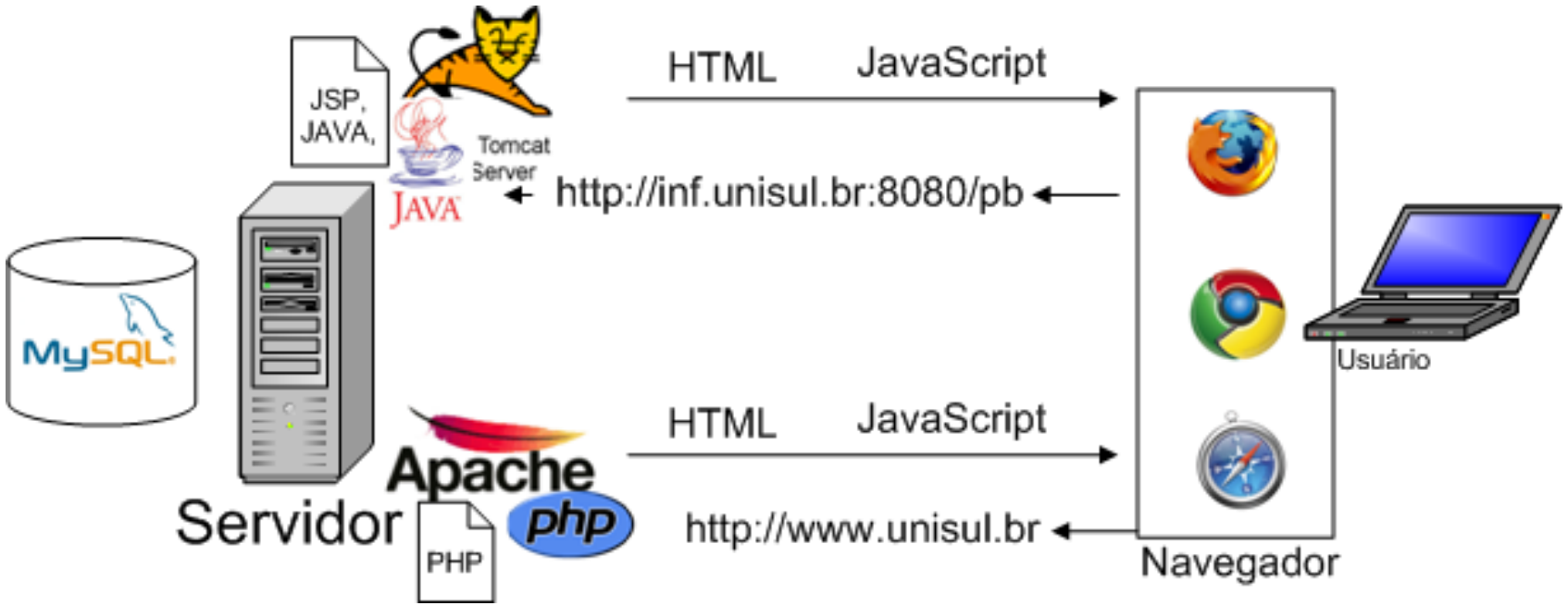
Programação Web Avançada

Revisão geral

http://dl.dropbox.com/u/3025380/PWA/aula1.pdf

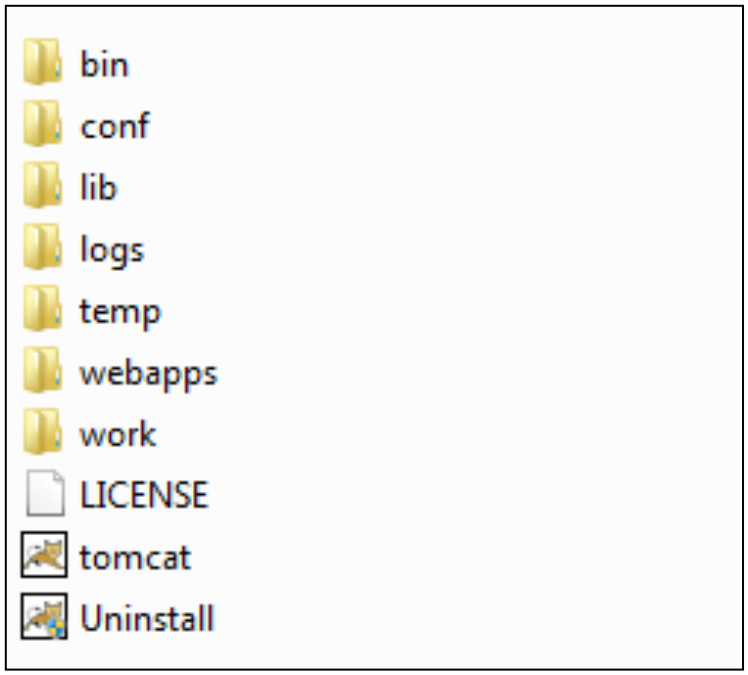
flavio.ceci@unisul.br

Arquitetura para soluções WEB

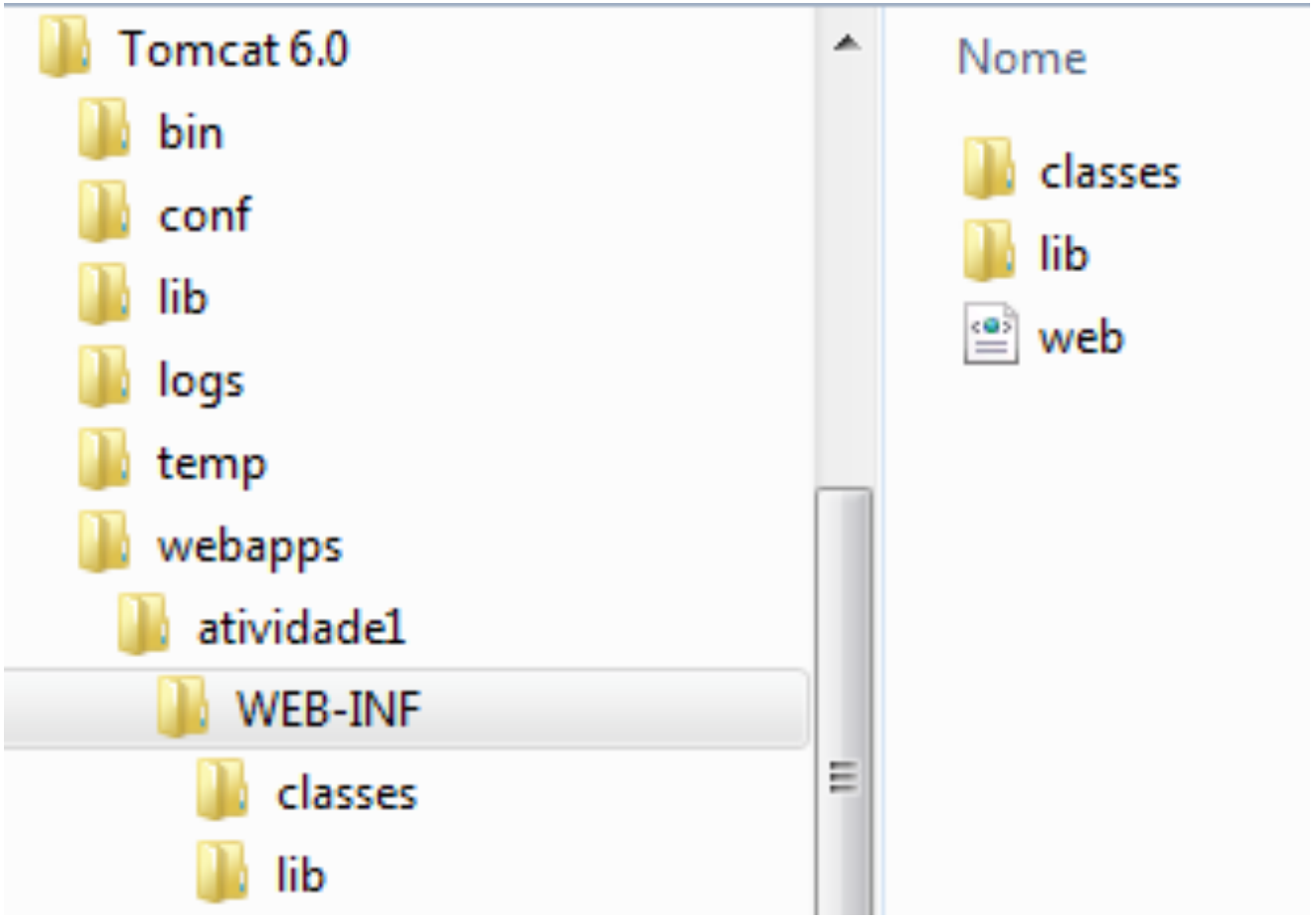


Tomcat

- **Catalina**: é um container web
- **Jasper**: faz a compilação do JSP
- **Coyote** (http) e **jk** (ajp) fornece os serviços de rede em si;



Estrutura de projetos no Tomcat



Servlet

- É um programa Java que executa computações e geram dados que são retornados ao navegador.
- Ao escrever um servlet, você deve estender a classe HttpServlet e implementar um ou ambos os métodos:
  - *void doGet(HttpServletRequest request, HttpServletResponse response)*
  - *void doPost(HttpServletRequest request, HttpServletResponse response)*

Exemplo

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class MeuPrimeiroServlet extends HttpServlet {

        public void doGet(HttpServletRequest request, HttpServletResponse response)
                                        throws IOException {
        PrintWriter saida = null;
        try {
                saida = response.getWriter();
                java.util.Date hoje = new java.util.Date();
                saida.println("<html>");
                saida.println("<body>");
                saida.println("Hoje é: " + hoje);
                saida.println("</body>");
                saida.println("</html>");
        } finally {
                if(saida != null) {
                        saida.close();
                }
        }
}
```

Configurar web.xml

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web
    Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
<servlet>
    <servlet-name>Recupera Parametros</servlet-name>
    <servlet-class>MeuPrimeiroServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Recupera Parametros</servlet-name>
    <url-pattern>/meuprimeiroservlet</url-pattern>
</servlet-mapping>
</web-app>
```

Servlet recebendo parâmetro

(URL: http://localhost:8080/atividade1/recebenome?nome=flavio)

```
public class RecebeNome extends HttpServlet{

private PrintWriter saida = null;

public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
          String nome = request.getParameter("nome");
          try {
                    PrintWriter saida = response.getWriter();
                    saida.println("<html>");
                    saida.println("<body>");
                    if(nome != null && nome.trim().length() > 0) {
                        saida.println("<b>Nome: </b>" + nome);
                    } else {
                        saida.println("<b>Nenhum nome foi passado!</b>");
                    }
                    saida.println("</body>");
                    saida.println("</html>");
          } finally {
                if(saida != null) {
                    saida.close();
                }
          }
}
```

Deficiências dos servlets

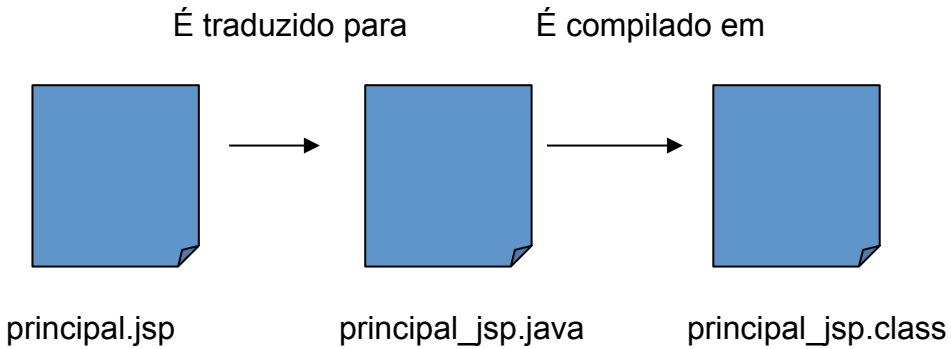
- Servlets não encorajam uma separação da camada de apresentação para a camada lógica;
- Todo o HTML retornado pelo Servlet está fixo ao código Java na forma de uma String;
- Manutenção difícil;

Java Server Pages (JSP)

- É uma tecnologia de páginas de servidor baseada em Servlet que permite que seus usuários embutam código Java dentro de uma página;
- Permite a divisão da camada de apresentação da camada lógica de negócio;
- Possui diversos objetos que fornecem acesso aos serviços applets de servlets.

Funcionamento JSP

- Java + HTML no mesmo código (arquivo .jsp);
- Um .jsp torna-se um servlet que você não cria.
- Container olha o .jsp, identifica o código Java e criar o servlet automaticamente.



Funcionamento JSP

- A tradução e a compilação acontecem somente na PRIMEIRA VEZ que o .jsp é invocado.
- Exemplo:

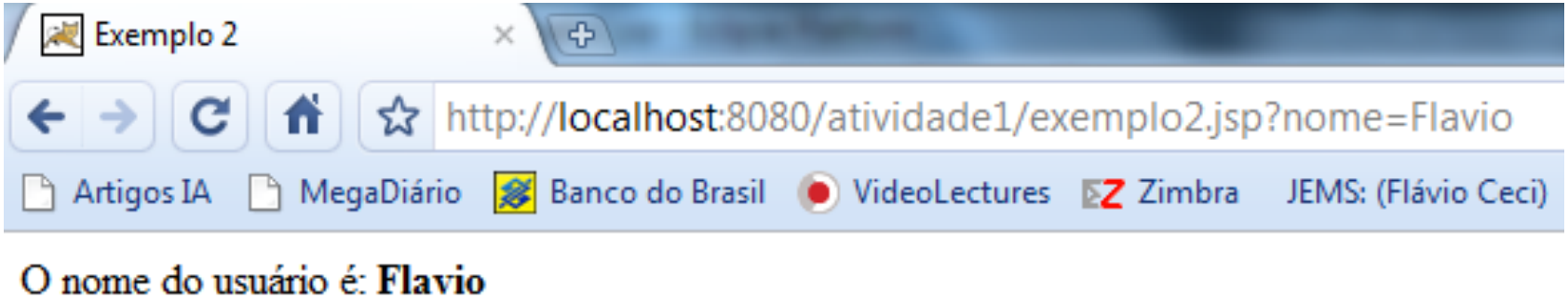
```
1<%@page import="java.util.Date"%>
2<html>
3    <head>
4        <title>Exemplo 1</title>
5    </head>
6    <body>
7        <h1>Exemplo 1 - Data</h1>
8        <p>O horário corrente é: </p><%= new Date()%>
9    </body>
10</html>
```

Variáveis implícitas

- **request** – faz consultas em parâmetros de formulário, cookies de chegada, cabeçalhos de solicitação;
- **response** – manipula a resposta: por exemplo, adiciona cookies, redireciona etc;
- **session** – acessa as informações relacionadas a sessão de navegação;
- **config** – obtém parâmetros de configuração para está página.

Exemplo

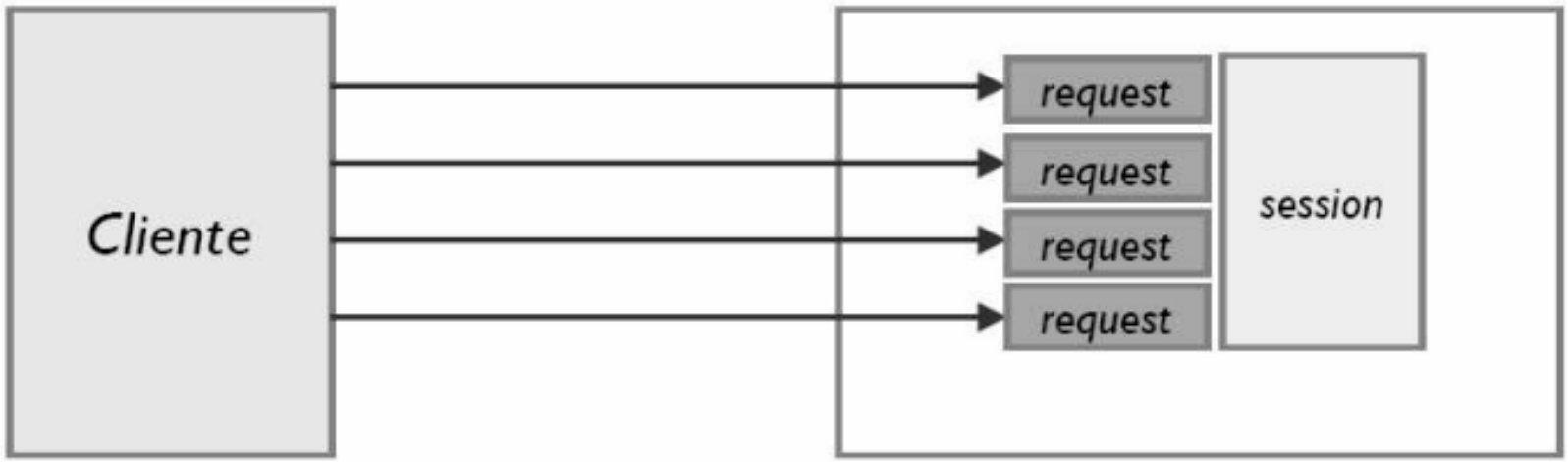
```
1<html>
2    <head>
3        <title>Exemplo 2</title>
4    </head>
5    <body>
6        <%
7            String nome = request.getParameter("nome");
8
9            if(nome == null || nome.trim().length() == 0) {
10               nome = "não informado";
11           }
12       %>
13       O nome do usuário é: <b><%= nome %></b>
14
15   </body>
16</html>
```



Sessões

- Tem como objetivo manter os estados das aplicações WEB;
- Permite que valores sejam comuns a páginas sem que os mesmos sejam passados por parâmetro ou via URL.
- Uma sessão é única para cada cliente e persiste através de várias requisições.

Sessões



Sessões

- Sessões são representados por objetos HttpSession e são obtidas a partir de uma requisição;

- Dois métodos podem ser usados:
  - HttpSession session = request.getSession(false);
  - (Se a sessão não existir, retorna null, caso contrário retorna sessão.)
  - HttpSession session = request.getSession();
  - (Retorna a sessão ou cria uma nova. Mesmo que getSession(true))

Compartilhamento de objetos na sessão

- Dois métodos permitem o compartilhamento de objetos na sessão:
  - setAttribute("nome", objeto);
  - Object getAttribute("nome");
- Exemplo:

Requisição 1

```
String[] vetor = {"um", "dois", "tres"};
HttpSession session = request.getSession();
session.setAttribute("dados", vetor);
```

Requisição 2

```
HttpSession session = request.getSession();
String[] dados = (String[])session.getAttribute("dados");
```

Tomcat 7.0
(Servlet 3 + JSP 2.2)

Tomcat 7

- Suporte as especificações Servlet 3.0;
- Suporte as especificações JSP 2.2;
- Suporte a EL 2.2 (*Expression Language*);
- Suporte a anotações;
- Configurações de plugins via web.xml fragments;
- Servlets assincronos;

Servlet 3

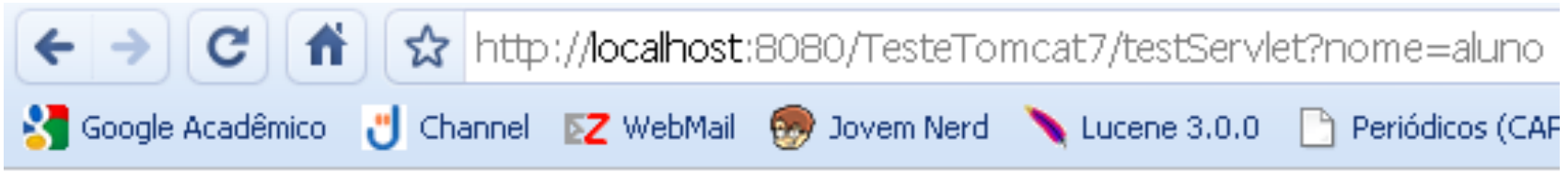
- Não é mais necessário mapear os Servlets no arquivo de configuração web.xml;
- Os servlets são disponibilizado através de anotações nas próprias classes;
- Faz parte da especificação do Java EE 6;

- *@WebServlet* – identifica a classe como um Servlet

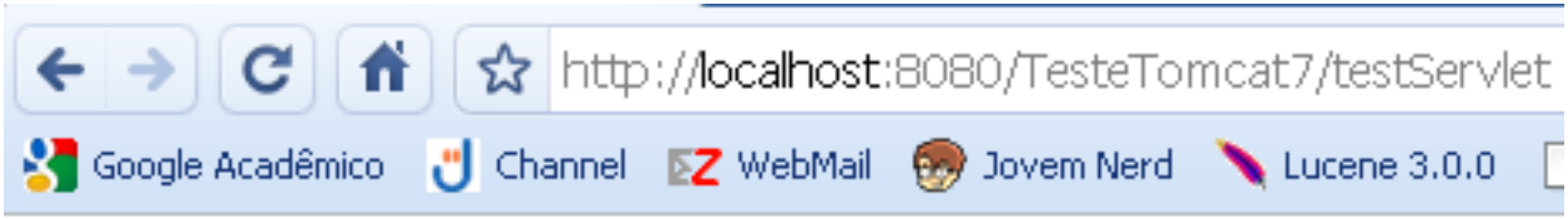
@WebServlet

```java
@WebServlet(name="testServlet", value="/testServlet")
public class TestServlet extends HttpServlet {
    private static final long serialVersionUID = -7128157329737565064L;

    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {

        final PrintWriter out = resp.getWriter();
        try {
            String nome = req.getParameter("nome");
            if(nome != null && nome.trim().length() > 0) {
                out.println("O nome é: "+nome);
            } else {
                out.println("Não foi possivel recuperar o nome");
            }
        } finally {
            if(out != null) {out.close();}
        }
    }
}
```

Exemplo




```
O nome é: aluno
```



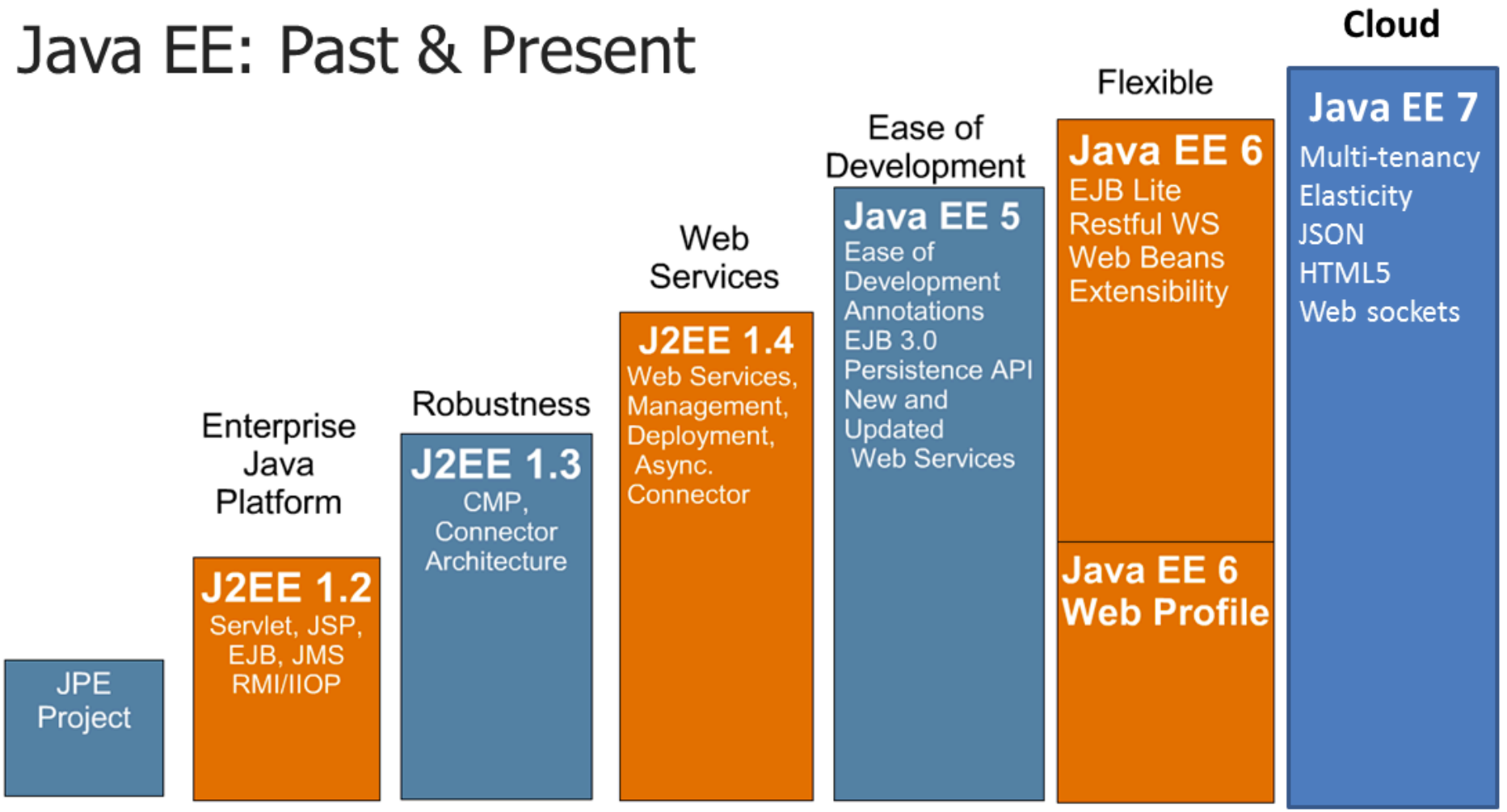

```
Não foi possivel recuperar o nome
```

Introdução ao Java EE

Introdução ao Java EE

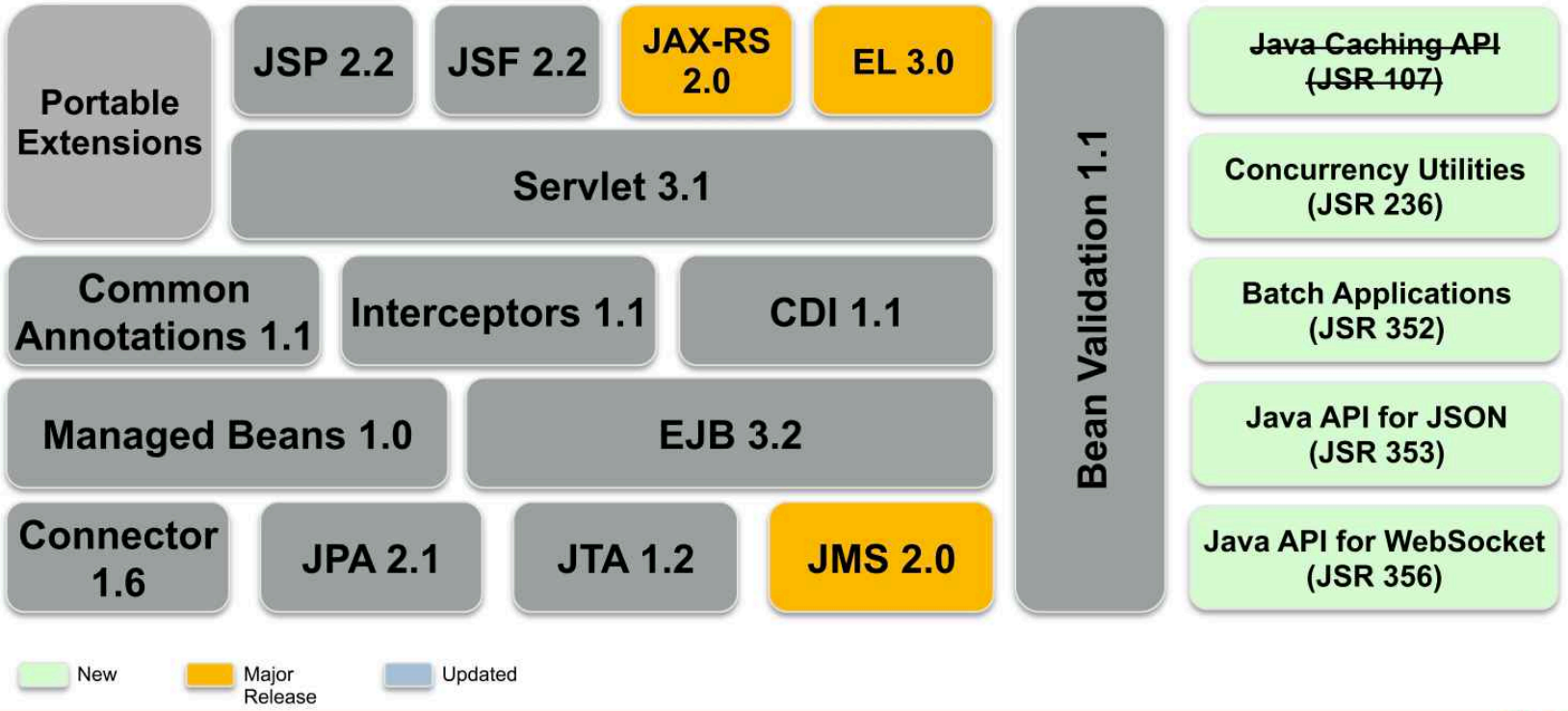
- O **Java EE** (*Enterprise Edition*) é uma plataforma amplamente usada que contém um conjunto de tecnologias coordenadas que reduz significativamente o custo e a complexidade do desenvolvimento, implantação e gerenciamento de aplicativos de várias camadas centrados no servidor.
- O **Java EE** é construído sobre a plataforma **Java SE** e oferece um conjunto de **APIs** (interfaces de programação de aplicativos) para desenvolvimento e execução de aplicativos **portáteis**, **robustos**, **escaláveis**, **confiáveis** e **seguros** no **lado do servidor.**

Evolução do Java EE



Java EE 7

Java EE 7 – Candidate JSRs



Preparando o projeto

Preparando o projeto

- Vamos a alguns pontos conhecidos:
  - O JSF não é uma implementação nativa do JDK;
  - Devemos adicionar as *libs* aos nossos projetos;
  - Muitas vezes quando utilizamos *libs* externas, as mesmas tem dependência com outras *libs* de terceiros;
    - Quais versões das *libs* são compatíveis?
    - Onde encontrá-las?
    - **Não existe uma forma mais fácil de gerenciarmos isso???**


Apache Maven Project
http:// maven.apache.org


Apache
maven

Apache Maven

- É uma ferramenta de automação de compilação;
- É similar à ferramenta Ant;
- É utilizado um arquivo XML (POM) para descrever o projeto de software a ser construído;
- São baixadas dinamicamente as dependências;
- Facilita o deploy da aplicação;

Fonte: http://bit.ly/1BX36EL

Archetypes (arquétipos)

- São estruturas comuns que definem uma solução para um domínio conhecido;
- Pode-se dizer que é um estereótipo de projetos;
- No nosso caso:
  - Uma estrutura de pastas que representam uma solução:
    - Aplicação Web;
    - Aplicação Mobile...

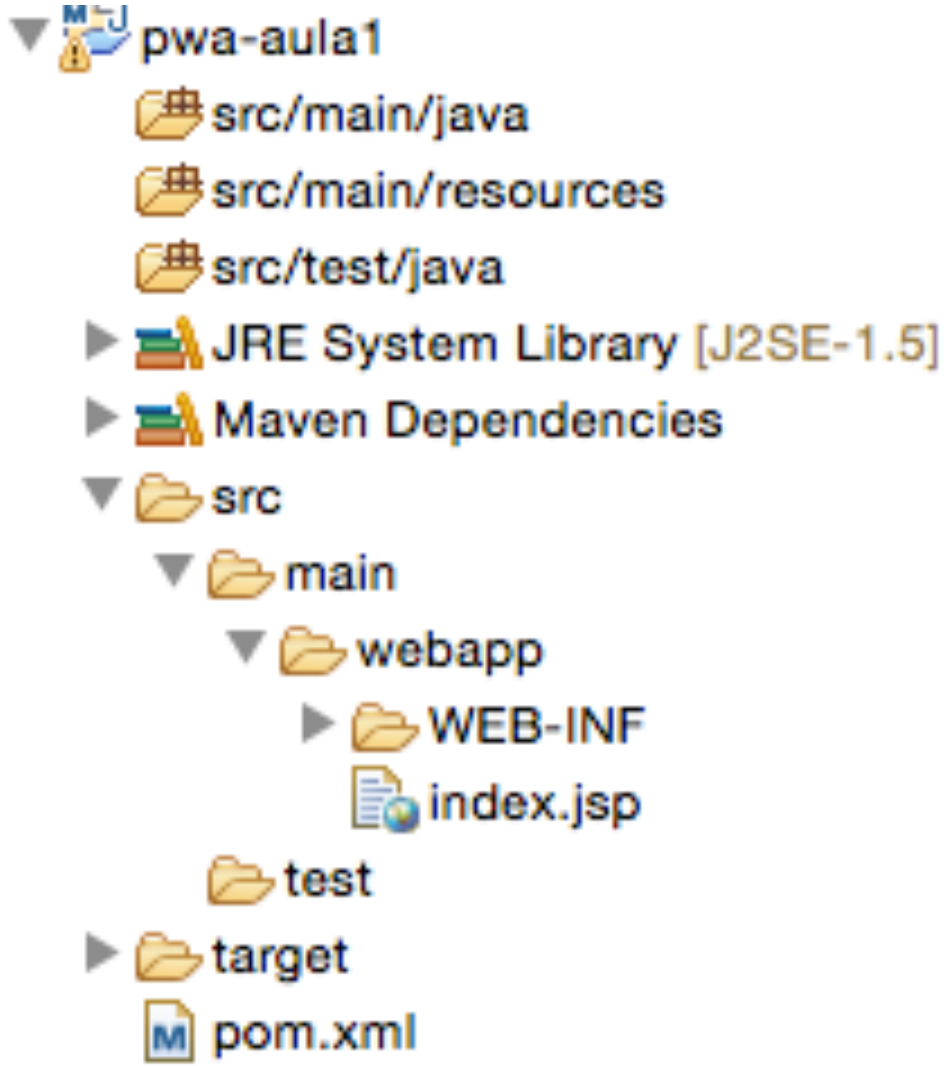
Informações fundamentais

- **groupId**: Informa qual a empresa que desenvolve o artefato;
  - Informação é utilizada como base para os pacotes;
  - Ex.: br.unisul.pwa
- **artifactId**: Identifica o artefato;
  - Ex.: aula1
- **version**: Versão do artefato de código;

Estrutura de diretórios

- *src/main/java*: local onde fica o código fonte;
- *src/main/resources*: arquivos de configuração;
- *src/main/webapp*: estrutura web
- *src/test/java*: local ontem ficam as classes de teste;
- *src/teste/resources*: arquivos de configuração
- *src/site*: documentação
- *target*: Pacotes gerados e fontes compilados.

Estrutura de diretórios



POM.xml

- Centraliza todas as configurações;
- Possui a seguinte estrutura:

```
1  <project>
2      <modelVersion>4.0.0</modelVersion>
3      <groupId>br.unisul.pwa</groupId>
4      <artifactId>aula1</artifactId>
5      <version>1.0</version>
6  </project>
7
```

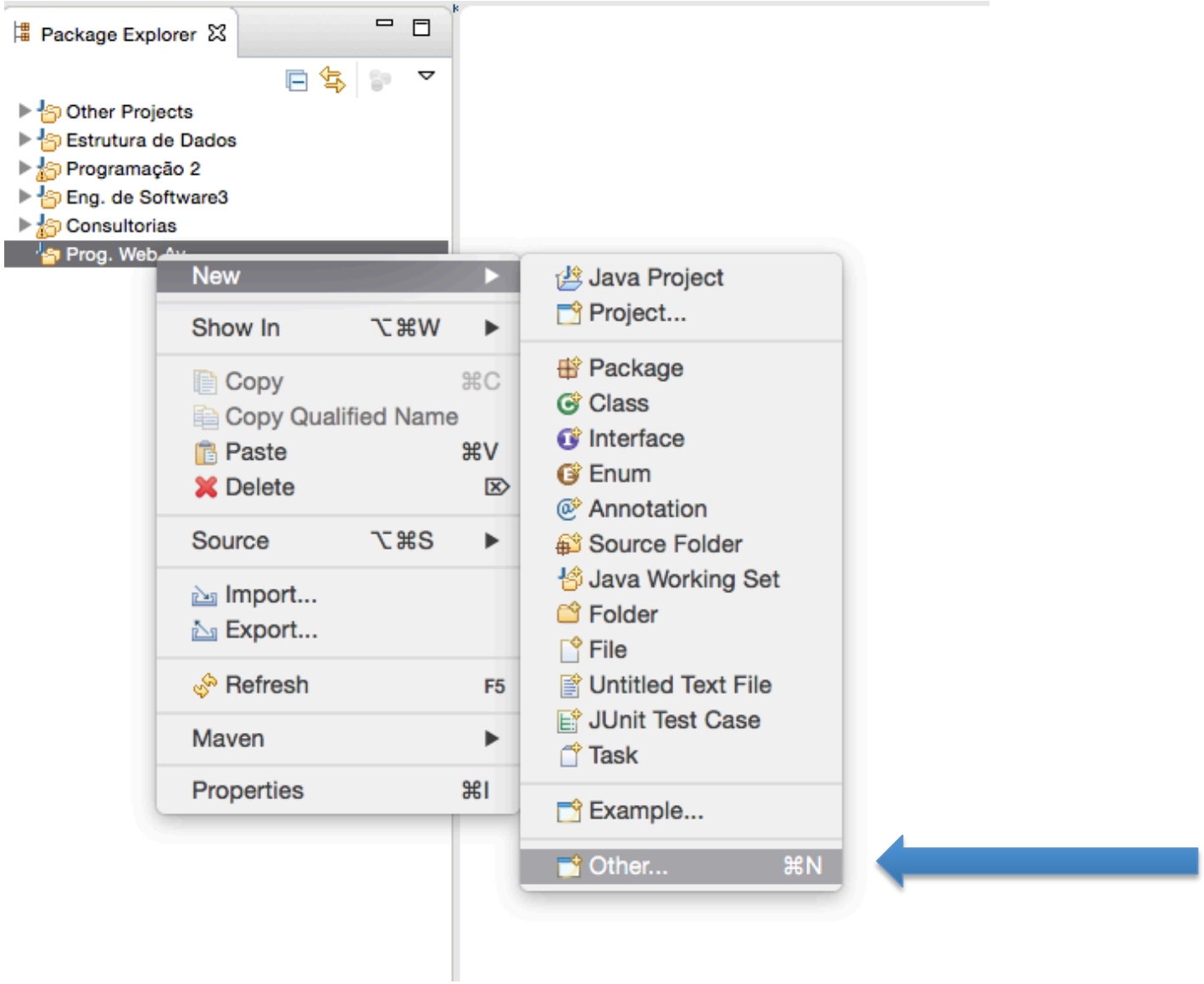
Gestão de dependência

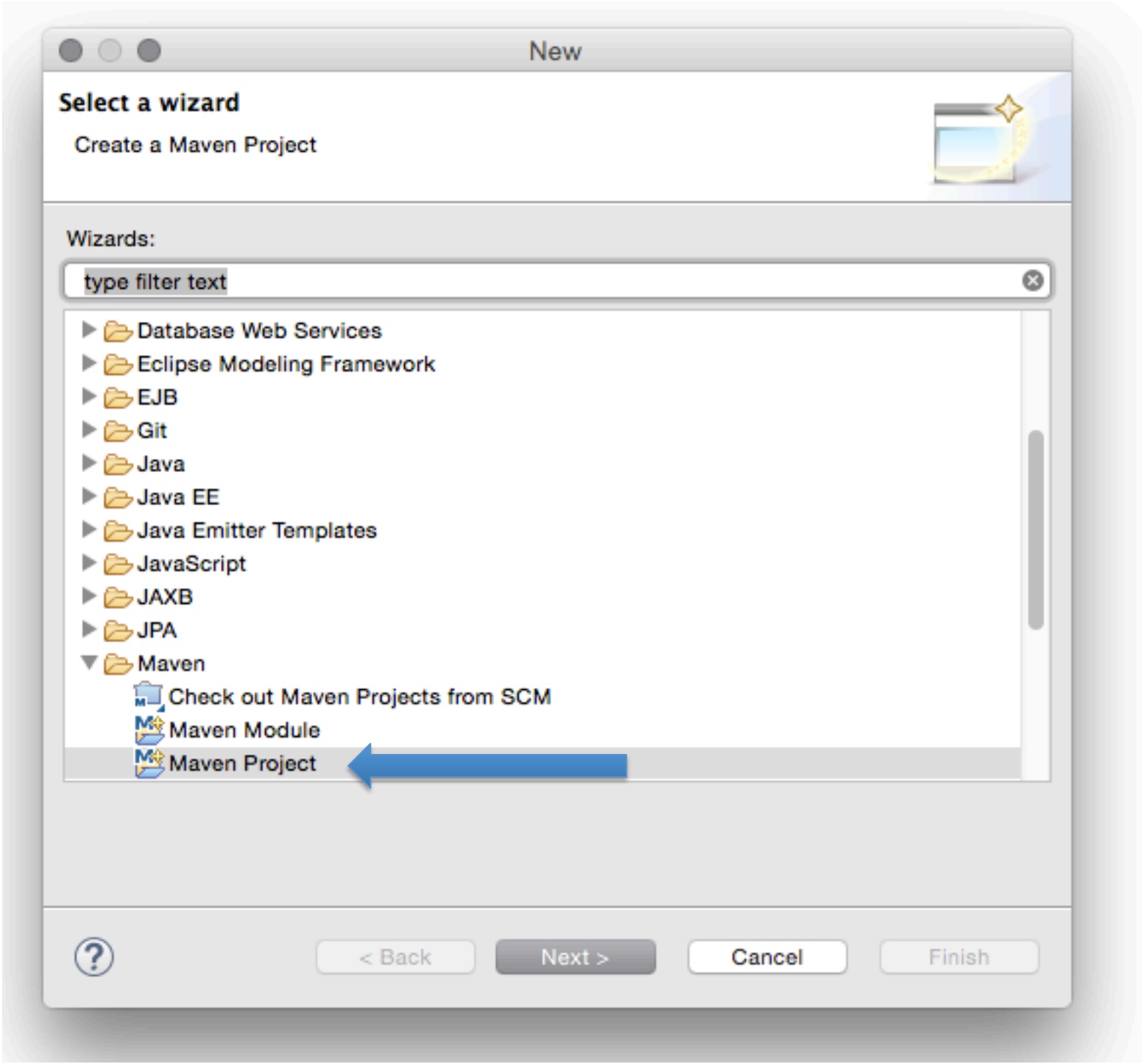
- São utilizados dois repositórios:
  - *Local Artifact Repository*;
    - Mantem uma cópia das libs configuradas na máquina onde se está desenvolvendo o projeto;
  - *Remote Artifact Repository*;
    - Local onde será buscado as versões das libs para serem incorporadas no novo projeto;
    - Quando se deseja construir uma lib que será utilizado por outros projetos, é necessário publicá-la em um repositório remoto.

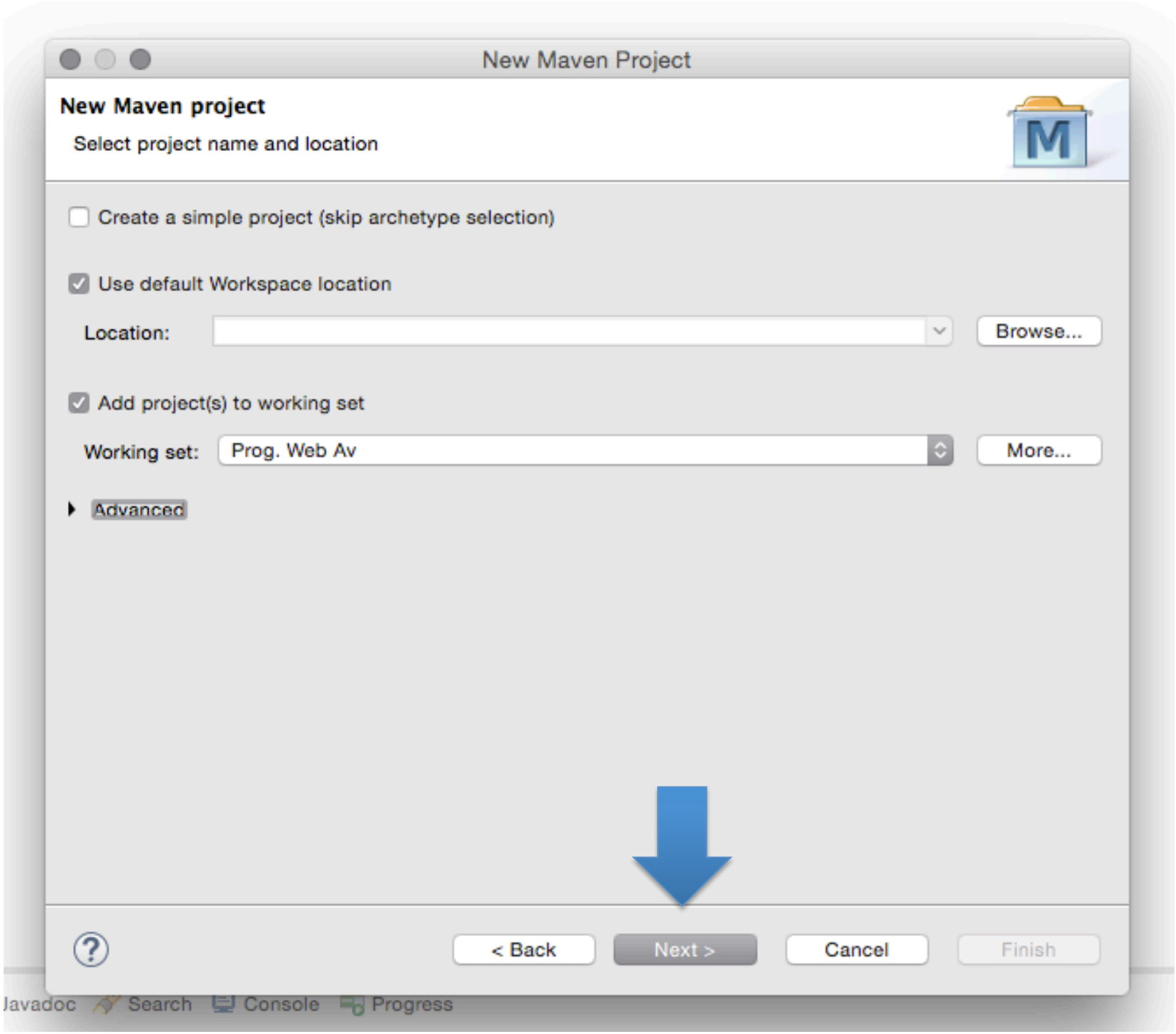
Gestão de dependência

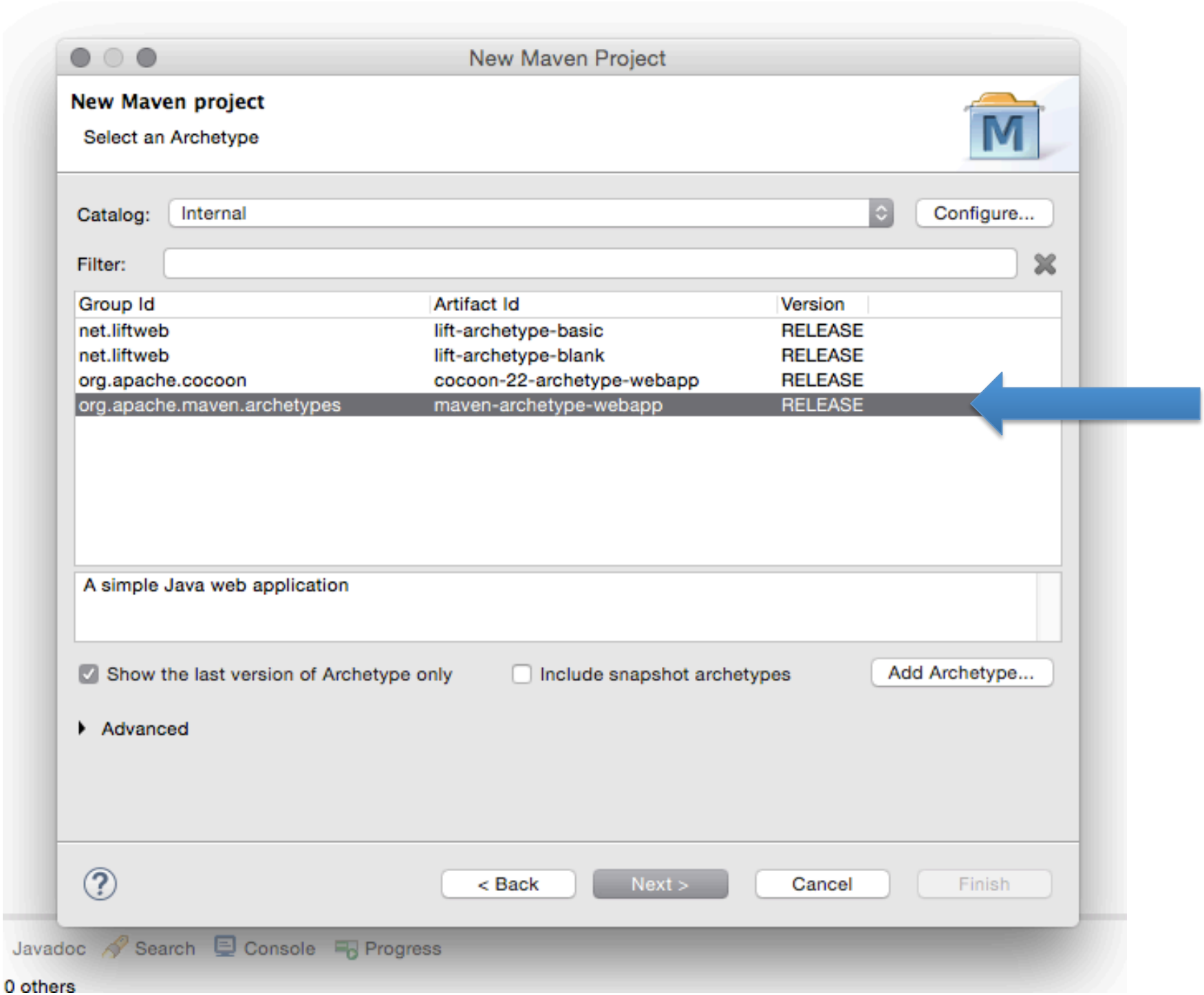
```xml
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>br.unisul.pwa</groupId>
    <artifactId>aula1</artifactId>
    <version>1.0</version>

    <dependencies>
        <dependency>
            <groupId>javax.faces</groupId>
            <artifactId>jsf-api</artifactId>
            <version>1.2</version>
            <type>jar</type>
        </dependency>
    </dependencies>

</project>
```

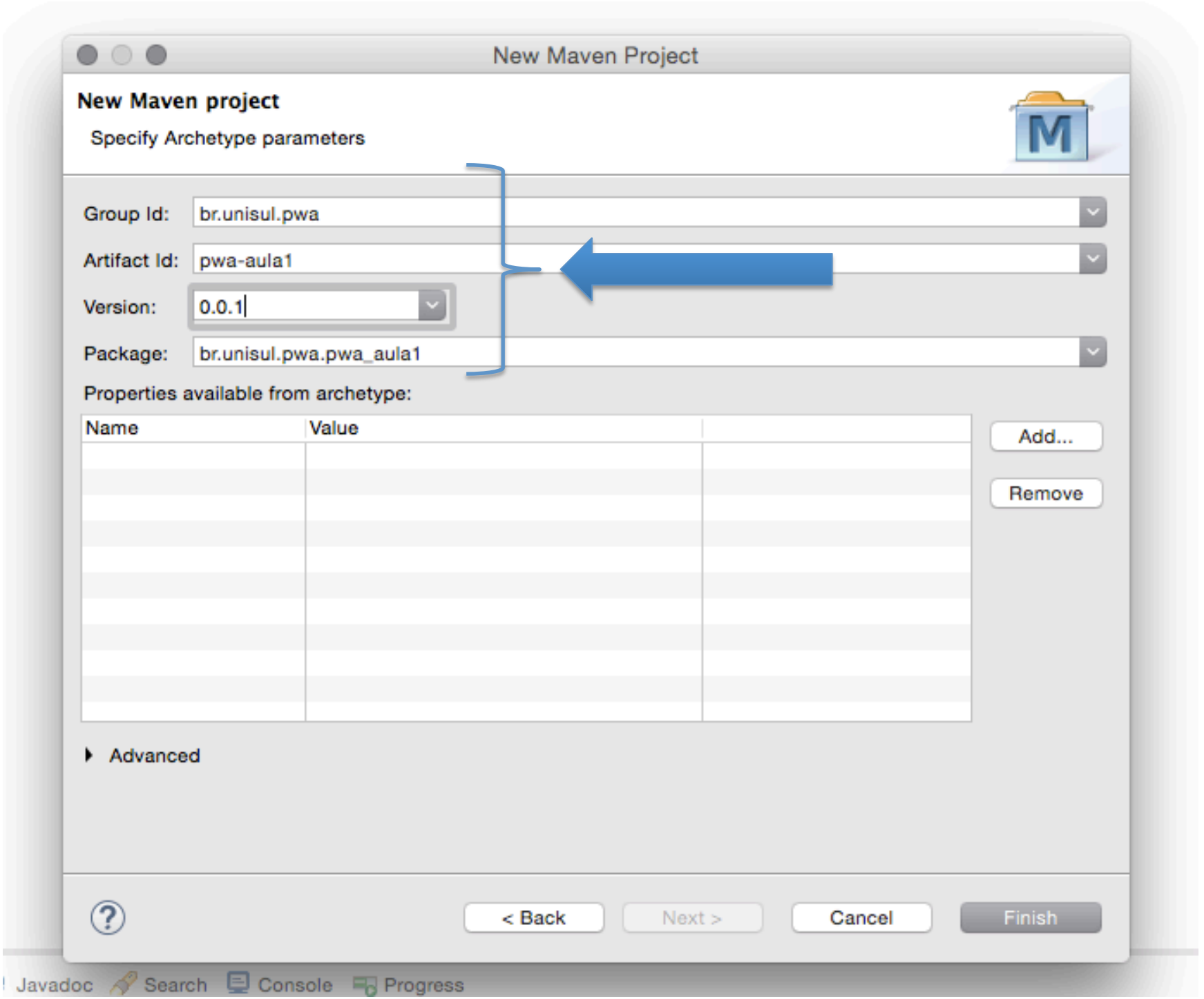
Montando o projeto no eclipse




New
Select a wizard
Create a Maven Project
Wizards:
type filter text
Database Web Services
Eclipse Modeling Framework
EJB
Git
Java
Java EE
Java Emitter Templates
JavaScript
JAXB
JPA
Maven
Check out Maven Projects from SCM
Maven Module
Maven Project
< Back
Next >
Cancel
Finish


New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
Prog. Web Av
More...
Advanced
< Back
Next >
Cancel
Finish
Javadoc
Search
Console
Progress


New Maven Project
New Maven project
Select an Archetype
Catalog:
Internal
Configure...
Filter:
Group Id
Artifact Id
Version
net.liftweb
lift-archetype-basic
RELEASE
net.liftweb
lift-archetype-blank
RELEASE
org.apache.cocoon
cocoon-22-archetype-webapp
RELEASE
org.apache.maven.archetypes
maven-archetype-webapp
RELEASE
A simple Java web application
Show the last version of Archetype only
Include snapshot archetypes
Add Archetype...
Advanced
< Back
Next >
Cancel
Finish
Javadoc
Search
Console
Progress
0 others


New Maven Project
New Maven project
Specify Archetype parameters
Group Id:
br.unisul.pwa
Artifact Id:
pwa-aula1
Version:
0.0.1
Package:
br.unisul.pwa.pwa_aula1
Properties available from archetype:
Name
Value
Add...
Remove
Advanced
< Back
Next >
Cancel
Finish
Javadoc
Search
Console
Progress

Dependências

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>
        <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>javax.faces</groupId>
        <artifactId>jsf-api</artifactId>
        <version>1.2</version>
        <scope>provided</scope>
    </dependency>

    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>javax.servlet-api</artifactId>
        <version>3.0.1</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

Exercícios

Exercício 1

- Construa um Projeto Web utilizando Maven na sua IDE de preferencia.

- Crie uma página simples no estilo “Olá mundo” para avaliar se a aplicação foi criada corretamente.

Exercício 2

- Desenvolva uma aplicação web para controlar o agendamento de consultas de um consultório médico, o sistema deve:
  - Ter uma página para cadastrar no máximo 10 pacientes em um dia, a sequencia de atendimento é definida pela ordem de cadastro.
  - Ter uma página para listar a partir de uma consulta os pacientes cadastrados para o dia consultado.
  - Ter uma pagina com os nomes dos pacientes que o médico deve atender e um combobox para ele mudar o status (pentende/atendido)
  - Desenvolver utilizando as collections do Java.