

# JEE – Atividade Hands-on em Laboratório - 02

## Criação de Projeto WEB com Apache Maven

### Prof. Dr. Aparecido Freitas

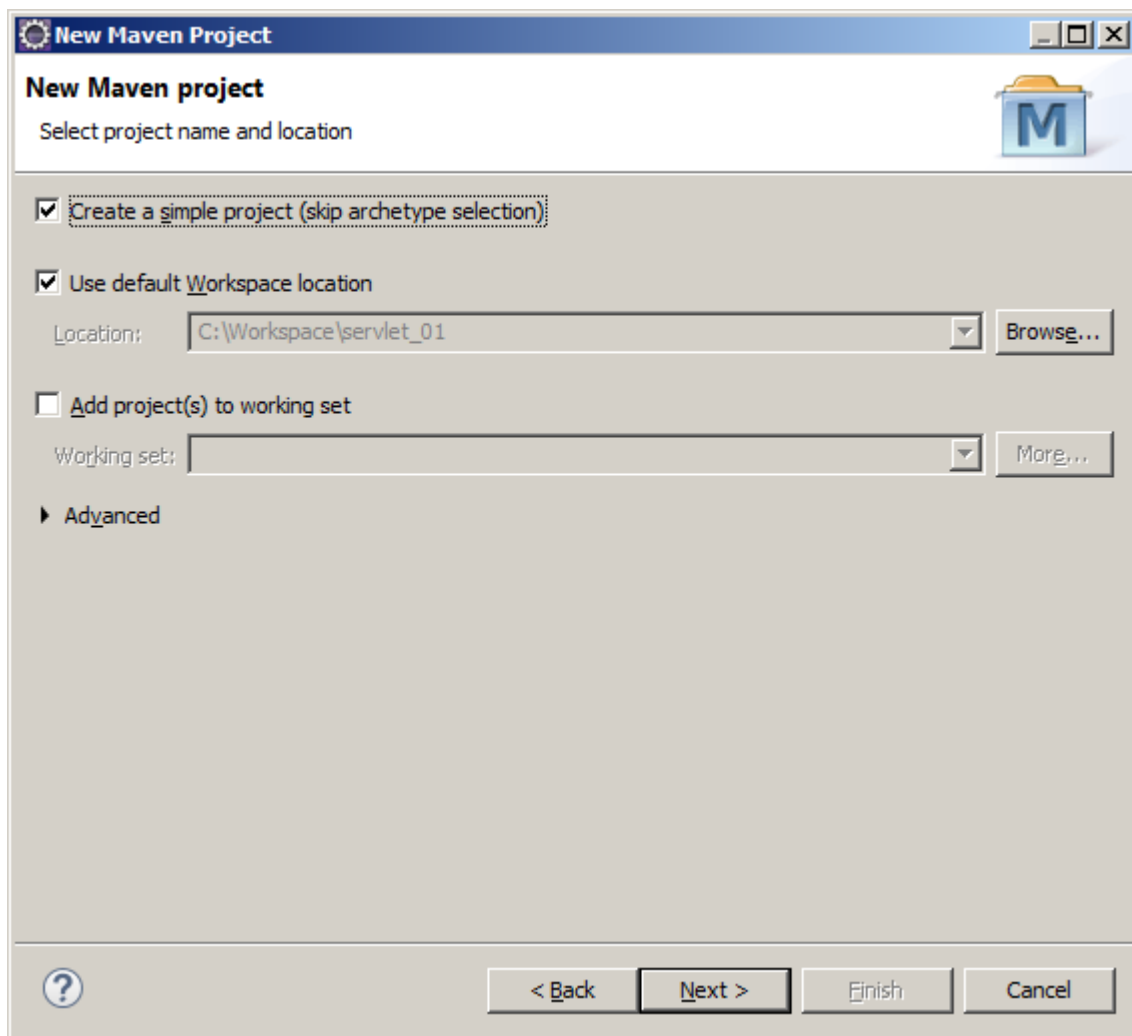
#### 1. Introdução

**Maven** é uma ferramenta da **Apache Software Foundation** para gerenciamento de dependências e automação de build, principalmente em projetos Java.

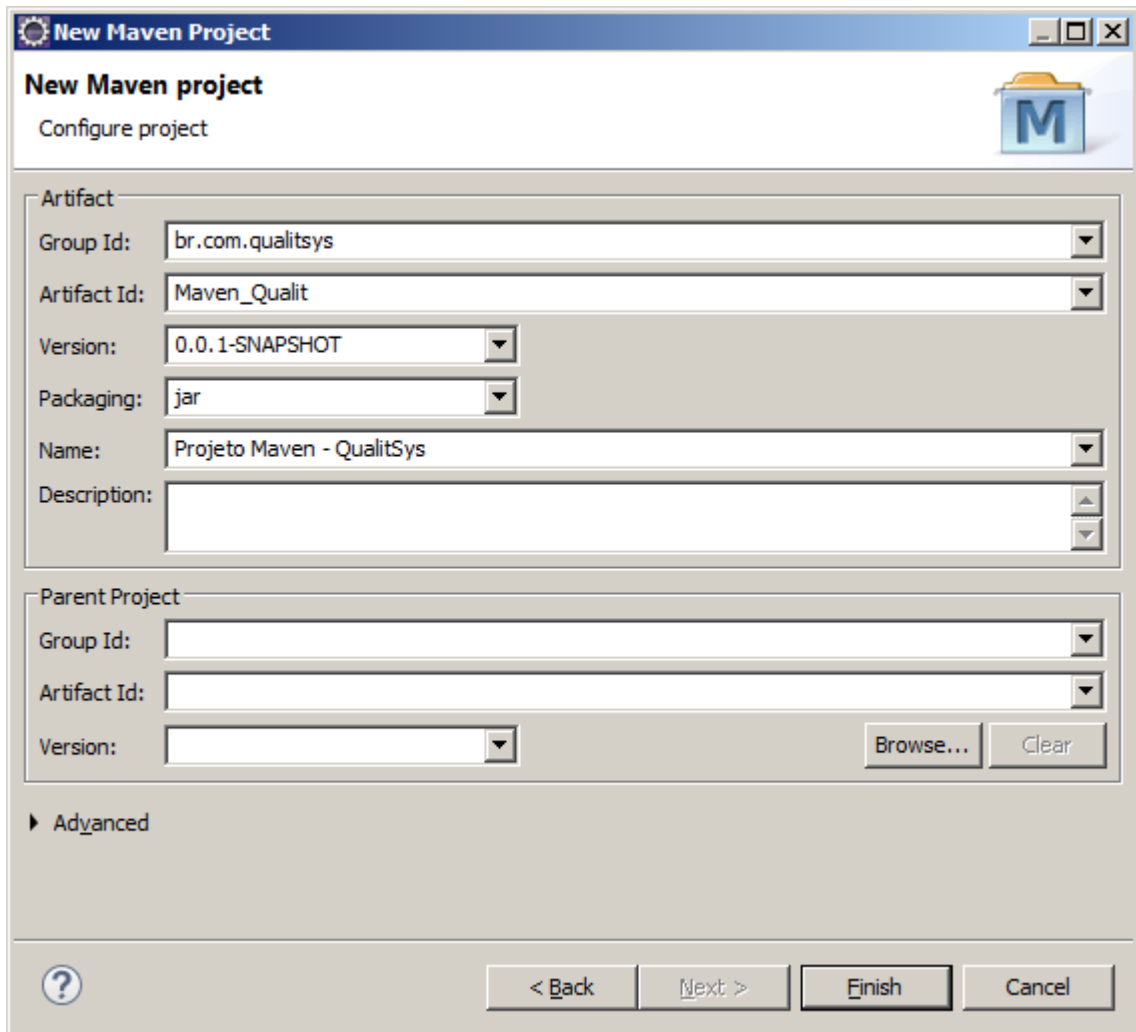
Um projeto que usa Maven possui um arquivo **XML** (**pom.xml**) que descreve o projeto, suas dependências, detalhes do build, diretórios, plugins requeridos, etc. Este arquivo é conhecido como **POM** (Project Object Model).

Usaremos **Maven** neste exercício. As versões mais recentes do Eclipse EE já possui um plugin para criar projetos com **Maven**.

Para criar um novo projeto com **Maven**, acesse o menu **File, New, Maven Project**. Marque a opção **Create a simple project** e clique em **Next >**.



Preencha o campo **Group Id** com um nome único que identifica sua organização (domínio ao contrário) e **Artifact Id** um identificador único do projeto dentro da organização. Selecione **war** na seleção **Packaging**. Depois, clique em **Finish**.



Um projeto web será criado dentro do workspace do Eclipse, com um arquivo **pom.xml** básico:

```
<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/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>br.com.qualitsys</groupId>
  <artifactId>Maven_Qualit</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Projeto Maven - QualitSys</name>

</project>
```

Vamos configurar uma propriedade do projeto para que o processo de build use a codificação **UTF-8** para copiar arquivos e, também, configurar o plugin de compilação para dizer que nosso projeto deve usar Java 8.

```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<build>
    <plugins>
        <plugin>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.0</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Após a inclusão do Código acima no arquivo pom.xml, dê **Ctrl + A** e em seguida **Ctrl + I** para providenciar endentação no arquivo pom.xml.

O arquivo **pom.xml** deverá ficar assim:

```

<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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>br.com.qualitsys</groupId>
  <artifactId>Maven_Qualit</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Projeto Maven - QualitSys</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.0</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>

```

Observação: O Eclipse poderá reclamar da falta do arquivo **web.xml**. Nas últimas versões **JEE** esse arquivo **web.xml** é opcional.

O que acontece é o seguinte. O projeto foi criado, mas está faltando o **web.xml**, conforme nos mostra a mensagem de erro:

**web.xml is missing and <failOnMissingWebXml> is set to true**

Que em outras palavras quer dizer que o **web.xml** não foi encontrado e o **failOnMissingWebXml** está configurado como **true**.

Como resolver o erro **web.xml is missing and <failOnMissingWebXml> is set to true?**

Temos ao menos duas formas de resolver o erro **web.xml is missing and <failOnMissingWebXml> is set to true**:

- 1- Devemos criar o arquivo web.xml ou
- 2 – Devemos setar o parâmetro failOnMissingWebXml como false para que não seja mais procurado em nosso projeto o arquivo web.xml.

**Qual a melhor opção?**

Depende do projeto. Caso você queira trabalhar apenas com **annotations**, basta setar o failOnMissingWebXml como **false** e ser feliz.

Vamos ajustar este erro e seguir em frente?

No seu **pom.xml** adicione a configuração conforme abaixo:

```
<properties>
  <failOnMissingWebXml>false</failOnMissingWebXml>
</properties>
```

O arquivo pom.xml, ficará assim:

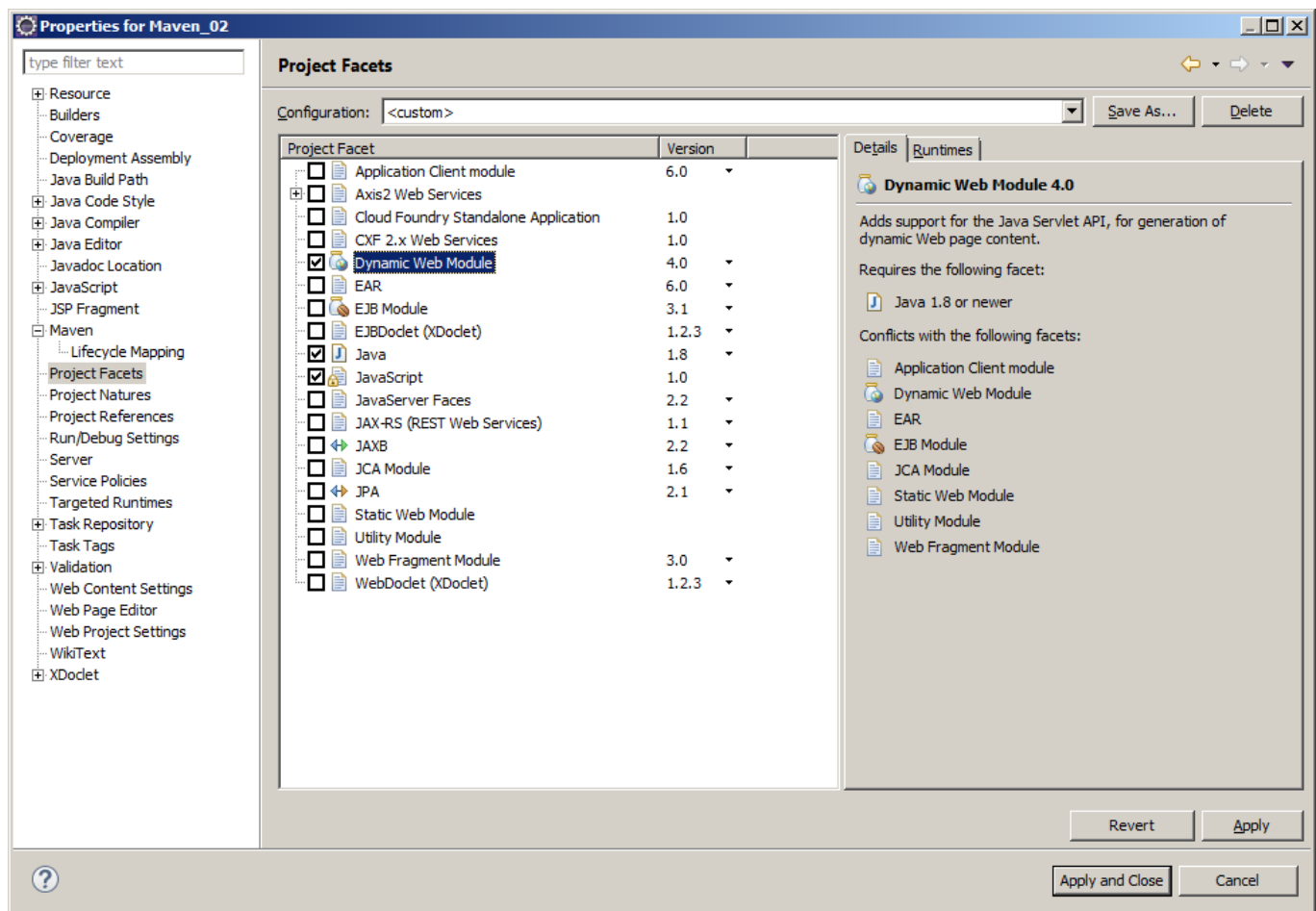
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>br.com.qualitsys</groupId>
  <artifactId>Maven_Qualit</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Projeto Maven - QualitSys</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <failOnMissingWebXml>false</failOnMissingWebXml>
  </properties>

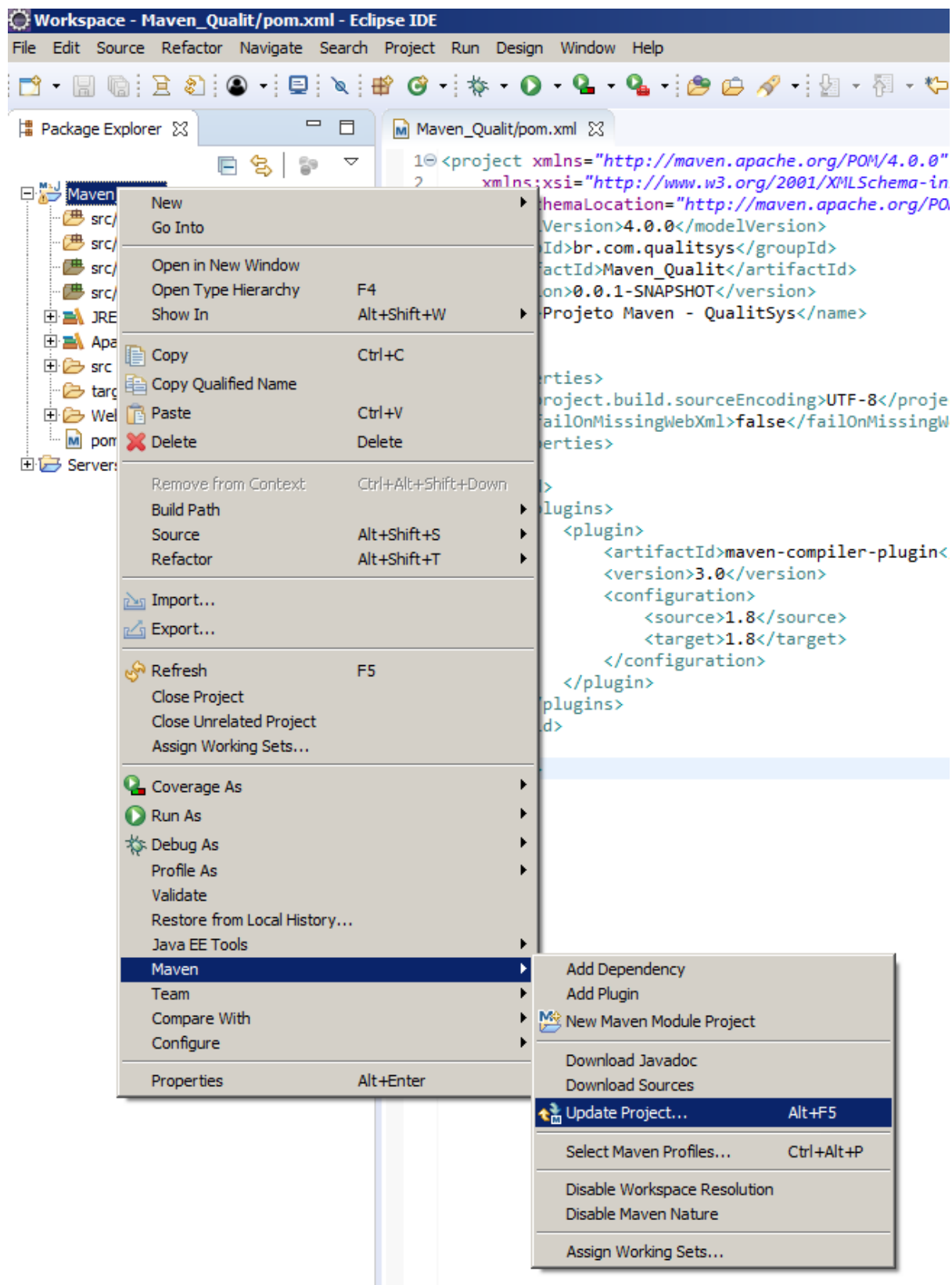
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.0</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

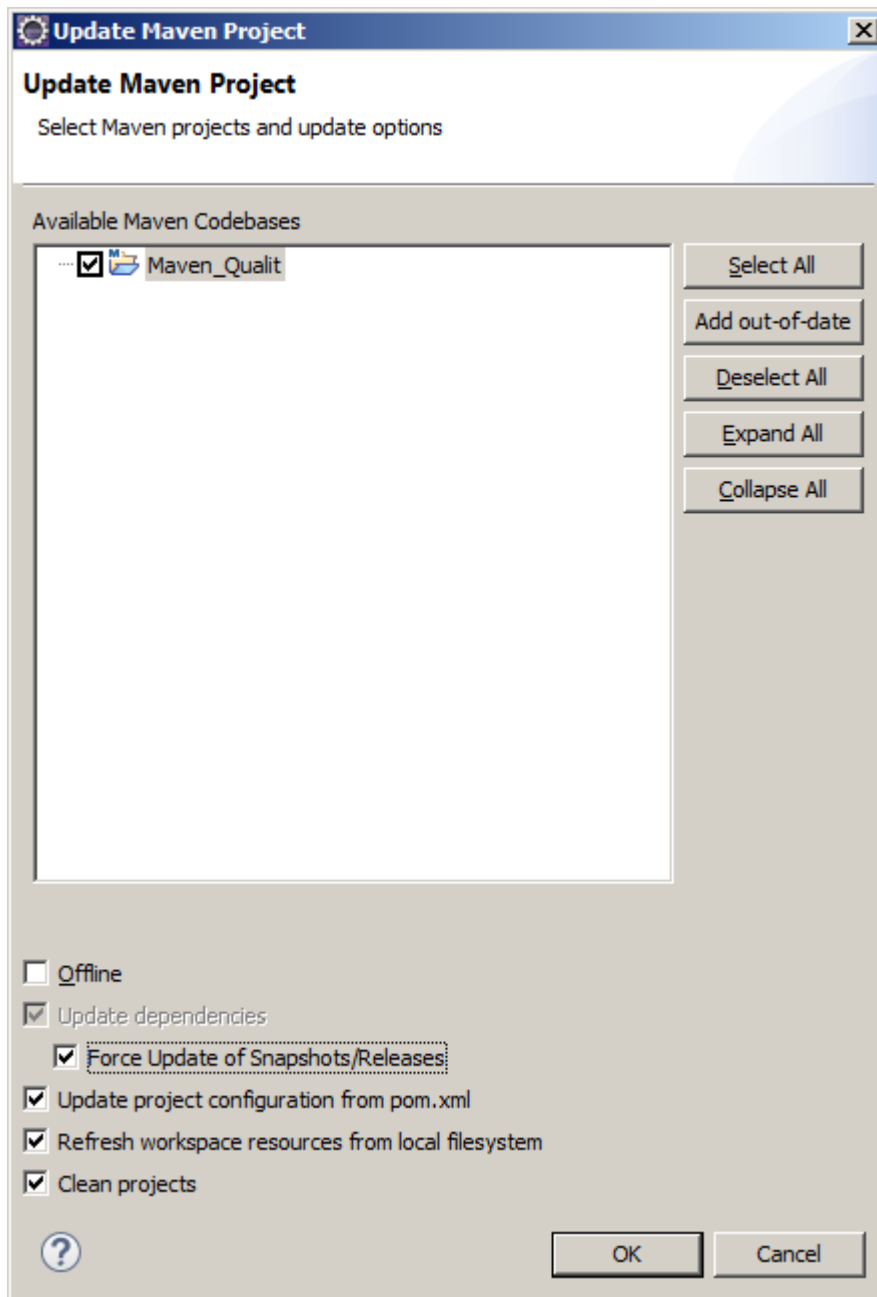
Agora falta adicionar suporte para a **Java Servlet API** para a geração dinâmica de páginas web. (**Dynamic Web Module**).

Vamos selecionar as propriedades do projeto e clicar em **Project Faces** e definir a versão **4.0**. Essa versão requer **Java 1.8** ou mais recente.



Vamos agora atualizar o Projeto Maven. Botão direito no Projeto Maven criado. Selecionar a opção



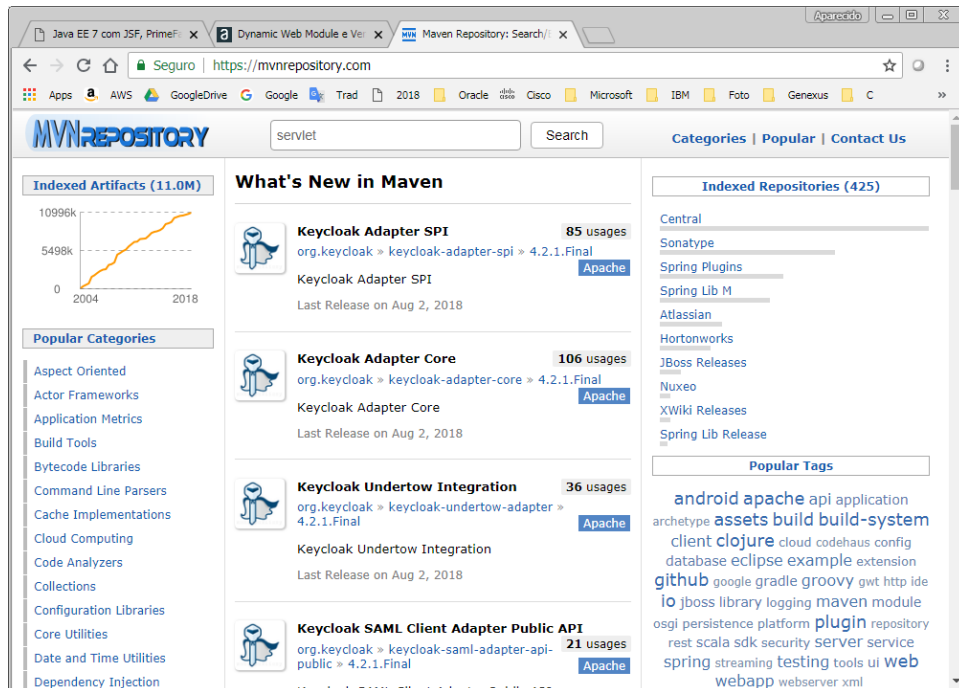


Selecionar a opção Force Update of Snapshots/Release.

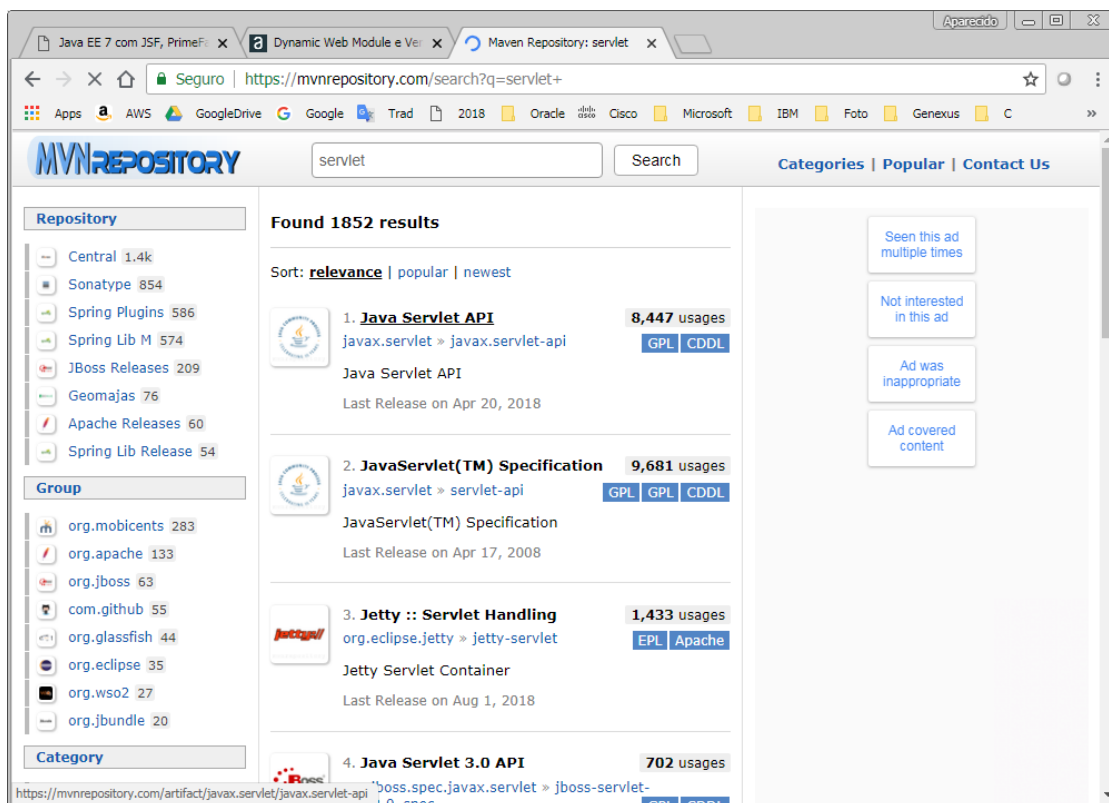


Vamos agora acrescentar ao Projeto Maven, a API de **Servlets**, a qual deve ser declarada como uma dependência do projeto.

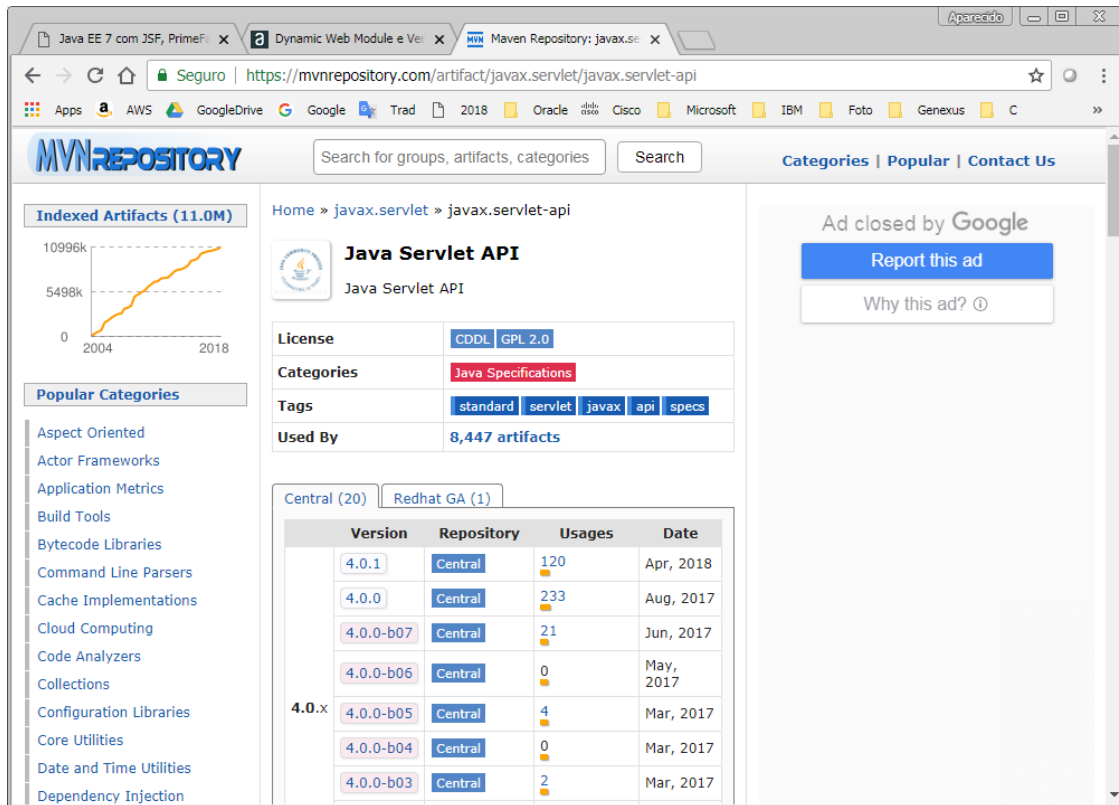
Precisamos adicionar essa dependência no **pom.xml**. Como saber o código desta dependência? Podemos consultar o repositório central do Maven. (<http://mvnrepository.com>)



Na caixa de pesquisa, digite servlet>



## Selecione Java Servlet API.



Home » javax.servlet » javax.servlet-api

### Java Servlet API

Java Servlet API

**License** CDDL | GPL 2.0

**Categories** Java Specifications

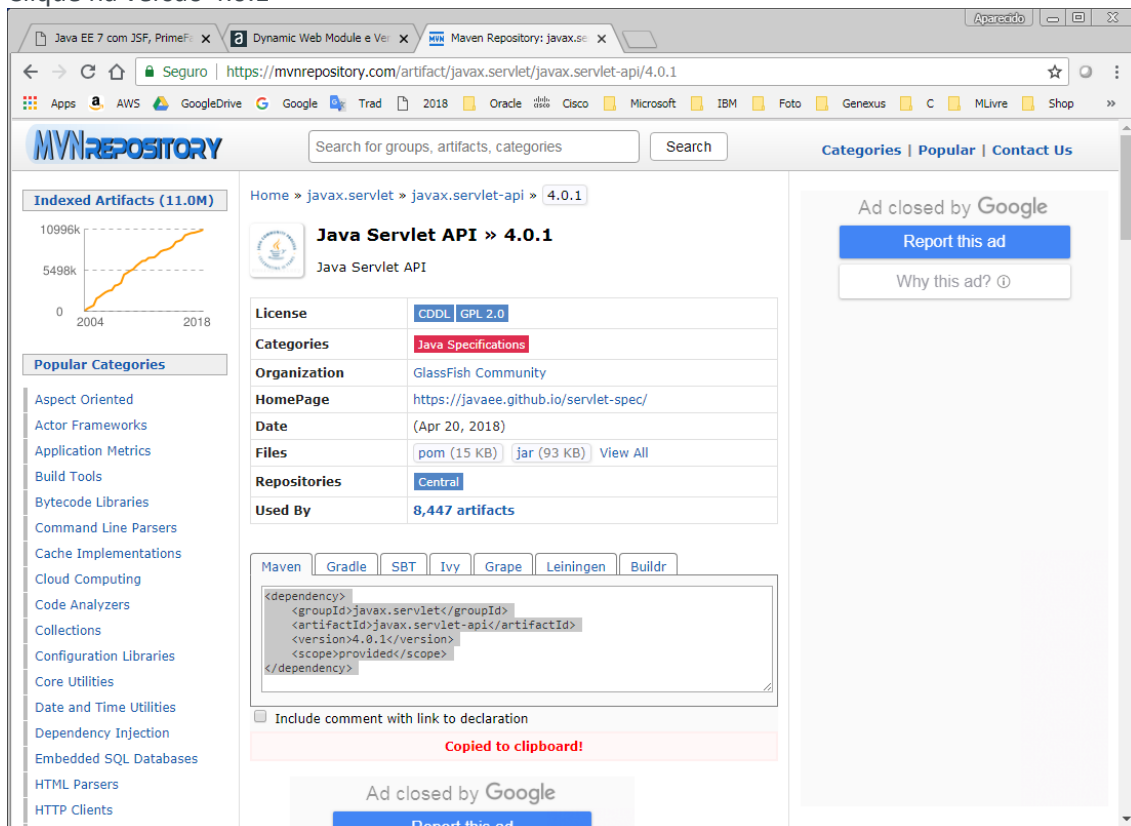
**Tags** standard servlet javax api specs

**Used By** 8,447 artifacts

Central (20) Redhat GA (1)

Version	Repository	Usages	Date
4.0.1	Central	120	Apr, 2018
4.0.0	Central	233	Aug, 2017
4.0.0-b07	Central	21	Jun, 2017
4.0.0-b06	Central	0	May, 2017
4.0.x	Central	4	Mar, 2017
4.0.0-b05	Central	0	Mar, 2017
4.0.0-b04	Central	0	Mar, 2017
4.0.0-b03	Central	2	Mar, 2017

## Clique na versão 4.0.1



Home » javax.servlet » javax.servlet-api » 4.0.1

### Java Servlet API » 4.0.1

Java Servlet API

**License** CDDL | GPL 2.0

**Categories** Java Specifications

**Organization** GlassFish Community

**HomePage** https://javaee.github.io/servlet-spec/

**Date** (Apr 20, 2018)

**Files** pom (15 KB) jar (93 KB) View All

**Repositories** Central

**Used By** 8,447 artifacts

Maven Gradle SBT Ivy Grape Leiningen Buildr

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
  <scope>provided</scope>
</dependency>
```

☐ Include comment with link to declaration

Copied to clipboard!

Na página copie o código **XML** para definir a dependência no arquivo **pom.xml** do projeto **Maven**.

```
<dependencies>
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
  <scope>provided</scope>
</dependency>
</dependencies>
```

Após a inclusão do código acima no arquivo **pom.xml**, tecla **Ctrl+A** e em seguida **Ctrl+I** para indentar o código **XML**.

O arquivo **pom.xml** deverá ficar assim:

```
<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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>br.com.qualitsys</groupId>
  <artifactId>Maven_Qualit</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Projeto Maven - QualitSys</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <failOnMissingWebXml>>false</failOnMissingWebXml>
  </properties>

  <dependencies>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.0</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

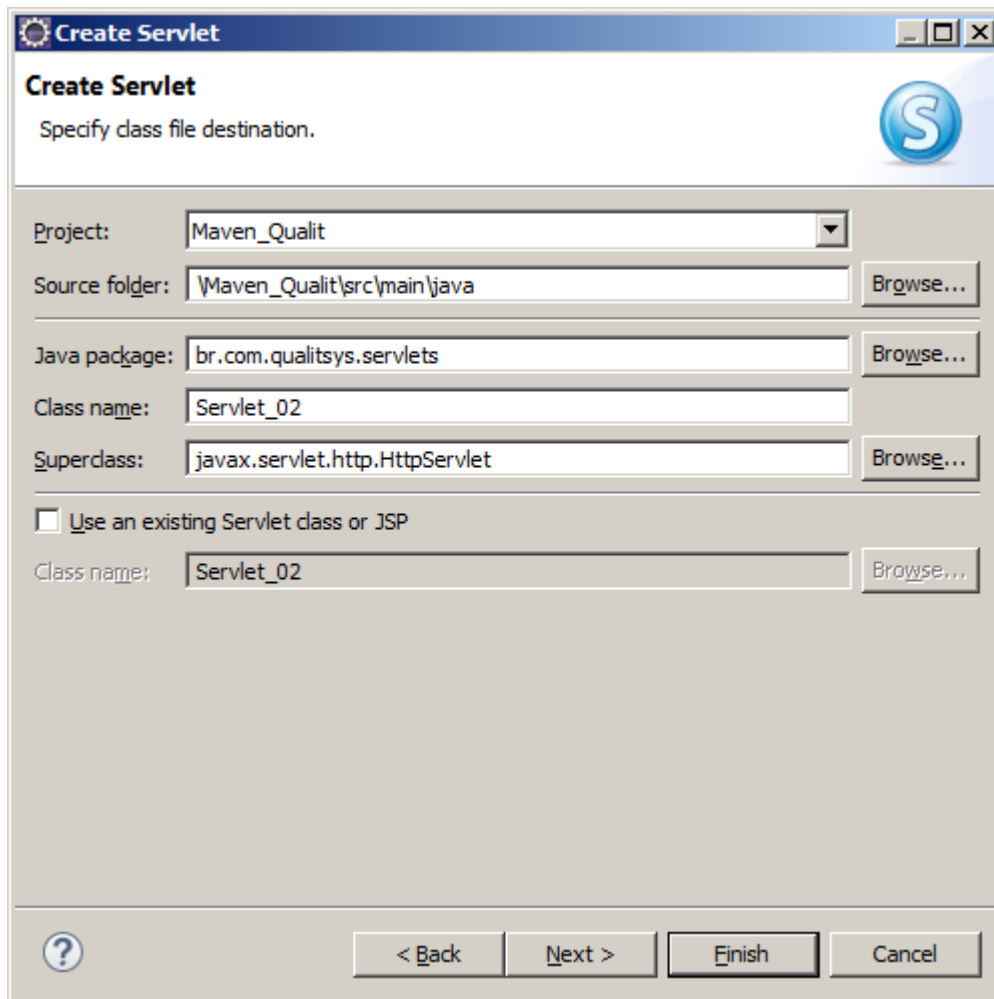
Não esqueça de salvar o arquivo **pom.xml**.

Vamos agora criar uma servlet muito simples, que apenas exibe "**Hello World...**" para o usuário.

Clique com o botão direito no projeto criado, acesse a opção **New** e clique em **Servlet**. Na tela que se abrirá, informe o nome do **pacote** e da classe da **servlet**. Depois, clique em **Finish**.

Package: **br.com.qualitsys.servlets**

Classe: **Servlet\_02**



Tecla **Next>**

**Create Servlet**

Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

Name	Value	Description

URL mappings:

/Servlet_02
-------------

☐ Asynchronous Support

Tecla **Finish**, para a criação do **Servlet**.

```

package br.com.qualitsys.servlets;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class Servlet_02
 */
@WebServlet("/Servlet_02")
public class Servlet_02 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Servlet_02() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at:
    ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
    response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}

```

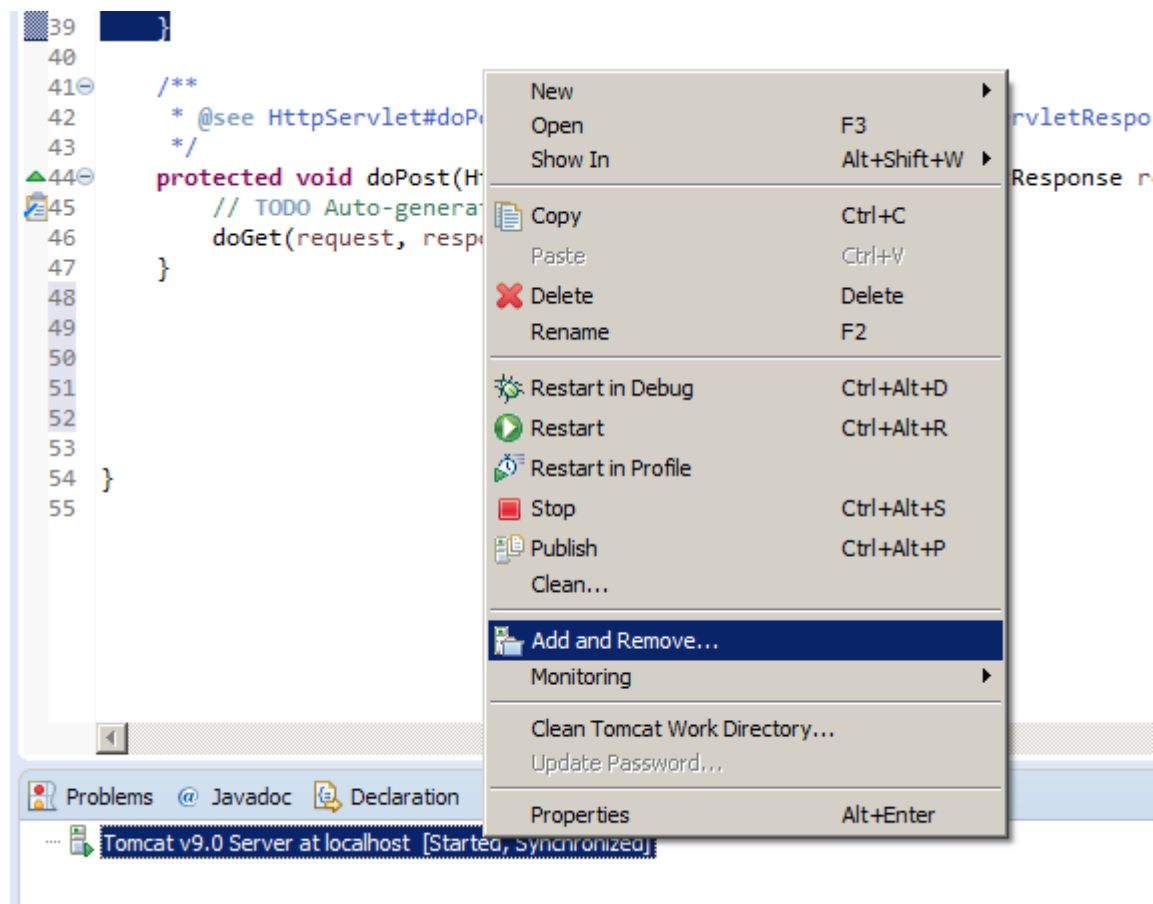
Vamos agora escrever o código no método **doGet** que irá providenciar a resposta **HTML** para o cliente.

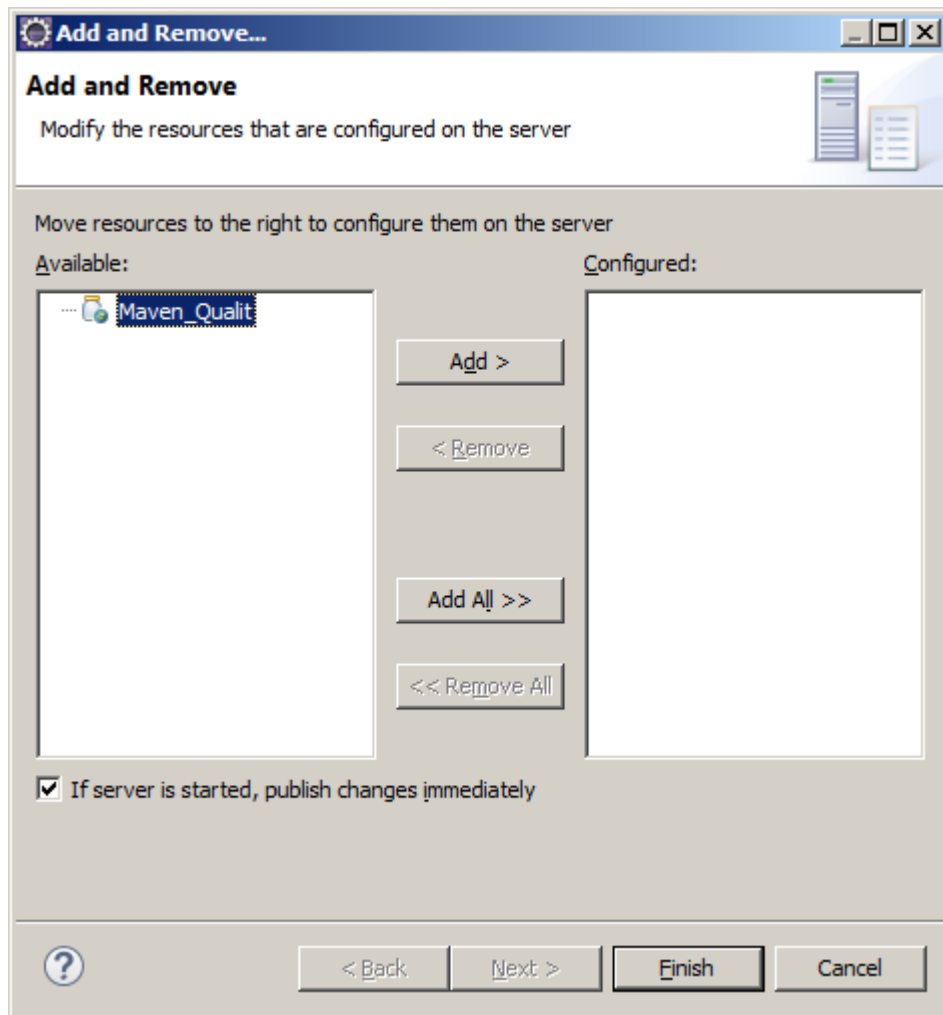
```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    // TODO Auto-generated method stub
```

```
    PrintWriter out = response.getWriter();
    out.print("<html>");
    out.print("<body><h1>Servlet 02 - Hello World...</h1></body>");
    out.print("</html>");
```

```
}
```

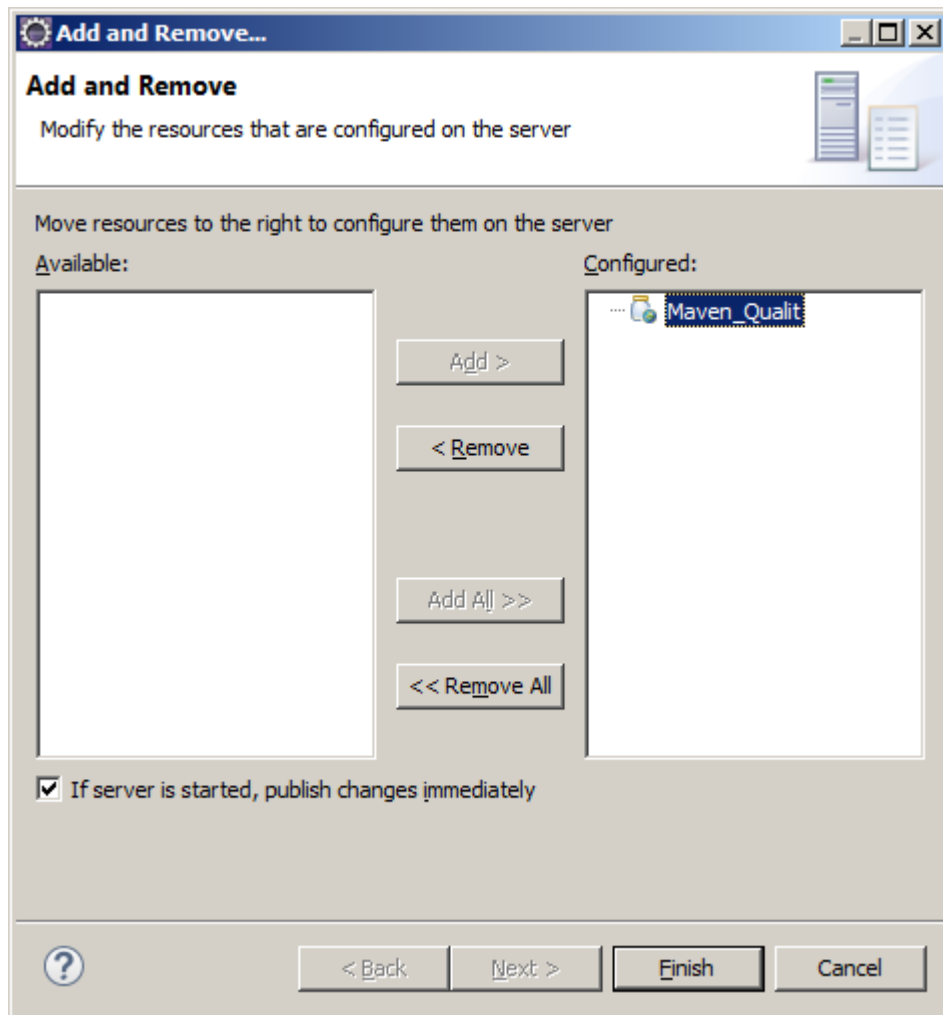
Agora, precisamos adicionar o projeto ao **Tomcat**, para que ele faça a implantação sempre que houver alguma modificação. Na view Servers, clique com o botão direito no servidor do Tomcat e acesse a opção **Add and Remove...**





Marque o projeto na listagem da esquerda e transfira para a listagem da direita, clicando no botão **Add >**. Depois, clique em **Finish**.

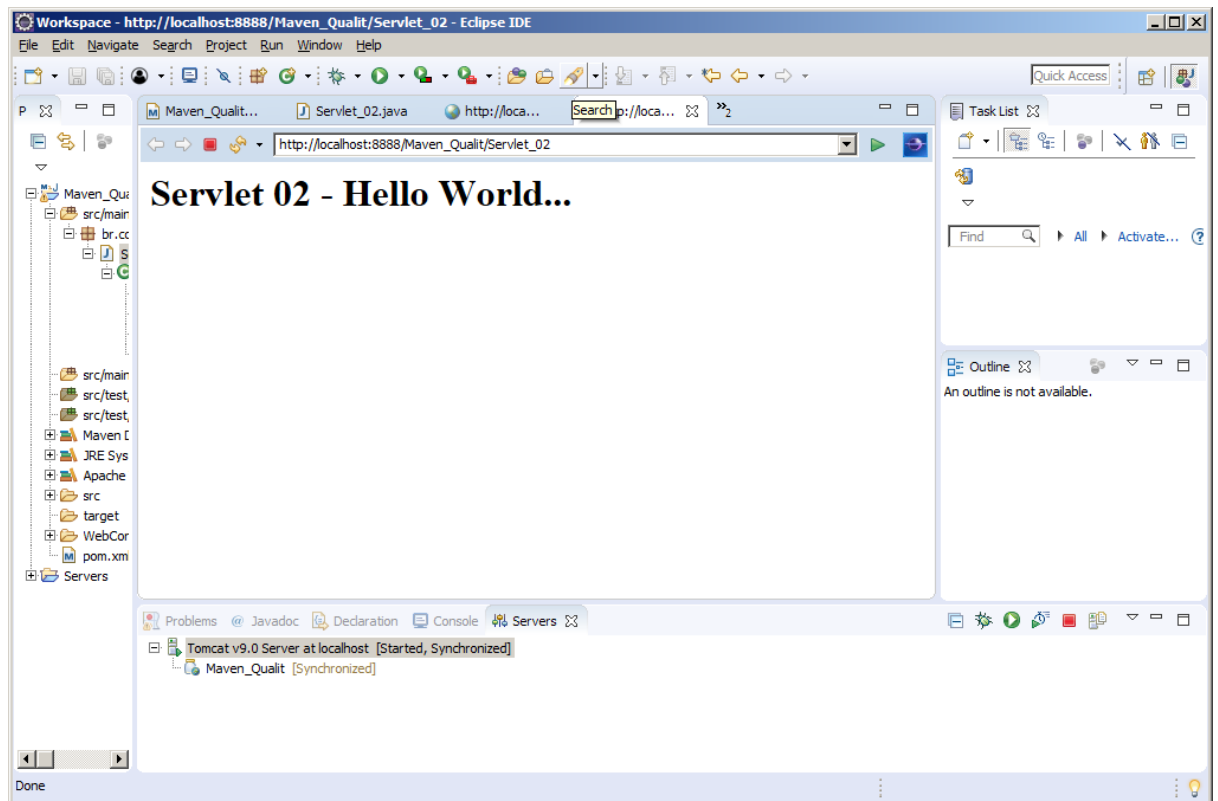




Inicie o **Tomcat**, se ele não estiver sendo executado, depois, acesse:

**[http://localhost:8888/Maven\\_Qualit/Servlet\\_02](http://localhost:8888/Maven_Qualit/Servlet_02)**.

A servlet responderá a requisição e apresentará a mensagem "**Servlet 02 - Hello World...**" no navegador.



Ou diretamente no **Browser**:

