="" type="checkbox"/> m2e-wtp's specific context root property	Properties	Sets the Web Project's context root, overriding pom.xml's <warName> and <finalName>.	on
<input checked="" type="checkbox"/> profile	Profiles	New profile element	on
<input checked="" type="checkbox"/> project	Document	New project element	on
<input checked="" type="checkbox"/> property	Properties	New property element	on
<input checked="" type="checkbox"/> repository	Repositories	New repository element	on
<input checked="" type="checkbox"/> tools.jar	Profiles	Profile for tools.jar	on

