DESENVOLVIMENTO DE WEB SERVICES COM ECLIPSE

LUIS HIDEO VASCONCELOS NAKAMURA
JÚLIO CÉZAR ESTRELLA
MARCOS JOSÉ SANTANA
REGINA HELENA CARLUCCI SANTANA

Nº 80

NOTAS DIDÁTICAS

São Carlos - SP

Desenvolvimento de Web Services com Eclipse

Luis Hideo V. Nakamura, Júlio Cézar Estrella, Marcos José Santana, Regina Helena Carlucci Santana

Departamento de Sistemas de Computação
Grupo de Sistemas Distribuídos e Programação Concorrente
Universidade de São Paulo - Instituto de Ciências Matemáticas e de Computação
{nakamura, jcezar, mjs, rcs}@icmc.usp.br
http://www.lasdpc.icmc.usp.br

Resumo Este documento tem o intuito de demonstrar o desenvolvimento de *Web Services* utilizando a linguagem Java com o apoio de um ambiente de desenvolvimento integrado denominado Eclipse. O foco deste trabalho está centrado na parte prática da criação de *Web Services* incluindo a configuração do ambiente de desenvolvimento e das ferramentas envolvidas na sua construção. Os *Web Services* são cada vez mais utilizados em aplicações distribuídas, principalmente naquelas onde o foco é o reuso e a interoperabilidade. Neste contexto, nota-se a importância e a necessidade do conhecimento de como criá-los de forma mais produtiva e eficiente. Para atender a esta finalidade este documento foi criado em forma de tutorial no qual é explicado o passo a passo do desenvolvimento por meio da combinação do Eclipse com o projeto WTP (*Web Tools Platform*).

1 Introdução

Atualmente, há diversas formas de desenvolvimento de *Web Services* em Java, e várias ferramentas visam facilitar a implementação desses serviços de forma produtiva.

O desenvolvedor tem a opção de escolher as ferramentas e formas de desenvolvimento que melhor se encaixem às suas necessidades. Apesar da criação de *Web Services* não exigir a instalação de um IDE (*Integrated Development Environment*)[1], a sua utilização é muito comum em ambientes acadêmicos e na indústria, visando um desenvolvimento mais rápido e eficiente.

Este relatório, em caracter técnico, foi elaborado com o intuito de fornecer aos usuários iniciantes uma breve introdução à utilização da IDE Eclipse[2] para o desenvolvimento de *Web Services* em Java.

Este documento é formado por seis capítulos. O segundo apresenta uma breve revisão sobre o tema "*Web Services*". O terceiro capítulo é voltado à configuração do ambiente de desenvolvimento apresentando algumas das ferramentas utilizadas. O quarto capítulo apresenta o passo a passo do desenvolvimento de um *Web Service* utilizando o Eclipe em conjunto com o WTP (*Web Tools Platform*)[3]. O quinto capítulo descreve o passo a passo da criação da aplicação Cliente que consume o serviço criado anteriormente. Finalmente, o sexto e último capítulo é destinado as conclusões sobre a utilização do Eclipse e WTP para a criação de *Web Services* em Java.

2 Web Services

Os *Web Services* são componentes ou unidades lógicas de aplicações, acessíveis por meio de protocolos baseados em padrões abertos disponíveis da Internet[6]. Eles são implementados adotando-se a arquitetura orientada a serviços (SOA - *Service Oriented Architecture*); ou seja, os *Web Services* são uma implementação de SOA [5].

A padronização de uma arquitetura SOA propociona a interoperabilidade, o reuso e facilita a implementação e manutenção dos serviços. A arquitetura de um *Web Service* é composta por, no mínimo, um provedor de serviços e clientes que acessam os serviços prestados. Além disso, pode haver um terceiro elemento, conhecido como registro UDDI (*Universal Description, Discovery, and Integration*), que presta o serviço de armazenamento de informações de *Web Services* [9]. Em princípio os clientes podem não conhecer os provedores de serviços e nem saber quais são os *Web Services* disponibilizados. Para descobrir informações a respeito de um determinado serviço, o cliente deve consultar o registro UDDI, que retorna uma interface de descrição do serviço (WSDL - *Web Services Description Language*). A WSDL[4] contém as informações necessárias para que o cliente

[1] Os Web Services e seus Clientes podem ser criados com a edição de arquivos e por meio de ferramentas que geralmente podem ser executadas por linha de comando, conforme pode ser observado em [6]

[2] http://www.eclipse.org

[3] http://www.eclipse.org/webtools/

[4] A WSDL especifica diversas informações funcionais como, por exemplo, o endereço do provedor, a forma de acesso ao serviço, portas de acesso e operações disponíveis.

consiga acessar e utilizar o serviço corretamente. Por este motivo, o provedor de serviços deve publicar as WSDLs de seus serviços no registro UDDI. A Figura 1 exibe o modelo básico de uma arquitetura orientada a serviços caracterizada para *Web Services*.

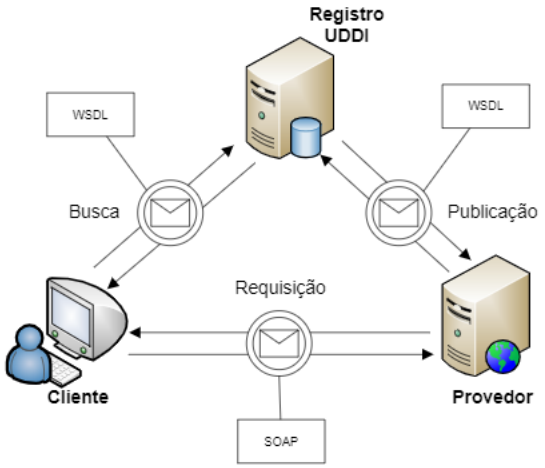

Figura 1. Modelo de SOA para *Web Services* [9].

Outra característica interessante dos *Web Services* é o seu acoplamento fraco, isso significa que não há grandes dependências entre os componentes, o que permite uma maior flexibilidade no caso de alterações [8]. A comunicação entre os *Web Services* e os Clientes (consumidores de serviço) também acontece de forma independente de linguagem de programação, plataforma ou sistema operacional. Dessa forma, a interoperabilidade é alcançada, possibilitando a comunicação de diversas aplicações distintas sem grandes modificações. Estes benefícios são obtidos em razão da padronização e utilização de protocolos abertos que exercem um papel fundamental na comunicação. Geralmente, estes protocolos são escritos em linguagem XML (*Extensible Markup Language*), que podem ser interpretada e processada por diversas linguagens de programação (Java, C, C++, C#, PHP, Python, etc). Os principais protocolos empregados em uma arquitetura de *Web Services* são:

- **Protocolo SOAP** (*Simple Object Access Protocol*): protocolo empregado na troca de mensagens entre *Web services*, permite a comunicação de forma simples e independente de linguagem ou plataforma [10];
- **Linguagem WSDL** (*Web Services Description Language*): linguagem que provê uma interface de descrição dos serviços em um *Web Service*. Ela especifica como o serviço deve ser acessado e quais as operações disponíveis [12];
- **Registro UDDI** (*Universal Description, Discovery, and Integration*): Um registro (uma base de dados ou diretório) que armazena as descrições de *Web Services*. Ele é utilizado por clientes e provedores para localizar e publicar serviços, respectivamente [7].

Geralmente, no processo de desenvolvimento de um *Web Services*, o desenvolvedor não precisa se preocupar com o processo de criação, empacotamento, envio e desempacotamento das mensagens SOAP, pois na maioria dos casos são empregados motores de processamento SOAP que possuem mecanismos desenvolvidos para executar tais tarefas. Além disso, a WSDL também pode ser gerada automaticamente por meio de ferramentas ou até mesmo pelo próprio motor de processamento SOAP[5].

3 Configuração do Ambiente

Este capítulo tem como objetivo auxiliar o usuário iniciante na configuração do ambiente de desenvolvimento de *Web Services* com o Eclipse + Web Tools Platform, Axis2, Apache Tomcat, Axis2 Service Archiver e Java.

3.1 Java

O desenvolvimento de código apresentado neste documento será na linguagem Java. O primeiro passo que deve ser realizado é instalar o Kit de Desenvolvimento para Java (JDK - *Java Development Kit*) da Sun/Oracle. O JDK é fornecido pela Oracle[6] e está disponível para *download* em:

◇ `http://www.oracle.com/technetwork/java/javaee/downloads/index.html`

Após a instalação do JDK, o usuário deve configurar o *Classpath*, ou seja, configurar as variáveis de ambiente necessárias para o desenvolvimento em Java. Este processo encontra-se detalhado no relatório técnico "*Web Services Tutorial*" [6].

Para usuários do sistema operacional Linux, mais especificamente aqueles que utilizam a distribuição Ubuntu, há uma forma mais automática de instalação e configuração do JDK. Uma vez conectado a Internet, o usuário precisa apenas digitar o seguinte comando[7]:

◇ `sudo apt-get install sun-java6-jdk`

A senha de *root* (super-usuário) é exigida para que o JDK seja descarregado, instalado e configurado automaticamente. Os exemplos apresentados neste relatório técnico foram criados no Ubuntu 10.04 LTS. É comum que a máquina virtual Java (JVM) já esteja instalada no sistema. Uma forma de verificar se a JVM está instalada e descobrir a sua versão é por meio do comando:

◇ `java -version`

Uma mensagem parecida com "*java version 1.6.0_20*" será exibida caso a JVM esteja instalada. O compilador Java, denominado javac, é instalado com o JDK. Para verificar se ele está instalado e descobrir qual é a sua versão, basta digitar o seguinte comando:

[5] O Axis2 é um exemplo de motor (*engine*) de processamento SOAP e WSDL para *Web Services* (http://axis.apache.org/axis2/java/core/)

[6] http://www.oracle.com/

[7] O mesmo comando pode ser utilizando nas distribuições Kubuntu e Xubuntu

◇ `javac -version`

Um mensagem parecida com "*javac 1.6.0_24*" exibe a versão do javac.

3.2 Eclipse

O Eclipse é um dos ambientes de desenvolvimento integrado mais completo existente atualmente. Ele é desenvolvido na linguagem Java e tornou-se um dos ambientes mais populares para o desenvolvimento de aplicações nesta linguagem. Porém, ele também permite a programação em diversas outras linguagens como C, PHP, etc. O projeto Eclipse foi criado pela IBM (*International Business Machines*), mas atualmente é hospedado pela comunidade de código aberto Eclipse (*Eclipse Foundation*), composta por indivíduos e organizações [3].

A partir deste ponto, durante todo este documento, o termo Eclipse será utilizado para referenciar a plataforma de desenvolvimento Eclipse. O termo também irá englobar, associada à IDE, a extensão WTP (*Web Tools Platform*). Como dito anteriormente, a WTP é uma plataforma de extensão do projeto Eclipse, que conta com ferramentas criadas com o objetivo de desenvolver aplicações Web e Java EE (*Enterprise Edition*). Para auxiliar os desenvolvedores, esta plataforma oferece editores gráficos e de código fonte, além de diversos *wizards*[8], ferramentas e APIs que facilitam a implantação (*deploy*), execução e teste das aplicações [4].

Há diversas versões do Eclipse disponíveis para *download*. A versão utilizada na criação dos exemplos deste documento é a *GALILEO*. Esta e outras versões mais recentes podem ser encontradas no seguinte endereço:

◇ `http://www.eclipse.org/downloads/`

A versão Galileo pode ser obtida diretamente em:

◇ `http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/galileo/SR2/eclipse-jee-galileo-SR2-linux-gtk.tar.gz`

A instalação do Eclipse é simples, sendo preciso apenas descompactar o arquivo de extensão tar.gz em um diretório da preferência do usuário. Para descompactar o arquivo, utiliza-se o seguinte comando:

◇ `tar -zxvf eclipse-jee-galileo-SR2-linux-gtk.tar.gz`

Um novo subdiretório com o nome "eclipse" será criado no diretório corrente. Para executar o Eclipse, deve-se entrar neste novo diretório e executar o comando:

◇ `./eclipse`

A aplicação irá inicializar e carregar a interface gráfica. Antes de abrir o ambiente de desenvolvimento, uma janela do tipo *pop-up* é aberta para que o usuário informe a localização do *workspace* que será utilizado. O *workspace* é um diretório padrão no qual o Eclipse armazena os projetos do usuário, bem como suas preferências. A Figura 2 mostra a tela inicial do Eclipse requisitando ao usuário a localização do *workspace*.

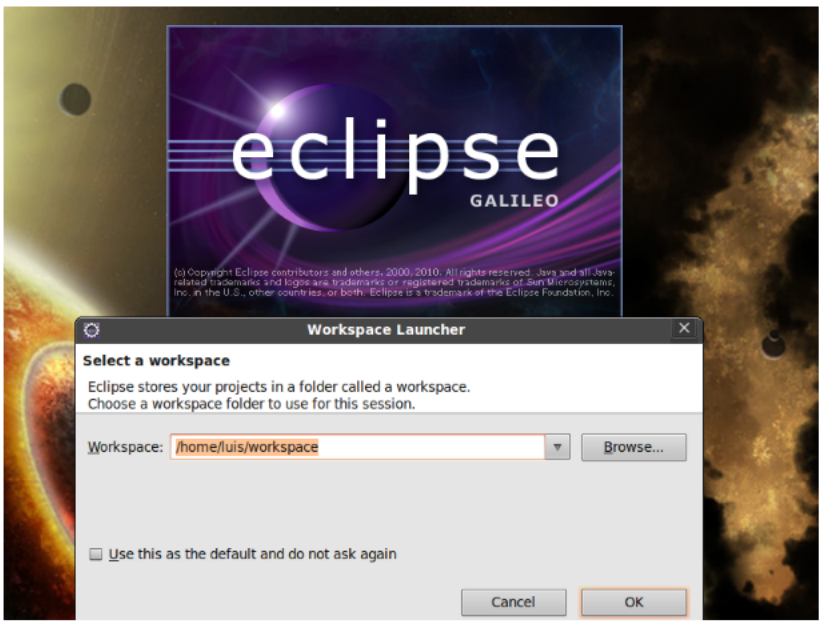

Figura 2. Janela do Eclipse requisitando o endereço do *Workspace*.

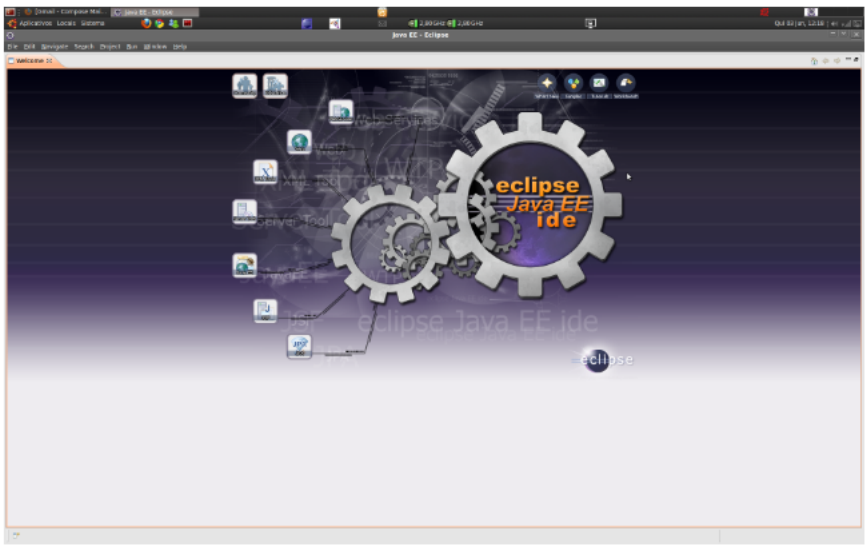

Figura 3. Janela de Boas Vindas do Eclipse (*Welcome*).

[8] Os *wizards* são mecanismos que guiam o usuário durante a execução ou configuração de algum processo. O processo é dividido em etapas ou passos que seguem um determinado fluxo lógico.

Após escolher o caminho do *workspace* e clicar sobre o botão "OK", o Eclipse criará o *workspace* do usuário (caso ele não exista) e abrirá a interface de boas vindas (*welcome*), conforme exibe a Figura 3.

Caso a extensão WTP ainda não esteja instalada, o usuário deve ir ao menu "*Help*" e selecionar a opção "*Install New Software...*". Uma nova janela com o título de *Install* é aberta, com o botão "*Add*" que abre uma caixa de dialogo para que seja adicionada uma descrição e o endereço do projeto WTP[9]. A Figura 4 ilustra estas etapas de instalação da extensão WTP.

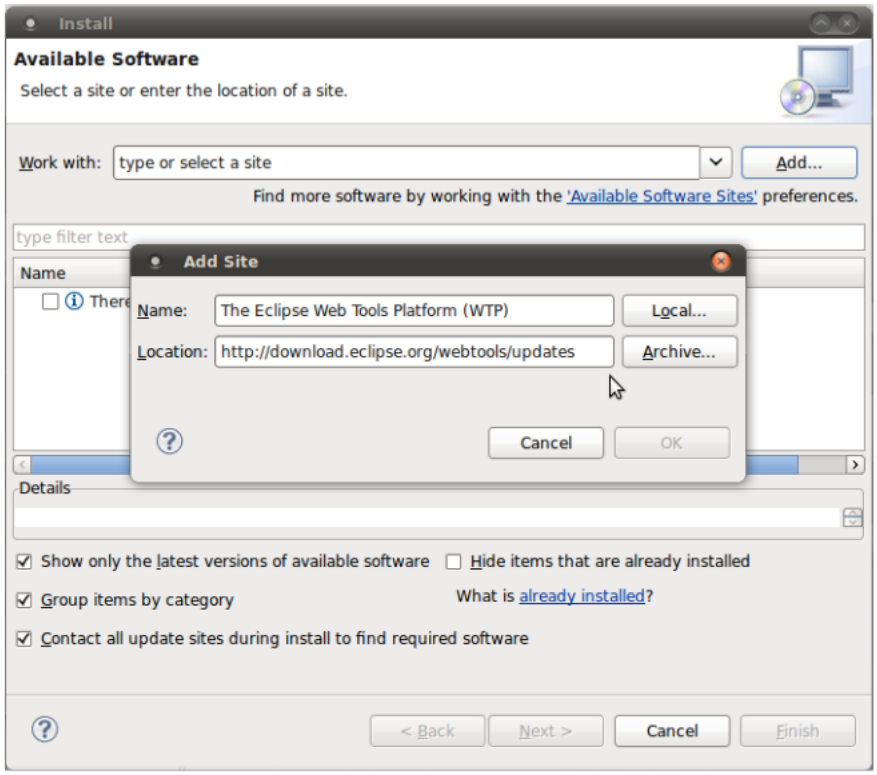

Figura 4. Janela de Instalação de *Softwares* (neste caso o WTP).

Ainda na tela de boas vindas do Eclipse há o botão "*Workbench*" que encaminha o usuário para a perspectiva de desenvolvimento de projetos. Mas, antes de iniciar um novo projeto, é recomentado terminar de instalar as outras ferramentas e programas que são necessários para o desenvolvimento dos *Web Services*. O próximo programa a ser instalado é o Apache Tomcat, e o procedimento que deve ser seguido para a sua instalação está descrito na próxima seção.

[9] http://download.eclipse.org/webtools/updates

3.3 Tomcat

O Apache Tomcat, ou simplesmente Tomcat, é um servidor de aplicação para implementações em Java como *Servlets* e páginas JSP (*JavaServer Pages*) [13]. O escopo deste documento não cobre a instalação e configuração completa ou detalhada do Tomcat; apenas uma forma de instalação simplificada do Tomcat será utilizada para que trabalhe juntamente com o Eclipse. O ponto é ter um ambiente de desenvolvimento funcional utilizando Tomcat e Axis2 que seja propício à criação de *Web Services*.

O *download* da versão 6 do Apache Tomcat, utilizada nos exemplos deste documento, pode ser feito no seguinte endereço:

◇ `http://linorg.usp.br/apache/tomcat/tomcat-6/`

Após realizar o *download* do binário compactado do Apache Tomcat, é preciso copiá-lo para o diretório de preferência do desenvolvedor, e então descompactá-lo. Para isso, execute o seguinte comando:

◇ `tar -zxvf apache-tomcat-6.0.26.tar.gz`

Um novo subdiretório com o nome de "*apache-tomcat-6.0.26*" será criado no diretório onde o arquivo foi descompactado. Dentro deste subdiretório, existem diversos subdiretórios dos quais três deles merecem destaque:

1. **bin:** Diretório onde estão localizados os arquivos para a inicialização e finalização do Tomcat; dentre eles, destacam-se:
   a. **startup.sh**: *Shell Script* para inicialização do servidor.
   b. **shutdown.sh**: *Shell Script* para a finalização do servidor.
2. **conf:** Diretório onde ficam localizados os arquivos de configuração. É recomendável que o servidor seja configurado corretamente; porém, esta configuração está além do escopo deste documento.
3. **webapps:** Diretório que armazena as aplicações Web, como páginas JSP e *Servlets*. É nesse diretório que o Axis2 será depositado.

Para testar se o servidor Apache Tomcat foi instalado corretamente, o usuário deverá entrar no diretório "*../apache-tomcat-6.0.26/bin*" e executar o seguinte comando:

◇ `./startup.sh`

O retorno do comando deverá ser algo próximo do contéudo da Listagem 1.

Se não ocorreu nenhum problema, abra um navegador (*browser*) e digite o endereço ***http://localhost:8080***. A página inicial do Apache Tomcat deverá ser exibida, conforme a Figura 5. O próximo passo é a instalação do Axis2, que será abordada na próxima seção.

Listagem 1 Retorno da inicialização do Apache Tomcat

```
luis@Broker:~/apache-tomcat-6.0.26/bin$ ./startup.sh
Using CATALINA_BASE:   /home/luis/apache-tomcat-6.0.26
Using CATALINA_HOME:   /home/luis/apache-tomcat-6.0.26
Using CATALINA_TMPDIR: /home/luis/apache-tomcat-6.0.26/temp
Using JRE_HOME:        /usr
Using CLASSPATH:       /home/luis/apache-tomcat-6.0.26/bin/bootstrap.jar
```

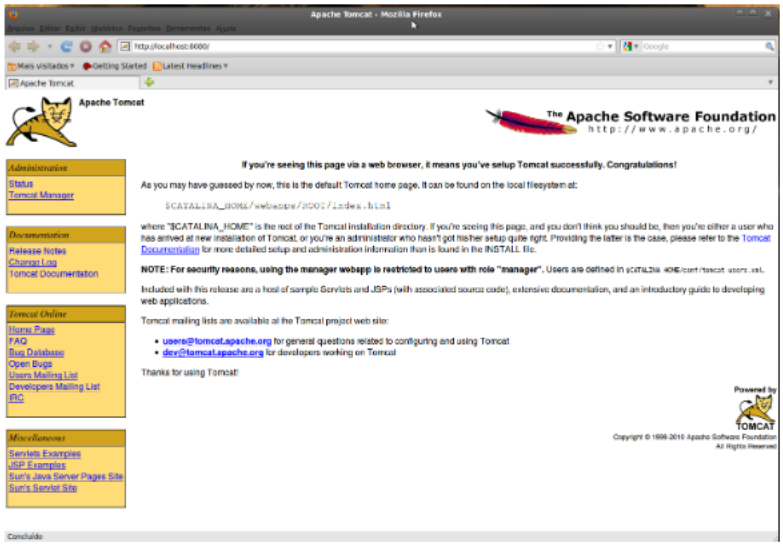

Figura 5. Página inicial do Apache Tomcat.

3.4 Axis2

A instalação do Axis2 é tão simples quanto a instalação do Apache Tomcat. O primeiro passo é fazer o *download* do arquivo no seguinte endereço:

◇ `http://archive.apache.org/dist/ws/axis2/1_4_1/axis2-1.4.1-war.zip`

No diretório de destino, o desenvolvedor deverá descompactar o arquivo com o seguinte comando:

◇ `unzip axis2-1.4.1-war.zip`

Um novo arquivo com o nome de "*axis2.war*" é criado no diretório corrente. Porém, este arquivo deve ser movido para dentro do diretório *webapps* do Tomcat. A mudança de diretório do arquivo pode ser feita da seguinte forma:

◇ `mv axis2.war /home/luis/apache-tomcat-6.0.26/webapps/`

O Apache Tomcat fará o *deploy* do Axis2 de forma automática uma vez que o arquivo foi copiado para a pasta "*webapps*" corretamente. Para confirmar se o *deploy* foi realizado com sucesso, é preciso acessar a URI: ***http://localhost:8080/axis2/***.

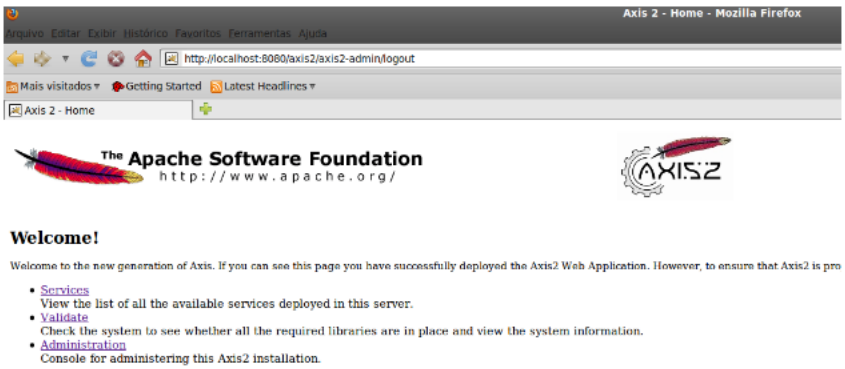

Figura 6. Página inicial do Axis2.

Uma página inicial do Axis2 semelhante a da Figura 6 será exibida, contendo três opções:

1. Visualizar a lista de serviços disponíveis no servidor (*link* **Services**);
2. Validar o sistema e checar se as bibliotecas requeridas estão instaladas, além de poder ver informações do sistema (*link* **Validate**);
3. Administrar o servidor do Axis2, inclusive tendo como opção realizar o *upload* de serviços web (*link* **Administration**). O *login* padrão do administrador é "*admin*" e a senha é "*axis2*", da mesma forma que no servidor Tomcat, é altamente recomendável que a senha seja trocada e o servidor seja configurado adequadamente, relembrando que estas tarefas não são abordada neste documento.

Com o Eclipse e o servidor Apache Tomcat e o Axis2 instalados, já é possível iniciar a implementação de um *Web Service*. Porém, é interessante a instalação de um *plugin* para o Eclipse denominado "*Axis2 Service Archiver*". Este *plugin* visa facilitar a implantação (*deploy*) dos *Web Services* no Axis2, e o procedimento para a sua instalação está na próxima seção.

3.5 Axis2 Service Archiver

O Axis2 Service Archiver[1] é um *plugin* para o ambiente de desenvolvimento integrado Eclipse. Ele facilita o trabalho do desenvolvedor na geração de arquivos de serviços com extensão "*jar*" e "*aar*", que podem ser implantados facilmente no Axis2.

Os arquivos de extensões do tipo "AAR" (***Axis Archive***) foram elaborados pela Apache Software Foudation e nada mais são que arquivos compactados (semelhantes ao JAR - ***Java Archive***), os quais contêm estruturas definidas de diretórios e arquivos. Por exemplo, um ARR deve possuir o diretório "META-INF", contendo um arquivo denominado "*services.xml*", que contém informações a respeito do *Web Service*.

O Axis2 automaticamente implanta os serviços criados com extensão AAR. O usuário deve compactar o *Web Services* dentro deste tipo de arquivo e copiá-lo

para o subdiretório "*../WEB-INF/services/*", no caso do exemplo deste documento, o caminho seria o seguinte:

- `/home/luis/apache-tomcat-6.0.26/webapps/axis2/WEB-INF/services`

Se o servidor Tomcat estiver em execução juntamente com o Axis2, o serviço ficará disponível sem a necessidade de reiniciar o servidor. Por este motivo, a geração do arquivo AAR se torna interessante, e portanto é aconselhável ao usuário instalar o *plugin* "*Axis2 Service Archiver*". Para a sua instalação, é preciso realizar o *download* no seguinte endereço:

- `http://archive.apache.org/dist/ws/axis2/tools/1_4/axis2-eclipse-service-archiver-wizard-1.4.zip`

A instalação do *plugin* é feita por meio dos seguintes passos:

1. Caso o Eclipse esteja em execução, é recomendável que o usuário encerre esta aplicação;
2. Mover o arquivo axis2-eclipse-service-archiver-wizard-1.4.zip para o subdiretório "*plugins*" dentro do diretório do Eclipse: `mv axis2-eclipse-service-archiver-wizard-1.4.zip eclipse/plugins`;
3. Descompactar o arquivo: `unzip axis2-eclipse-service-archiver-wizard-1.4.zip`.

Neste ponto, todas as ferramentas que serão utilizadas foram instaladas e configuradas. No próximo capitulo, é explicado o passo a passo para a criação de um *Web Service* simples.

4 Desenvolvimento do Web Service

A configuração realizada no capítulo anterior deve ser bem sucedida para que os passos recomendados neste capítulo sejam eficazes. Além disso, admite-se que o sistema operacional, a Java Virtual Machine (JVM) e as variáveis de ambiente estejam previamente configuradas.

Caso o desenvolvedor tenha problemas durante a configuração, ele poderá procurar auxílio com o administrador de rede ou o responsável por essa finalidade. Para iniciar a configuração do Eclipse e iniciar a criação de um *Web Service* simples, é preciso seguir os passos:

1. Na página de boas vindas do Eclipse, deve-se clicar no botão "*Workbench*", que levará o desenvolvedor a uma perspectiva de ambiente de trabalho;
2. O segundo passo é configurar as preferências do Axis2 e do Servidor Apache no Eclipse. Para isso, o desenvolvedor deverá ir ao menu "*Windows*" e entrar na opção "*Preferences*". Uma janela igual a da Figura 7 ficará disponível para a configuração do Eclipse de acordo com as preferências do usuário. Ao lado esquerdo da janela, o usuário deverá entrar na pasta "*Web Services*" e, em seguida, no item "*Axis2 Preferences*", indicando o caminho do Axis2, como por exemplo: ***/home/luis/apache-tomcat-6.0.26/webapps***. Se o caminho estiver correto, a mensagem: "*Axis2 runtime loaded sucessfully*"

será exibida. Isto indica que o Axis2 foi configurado corretamente e, assim, o desenvolvedor pode aplicar ("*Apply*") esta configuração. Para finalizar a configuração das preferências, o desenvolvedor deverá voltar à pasta "*Web Services*" e entrar na opção "*Server and Runtime*" (Figura 8). Nesta nova opção, o usuário deverá selecionar o servidor "*Tomcat v6.0 Server*" e escolher a opção "*Apache Axis2*". Novamente é preciso aplicar (botão *Apply*) e confirmar as preferências (pressionando o botão "*OK*").

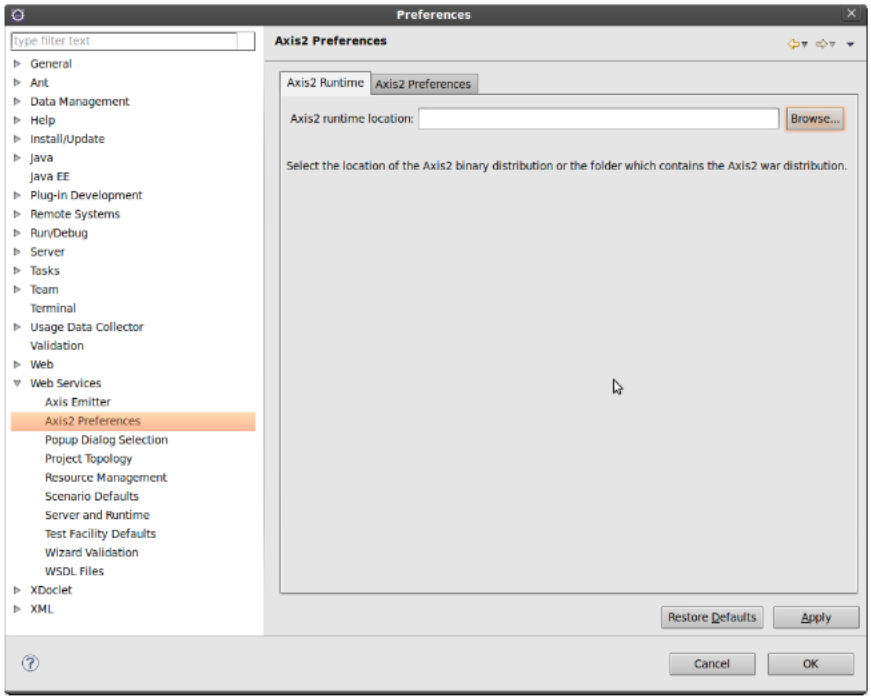

Figura 7. Configurar a preferência do Axis2 no Eclipse.

3. Para a construção do *Web Service*, é preciso inicialmente criar um projeto web dinâmico. Para isso, deve-se entrar no item de menu "*File*" > "*New*" > "*Dinamic Web Project*", onde será apresentada uma janela semelhante a da Figura 9.
4. Na opção "*Target runtime*", é preciso clicar sobre o botão "*New...*". Esta opção abre outra janela para a seleção de um servidor utilizado no ambiente de tempo de execução. O Eclipse disponibiliza uma série de opções de servidores que podem ser selecionados, como diversas versões do Apache Tomcat, o WebSphere da IBM, JBoss, etc. Nesta nova janela, é preciso expandir a pasta Apache e selecionar o servidor "*Apache Tomcat v6.0*", conforme exibe a Figura 10.

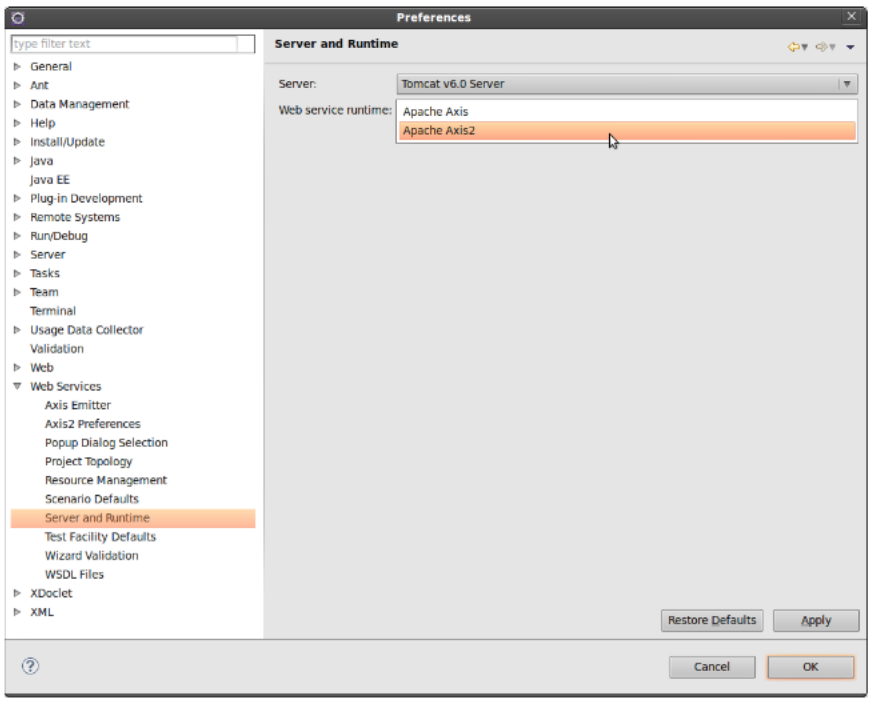

Figura 8. Configurar a preferência do Servidor no Eclipse.

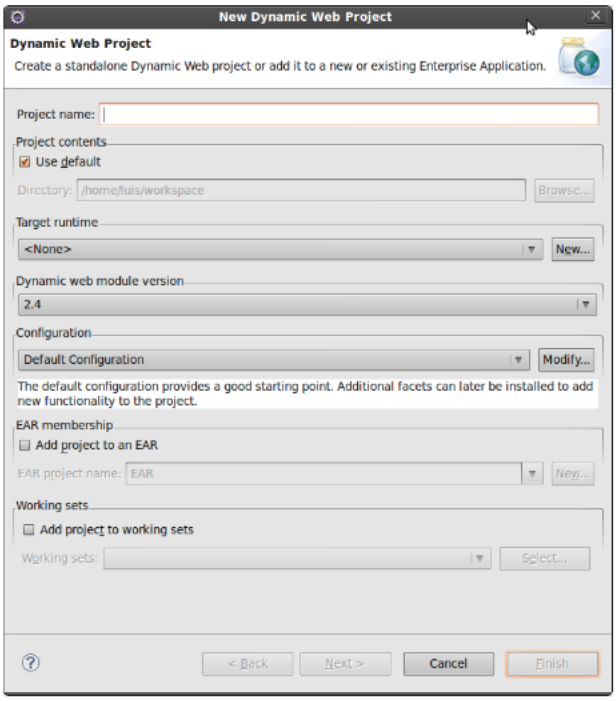

Figura 9. Criar um Projeto Web Dinâmico (*Web Services*).

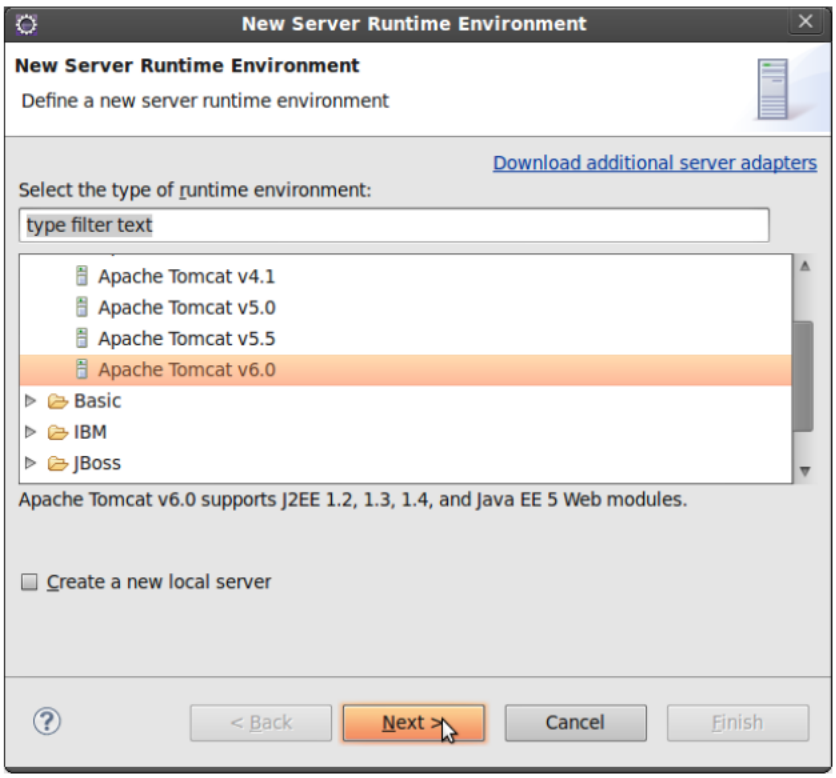

Figura 10. Selecionar o Servidor (*Target runtime*).

5. No quinto passo, o desenvolvedor deve pressionar o botão "*Next*". Uma janela igual à Figura 11 será exibida. Essa é a janela de criação do servidor Tomcat em tempo de execução. O desenvolvedor deverá especificar o local onde o servidor Apache Tomcat está instalado, como por exemplo: ***/home/luis/apache-tomcat-6.0.26*** e pressionar o botão "*Finish*" para finalizar a configuração. Para evitar conflitos de utilização de portas, a instância anterior que foi iniciada manualmente deve ser terminada. Para isso, pode-se abrir um terminal, entrar no diretório *bin* do Apache Tomcat e executar o *Shell Script* "*./shutdown.sh*".
6. Após o quinto passo, a janela da Figura 11 será fechada, e o desenvolvedor estará novamente na tela da Figura 9, porém com o servidor de tempo de execução (*runtime*) configurado. Dessa forma, o desenvolvedor terá o controle de iniciar e finalizar uma instância do servidor Tomcat pelo Eclipse. Além disso, neste sexto passo, ele deverá pressionar o botão "*Modify*" para alterar a configuração padrão ("*Default Configuration*") e utilizar o Axis2 (Figura 12). Em adição, ele deverá preencher o campo do nome do projeto (*Project name*). Neste exemplo, o nome do projeto é "***MyWebService***". Finalmente, o desenvolvedor pressiona o botão "*Finish*" para que o Eclipse crie os arquivos necessários para o projeto dentro do diretório que o usuário especificou (*workspace*).

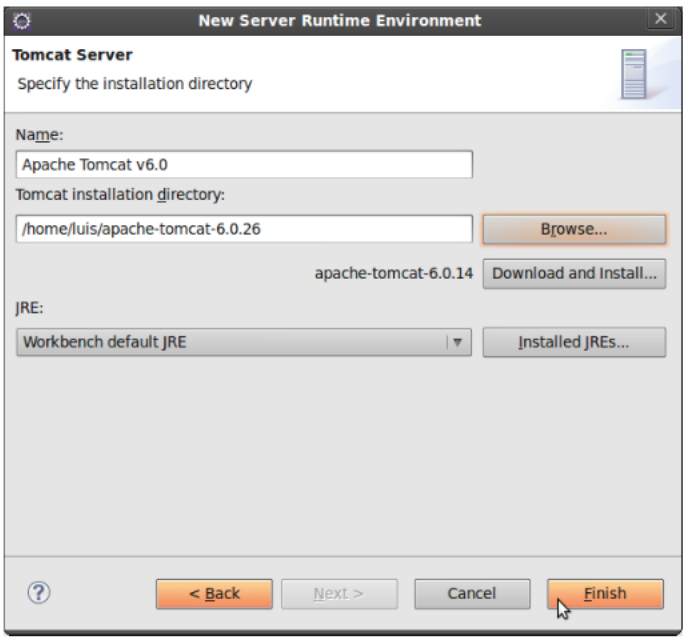

Figura 11. Informar o diretório do Tomcat e finalizar a configuração do Servidor no Eclipse.

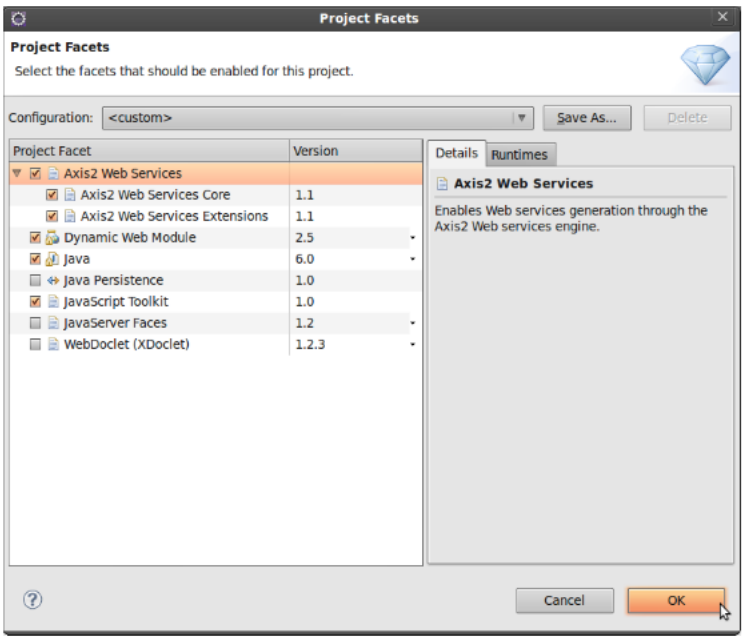

Figura 12. Alterar Configuração para utilizar *facets* do Axis2.

7. A "navegação" dentro do projeto é feita pela aba "*Project Explorer*", que fica geralmente ao lado esquerdo da interface de desenvolvimento (Figura 13).

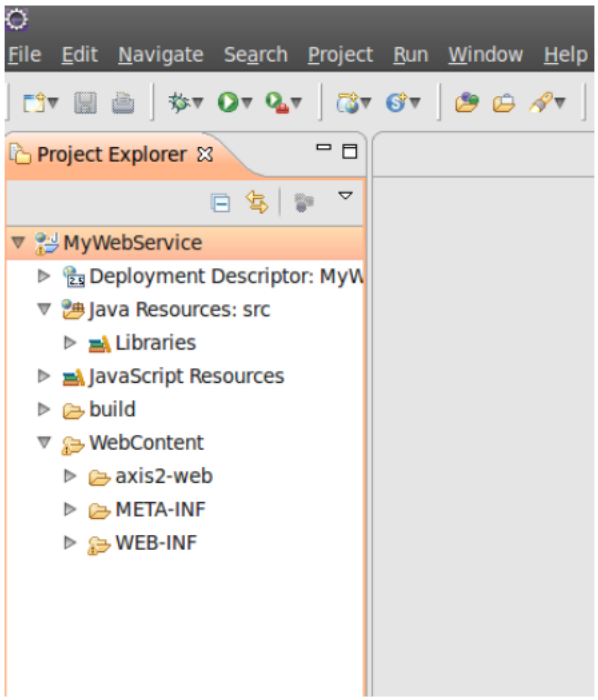

Figura 13. Aba *Project Explorer* do Eclipse.

8. Dentro da pasta "*Java Resources: src*", o desenvolvedor deverá criar uma classe em Java. Para isso, é preciso clicar sobre aquela pasta com o botão direito do mouse e ir na opção "*New*" > "*Class*". Uma janela de *pop-up* será aberta para o preenchimento do nome da classe, neste exemplo, "***MeuServico***", opcionalmente, é permitido colocar a classe dentro de um pacote (*Package*), como por exemplo: "*usp.br*". Nessa nova classe, haverá o método *HelloWorld*, que recebe e também retorna uma *String* como parâmetro. O código da classe está no ANEXO 1, e a estrutura do "*Project Explorer*" ficará próxima da exibida na Figura 14.
9. Após a criação da classe "***MeuServico***", é relativamente simples criar o *Web Service* com o Eclipse, pois há um *wizard* que facilita este trabalho. O desenvolvedor deverá clicar com o botão direito do mouse sobre a classe criada (*MeuServico*), ir na opção "*Web Services*", e então clicar em "*Create Web service*". Estes passos estão ilustrados na Figura 15.
10. A primeira tela do *wizard* é apresentada na Figura 16, e está dividida horizontalmente em três partes: a primeira para configuração do *Web Service*, a segunda para a configuração do Cliente que consumirá o *Web Service* e a última para a publicação e monitoração do *Web Service*. O *wizard* oferece ao usuário a opção de escolha de dois tipos de criação de *Web Services*:

- **Top down Java bean Web Service:** Neste tipo, o desenvolvedor deve especificar uma URI ou um documento WSDL, WSIL ou HTML para que o Eclipse gere os *stubs*,[10] posteriormente, ele deverá implentar a lógica do serviço para a publicação. Este tipo de criação não será abordado neste documento. Um tutorial de como criar *Web Services* dessa forma está disponível em [11].

- **Bottom up Java bean Web Service:** Neste tipo, o desenvolvedor fornecerá uma classe que será a implementação do serviço. Caso esta classe esteja dentro de um pacote, todo o caminho deverá ser especificado, como por exemplo: "*usp.br.MeuServico*". Este é o tipo de criação utilizado neste documento.

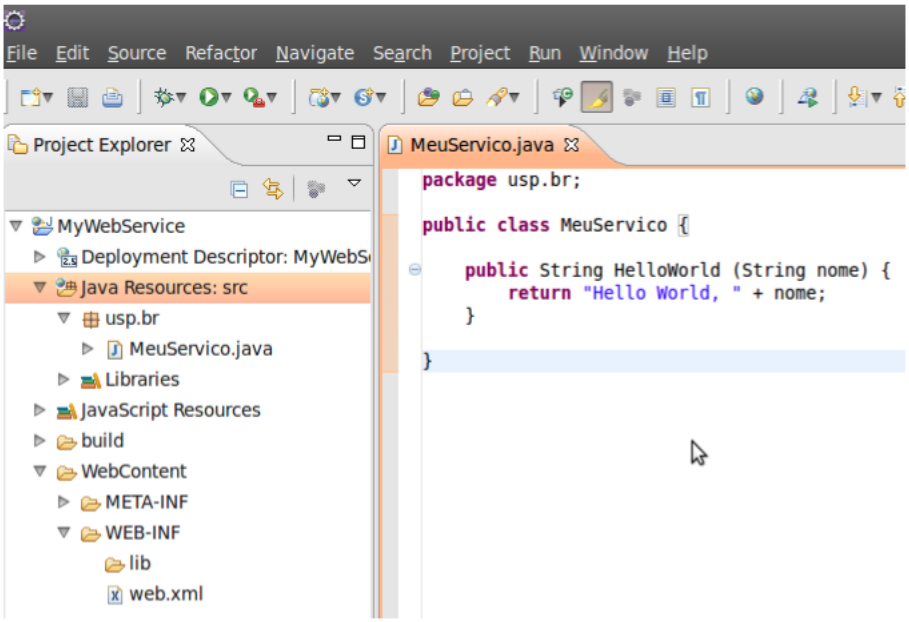

Figura 14. Estrutura do Projeto após inclusão da Classe "MeuServico".

[10] Os procedimentos *stubs* garantem a transparência para o cliente, que tem a impressão de que o procedimento é executado no local, quando na verdade ele é executado remotamente. O *stub* empacota a mensagem e seus parâmentros e a envia para ser executada no servidor [2].

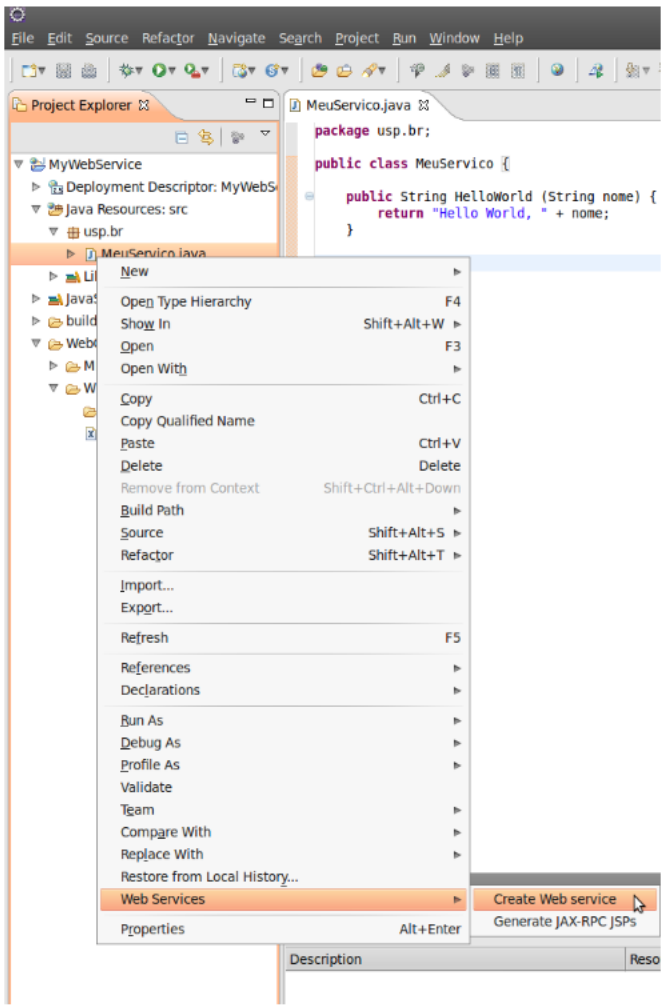

Figura 15. Criar um novo *Web Service*.

É possível observar, na Figura 16, que a configuração do *Web Service* já está de acordo com as preferências do desenvolvedor (utilizando as versões escolhidas anteriormente do servidor Tomcat e do Axis2). Na parte do Cliente, o desenvolvedor poderá optar pela criação de um cliente para consumir o serviço. Tanto na parte do *Web Service* quanto na parte do Cliente, o desenvolvedor poderá selecionar o nível de geração de código. No caso do Cliente, este nível pode variar desde "*Nenhum*' (cliente) até o (cliente) "*Testado*". Já no caso do *Web Service*, há o (serviço) "*Desenvolvido*" até o (serviço) "*Testado*". O Cliente sairá do nível "*Nenhum*" somente quando o *Web Service* estiver, no mínimo, no nível "*Iniciado*". As opções "*Publish the Web service*" e "*Monitor the Web service*" não serão utilizadas nesse documento. Apenas para critério de informação, a primeira trata-se da publicação do *Web Service*

em um registro UDDI (*Universal Description, Discovery and Integration*) e a segunda trata-se de uma monitoração do tráfego de mensagens SOAP.

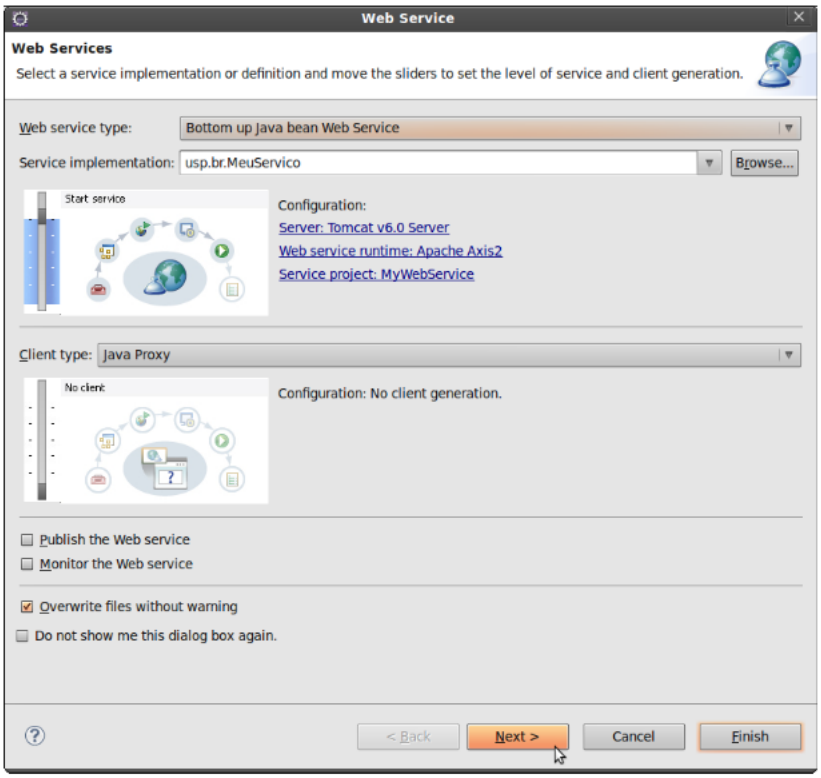

Figura 16. Primeira tela do *Wizard* para a criação do *Web Service*.

Para prosseguir no exemplo deste documento, o tipo de criação de serviço "*Botton up Java bean Web Service*" de ser escolhido. Neste momento, o cliente não será criado. Esta etapa será feita no próximo capítulo. Neste ponto, o nível do *Web Service* deve estar em "*Iniciado*" (*Start service*), o do Cliente em "*Nenhum*" (*No Client*), e as opções "*Publish the Web service*" e "*Monitor the Web service*" devem estar desabilitadas. Dessa forma, o desenvolvedor poderá pressionar o botão "*Next*" e continuar seguindo este documento.

11. A segunda tela do *wizard* é apresentada na Figura 17. Nesta etapa pode-se optar por especificar a localização de um arquivo "*services.xml*" existente, ou então permitir que o *wizard* crie um arquivo padrão. Este arquivo é utilizado para especificar informações (*java bean*) sobre o *Web Service* e será útil na criação da WSDL. Neste ponto, o desenvolvedor deverá manter a opção padrão e pressionar o botão em "*Next*".
12. Na terceira tela, o *wizard* requisita que o servidor seja iniciado (Figura 18). O desenvolvedor deve pressionar o botão "*Start server*" ("*Iniciar o servidor*") para iniciar o Apache Tomcat. Após o servidor iniciar, o botão "*Next*" ficará

disponível para a continuação do *wizard*. O desenvolvedor deve pressioná-lo para ir para a última tela (Figura 19).

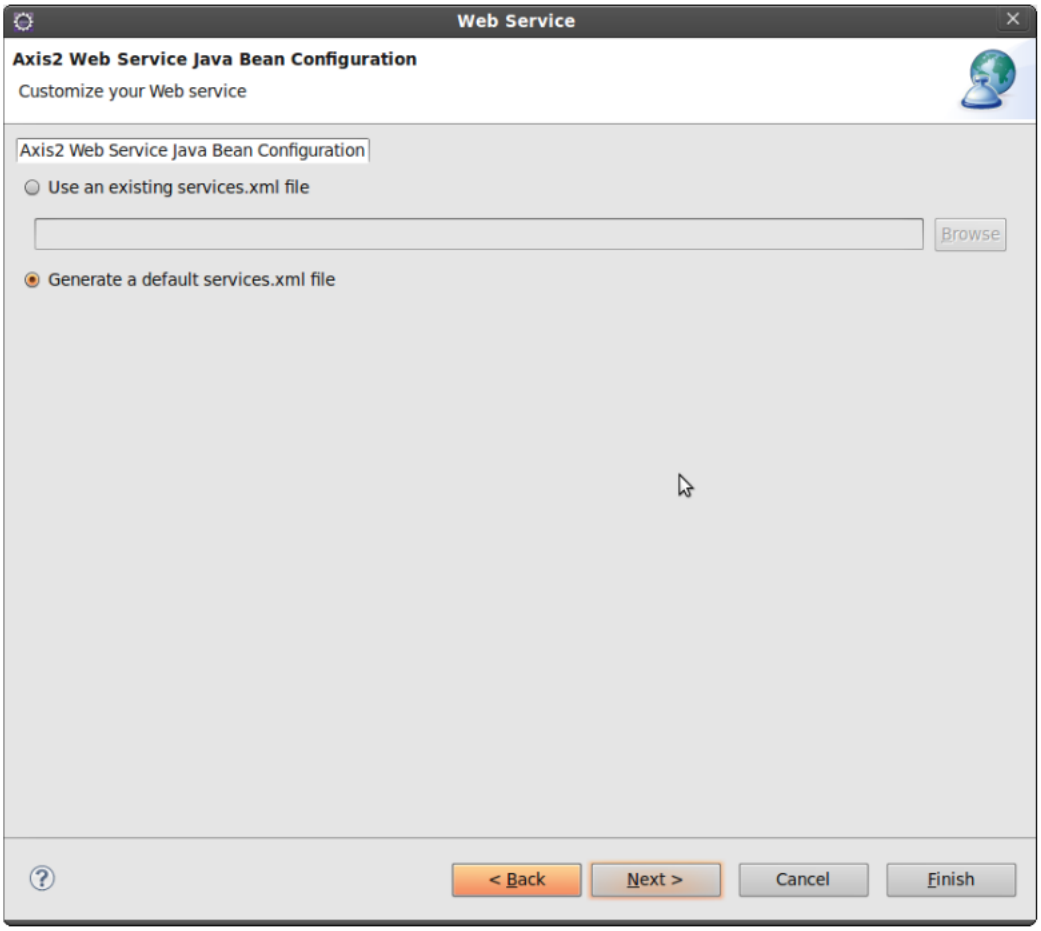

Figura 17. Segunda tela do *Wizard* para a criação do *Web Service*.

13. Na quarta e última tela do *wizard* (Figura 19), o desenvolvedor tem a opção de executar o *Web Service* para realizar publicação de sua WSDL no registro UDDI para teste unitários (Unit Test UDDI Registry) e/ou diretamente no registro UDDI final. Apesar da publicação e da execução dos testes unitários serem recomendadas, por motivo de não prolongar o tutorial, estes tópicos não serão abordados neste documento. O desenvolvedor poderá deixar as duas opções desmarcadas e pressionar o botão "*Finish*".

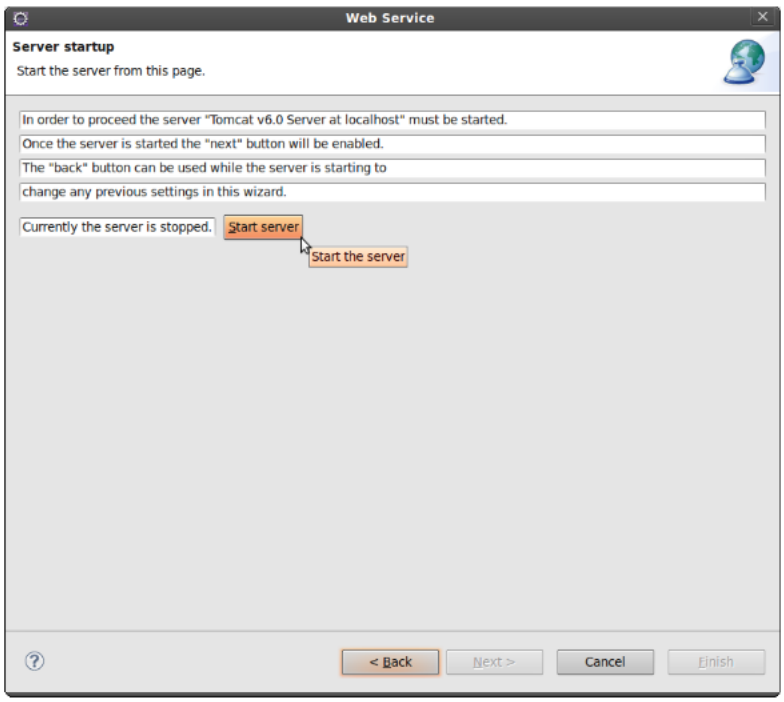

Figura 18. Iniciar o servidor Tomcat pelo Eclipse.

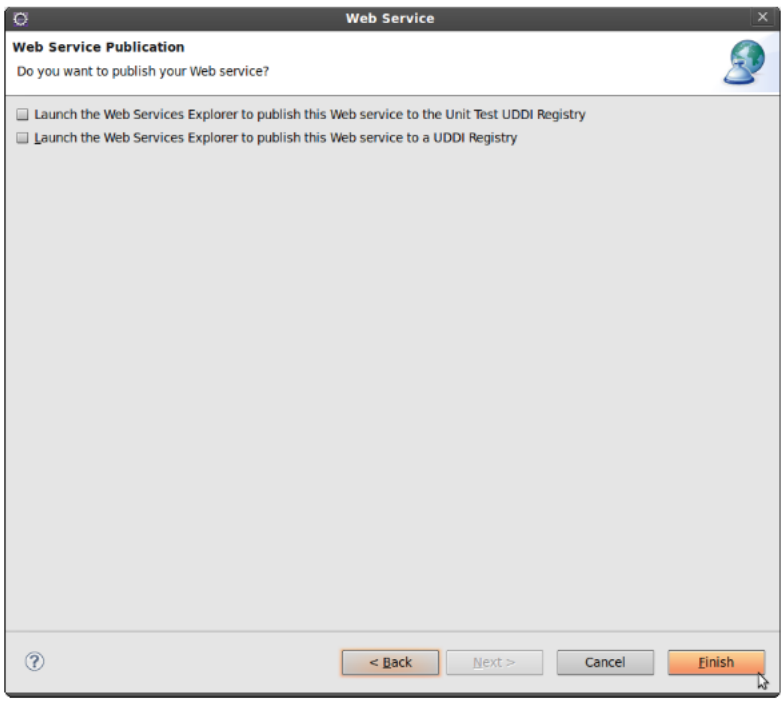

Figura 19. Finalizar o *wizard*.

Após o encerramento do *wizard*, o *Web Service* estará criado dentro do projeto em: "***WebContent/WEB-INF/services/MeuServico***". A Figura 20 destaca a estrutura e localização do *Web Service* dentro do Projeto.

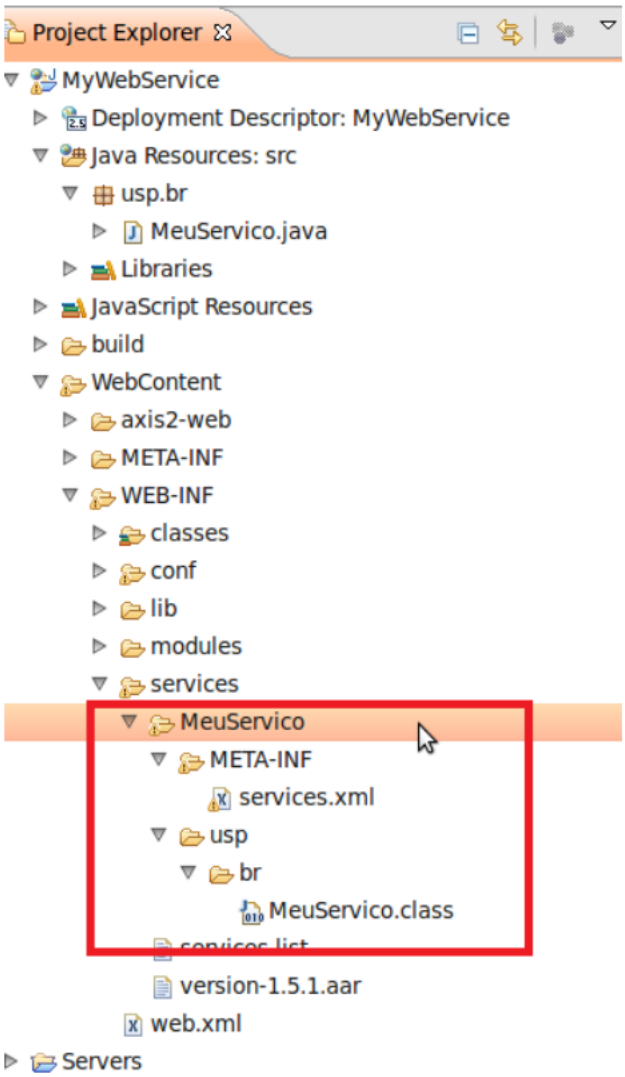

Figura 20. *Web Service* criado no Projeto.

Se o diretório "**MeuServico**" for copiado para dentro do diretório de serviços web do Axis2: ("*/home/luis/apache-tomcat-6.0.26/webapps/axis2/WEB-INF/services*") então o serviço "**MeuServico**" seria automaticamente publicado e estaria disponível para utilização. Porém, há uma forma mais fácil e elegante para a publicação de serviços web, que é por meio da geração de arquivos AAR.

Para criar estes arquivos, é utilizado o *plugin* "*Axis2 Service Archiver*", que fornece um outro *wizard* ao desenvolvedor.

Para abrir o *wizard* é preciso ir ao menu do Eclipse na opção "*File*" > "*New*" > "*Other...*"; uma janela de *pop-up* é apresentada conforme a Figura 21. Este *pop-up* lista diversos tipos de *wizards*, o desenvolvedor deverá escolher aquele da pasta "*Axis2 Wizards*" que tem a descrição "*Axis2 Service Archiver*", e então pressionar o botão "*Next*".

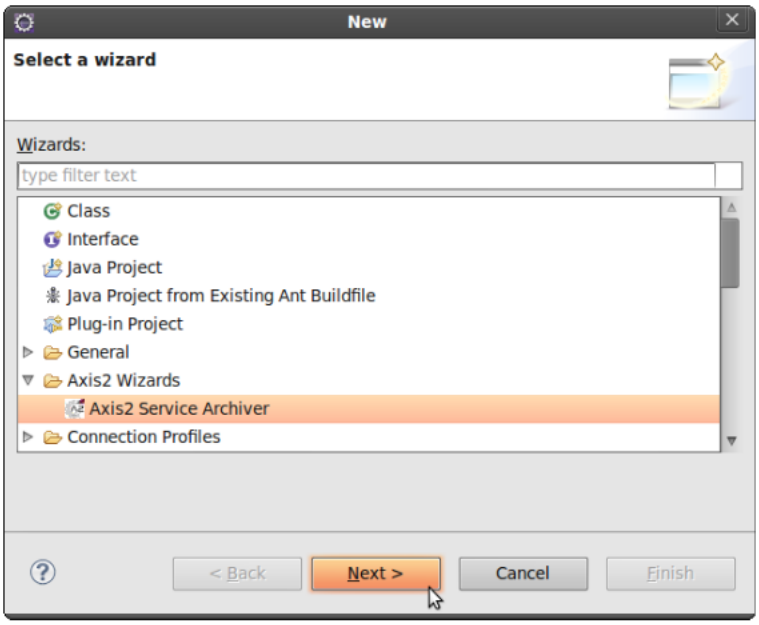

Figura 21. Abrir o *wizard* para criação do ARR.

O passo a passo da utilização do *wizard* de criação do arquivo ARR é descrito a seguir:

1. Uma tela igual à da Figura 22 é apresentada ao desenvolvedor, que deverá especificar a localização do *Web Service*. No caso do exemplo utilizado neste documento, o caminho seria "***/home/luis/workspace/MyWebService/WebContent/WEBINF/services/MeuServico/***". O desenvolvedor tem ainda a opção de incluir apenas os arquivos "*.class*", e então pode prosseguir com o *wizard* pressionando o botão "*Next*".
2. A Figura 23 exibe a terceira etapa do *wizard* de criação do arquivo ARR. Nela pode-se informar uma WSDL específica já existente para que seja adicionada junto ao arquivo AAR. Porém, no caso deste exemplo, não é preciso adicionar uma WSDL. Assim, a opção "*Skip WSDL*" ("pular a WSDL") deverá ser habilitada, e o desenvolvedor poderá passar para o próximo passo pressionando o botão "*Next*".

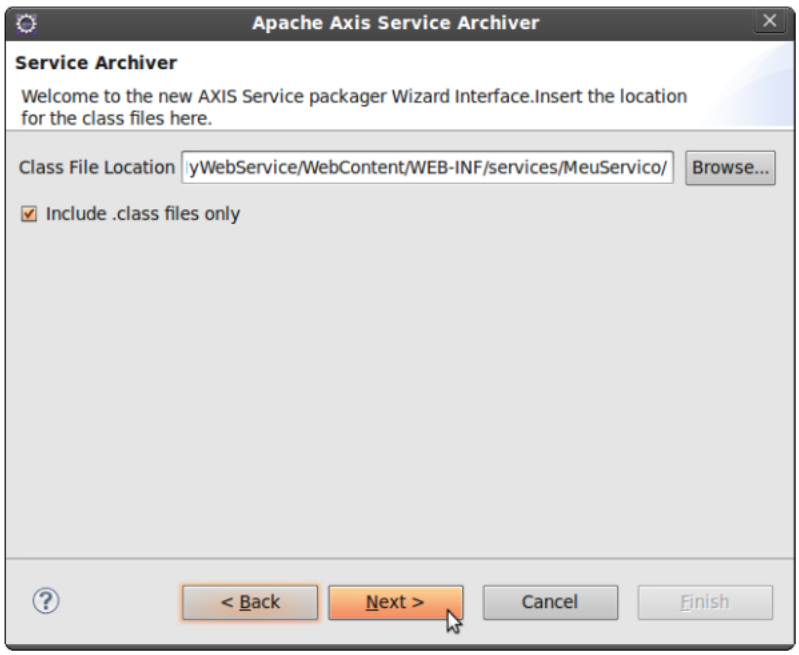

Figura 22. Informar a localização dos arquivos.

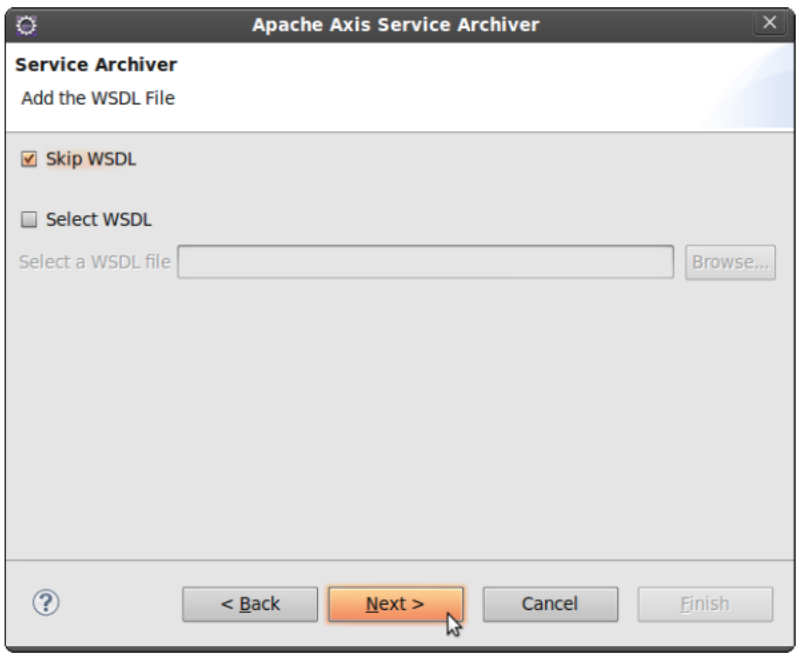

Figura 23. Não adicionar a WSDL ao arquivo AAR.

3. A terceira tela do *wizard* permite que o usuário selecione bibliotecas externas no arquivo compactado AAR. Este passo é interessante para serviços complexos, que dependam de bibliotecas externas. No caso do "*MeuServico*", que é um serviço simples, nenhum biblioteca externa foi utilizada, de modo que o desenvolvedor pode passar para a próxima etapa sem nenhuma alteração (Figura 24).

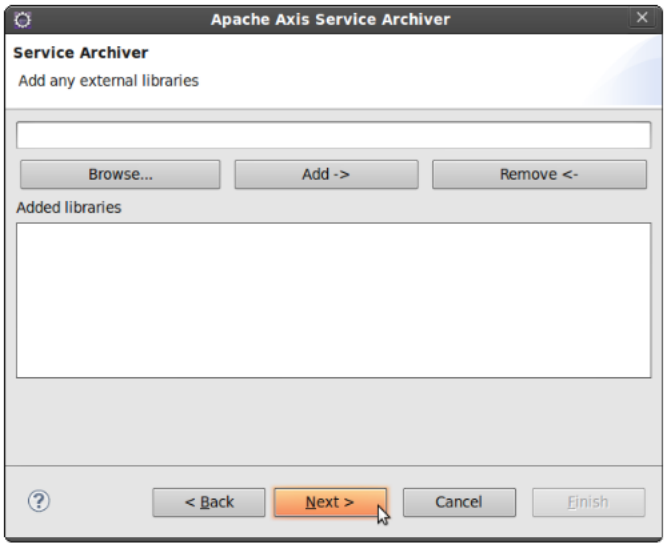

Figura 24. Nenhuma biblioteca externa é necessária para o serviço *MeuServico*.

4. O quarto passo do *wizard* é selecionar o arquivo XML que contenha a especificação do serviço. Este arquivo é nomeado como "*services.xml*". Conforme ilustra a Figura 25, o desenvolvedor pode optar para que o arquivo seja gerado automaticamente. Entretanto, é recomendável utilizar o arquivo que foi gerado pelo *wizard* do Eclipse durante a criação do *Web Service*. Este arquivo, no caso do exemplo utilizado, se encontra no diretório: ***/home/luis/workspace/MyWebService/WebContent/WEB-INF/services/MeuServico/META-INF/.***
5. O último passo da geração do arquivo ARR é a tela de configuração da saída do arquivo. A Figura 26 exibe os dois campos que o desenvolvedor deve preencher:
   - ***Output file location:*** É a localização (o diretório) onde o arquivo AAR deve ser depositado. Neste exemplo, ele é depositado dentro do Axis2 (que está dentro do Apache Tomcat), no diretório: ***/home/luis/apache-tomcat-6.0.26/webapps/axis2/WEB-INF/services***. Dessa forma, se o servidor estiver executando, o arquivo ARR criado pelo *wizard* será automaticamente processado, e o serviço "*MeuServico*" será disponibilizado;
   - ***Output file name:*** É o nome que o arquivo deve receber. No exemplo, o nome dado foi "*MeuServico.aar*".

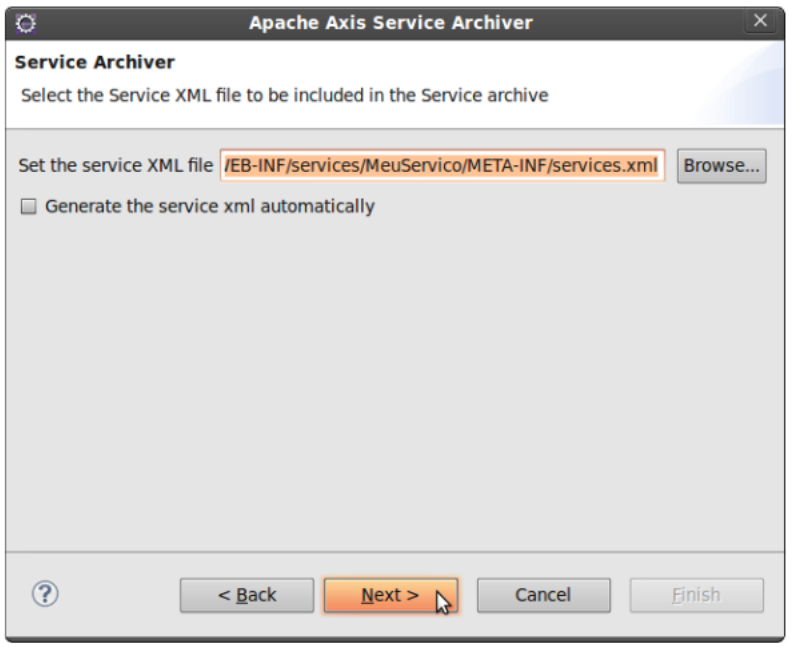

Figura 25. Informando o arquivo "*services.xml*" para o *wizard*.

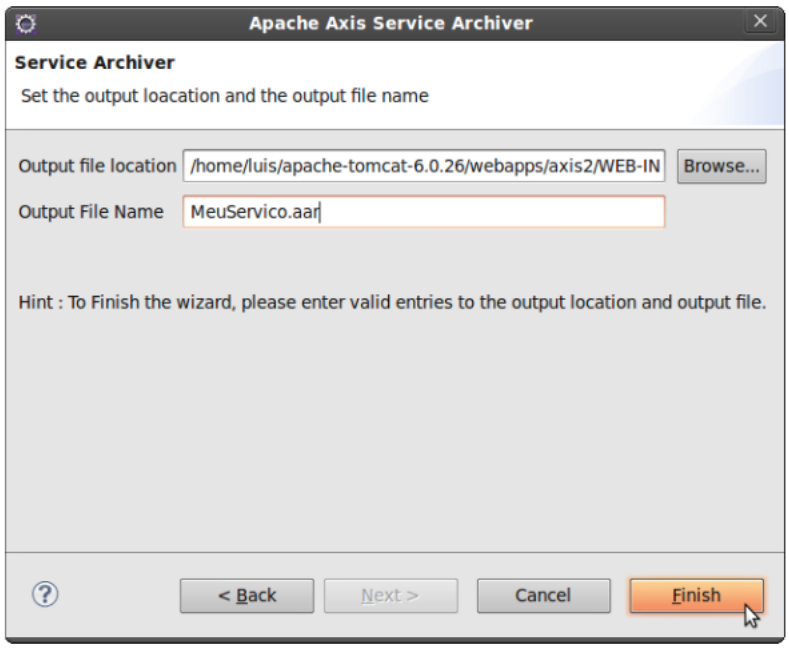

Figura 26. Configurar o local de criação e nome do arquivo AAR.

Uma vez que o arquivo foi depositado no local correto, ele deverá ser automaticamente carregado pelo Axis. Porém, para verificar se ele está realmente disponível, o desenvolvedor precisa acessar a URL: ***http://localhost:8080/axis2/services/listServices***. Neste ponto, talvez seja preciso iniciar o Apache Tomcat (seção 3.3). No endereço da URL, uma lista de serviços será exibida conforme mostra a Figura 27. Cada descrição de serviço é também um *link* que leva a sua WSDL. Se os passos foram seguidos corretamente, o serviço com a descrição "*MeuServico*" estará presente. Há outras formas de publicar o serviço além de copiá-lo ou criá-lo diretamente no diretório "*services*" do Axis2; pode-se, por exemplo, pela página de administração do Axis2 realizar o *upload* do arquivo AAR. Tal procedimento é muito utilizado por administradores de provedores de serviços que recebem os *Web Services* desenvolvidos (testados e validados) para serem implantados em servidores remotos.

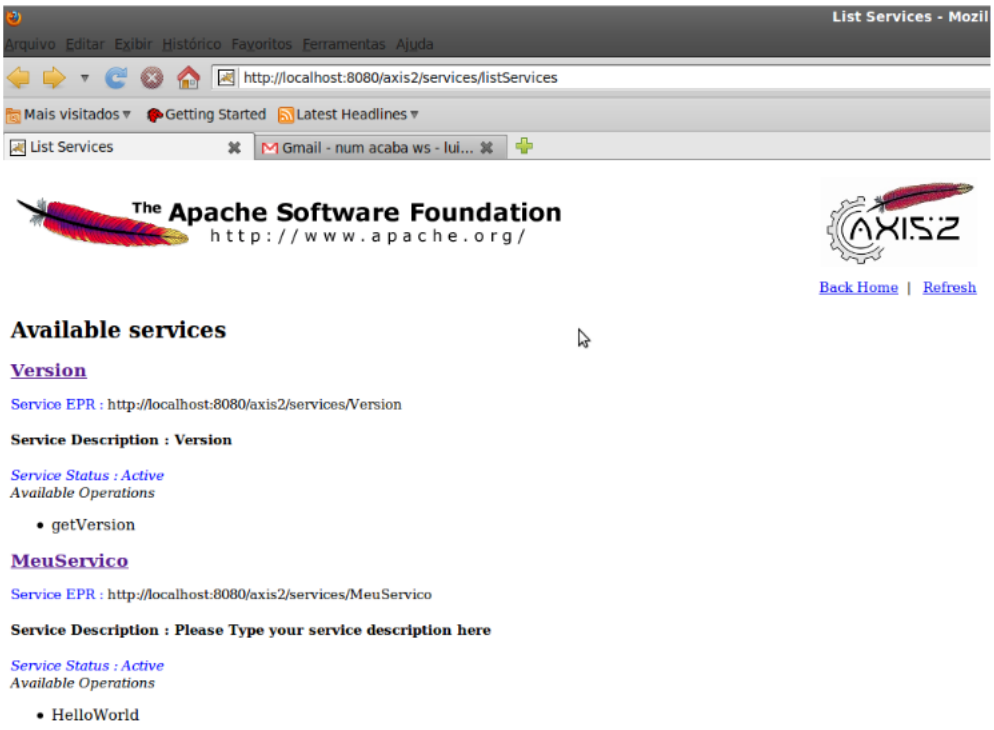

Figura 27. Lista de Serviços publicados no Axis2.

Neste capítulo, foi apresentado o passo a passo do desenvolvimento de um *Web Service* simples em Java com Eclipse. Foi possível observar que a criação de *Web Services* torna-se mais fácil e produtiva quando se utiliza o Eclipse. De acordo com o exemplo apresentado, dois *wizards* foram utilizados. Um deles

para a criação do *Web Service* e outro para o seu "*empacotamento*" em um arquivo de extensão "*.aar*". Além disso, foram abordadas algumas formas de publicação/implantação de *Web Services* no Axis2. Apesar da opção da criação do Cliente (consumidor do *Web Service*) ter sido comentada durante um dos passos da criação do *Web Service*, esta possibilidade foi propositalmente rejeitada para que possa ser explicada isoladamente, a fim de evitar confusão durante a criação do *Web Service*, e assim facilitar a didática. O próximo capítulo aborda a criação de um consumidor de serviço *web*, ou seja, da aplicação cliente.

5 Desenvolvimento do Cliente

O desenvolvimento do cliente, para consumir o *Web Service* desenvolvido no capítulo anterior, é bem parecido com o procedimento adotado para a própria criação do serviço. Neste capítulo, é apresentado o passo a passo para a criação do cliente. Contudo, os passos semelhantes aos da criação do serviço não serão explicados novamente. O desenvolvedor deve seguir os seguintes passos para a criação do aplicativo cliente:

1. Criar um novo projeto, conforme foi mencionado na criação do *Web Service*, repetindo os passos 3 a 6, porém com duas diferenças. A primeira é que o nome do projeto deverá ser diferente. Para o exemplo, foi utilizado o nome "***MyWebServiceClient***". A segunda diferença é que, no caso do cliente, o servidor Apache deve estar sendo executado. Caso não esteja, o desenvolvedor deve executar o *Shell Script* "***startup.sh***". (Obs: caso a instância do servidor Tomcat configurado no Eclipse esteja em execução, haverá conflito entre eles. A instância do Eclipse deve ser encerrada e, para isso, deve-se acessar a aba "*Servers*", clicar com o botão direito do mouse sobre o servidor e selecionar a opção "***Stop***").
2. A aba "Project Explorer" apresentará uma estrutura semelhante à da Figura 28. Uma vez que o projeto foi criado, será preciso criar os *stubs* do cliente. Para realizar esta tarefa, o Eclipse possui o *wizard* de criação do cliente, mas é preciso conhecer a URI ou a localização do arquivo com a WSDL do *Web Service*. Para iniciar o *wizard*, será preciso clicar com o botão direito sobre o projeto "***MyWebServiceClient***", ir na opção "*New*" e depois na opção "*Other...*" conforme mostra a Figura 29.
3. Uma janela igual à da Figura 30 é apresentada, e o desenvolvedor deve selecionar a pasta "*Web Services*", em seguida selecionar o item "*Web Service Client*" e pressionar o botão "*Next*".
4. Nesta etapa, a janela do *wizard* será aberta, de acordo com a Figura 31. O desenvolvedor precisará informar o endereço URI ou caminho do arquivo da WSDL e, para isso, ele deverá pressionar o botão "*Browse*".
5. Uma nova janela (Figura 32) de *pop-up* será apresentada, para que o desenvolvedor informe a WSDL. Para este exemplo, o endereço da WSDL será ***http://localhost:8080/axis2/services/MeuServico?WSDL***. Para obter o endereço da WSDL, o desenvolvedor precisará acessar a lista de serviços: ***http:/localhost:8080/axis2/services/listServices*** e clicar sobre a descrição (*link*) do serviço desejado. Assim, a WSDL é exibida e o desenvolvedor pode copiar o endereço do *browser*. Por fim, é preciso pressionar o botão "*OK*".

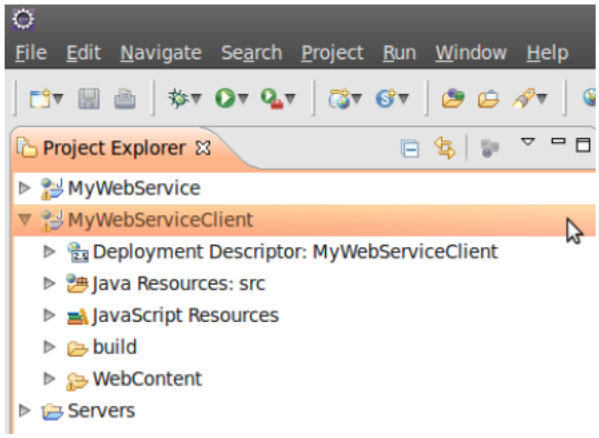

Figura 28. Aba Project Explorer com o novo projeto web dinâmico para o Cliente.

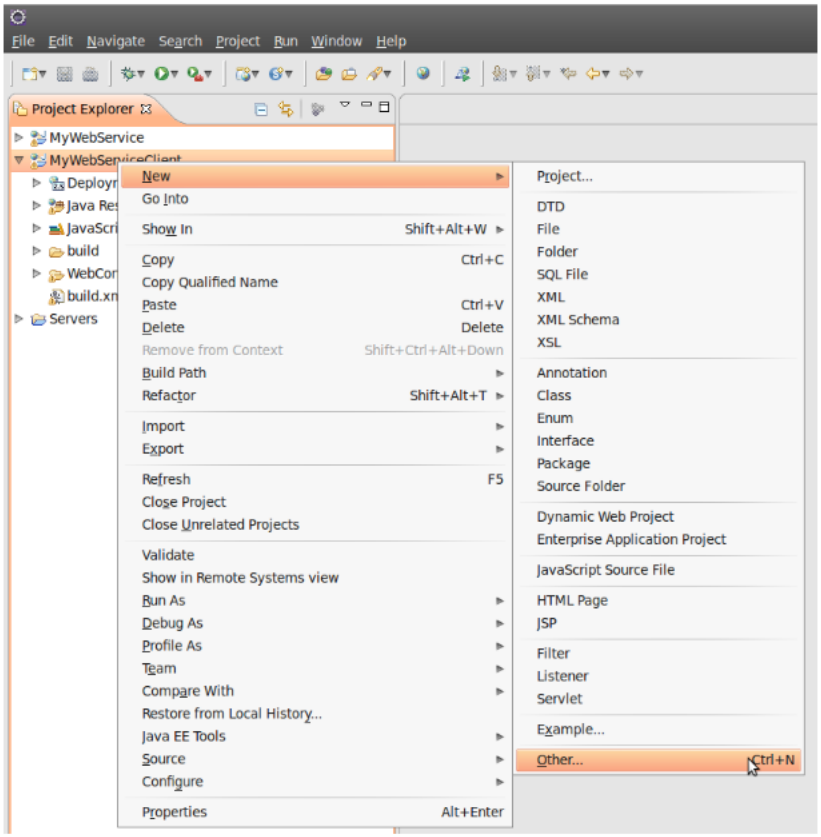

Figura 29. Abrir a janela de *wizard*.

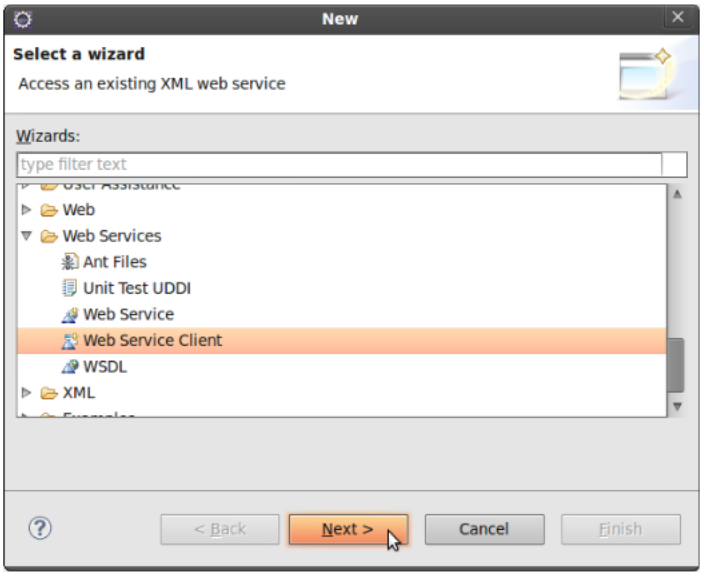

Figura 30. Janela para seleção do *wizard* com o *Web Service Client* selecionado.

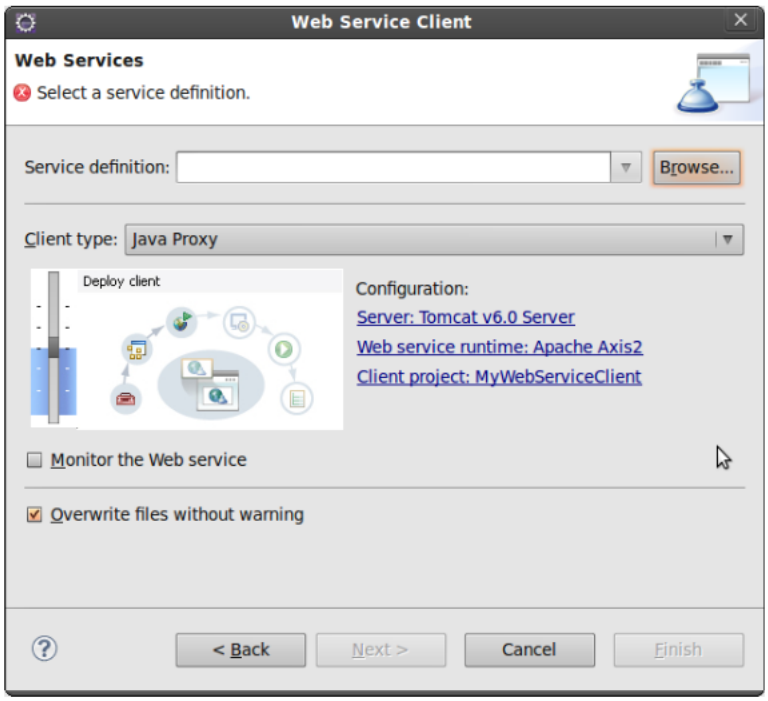

Figura 31. *Wizard* para criação do Cliente.

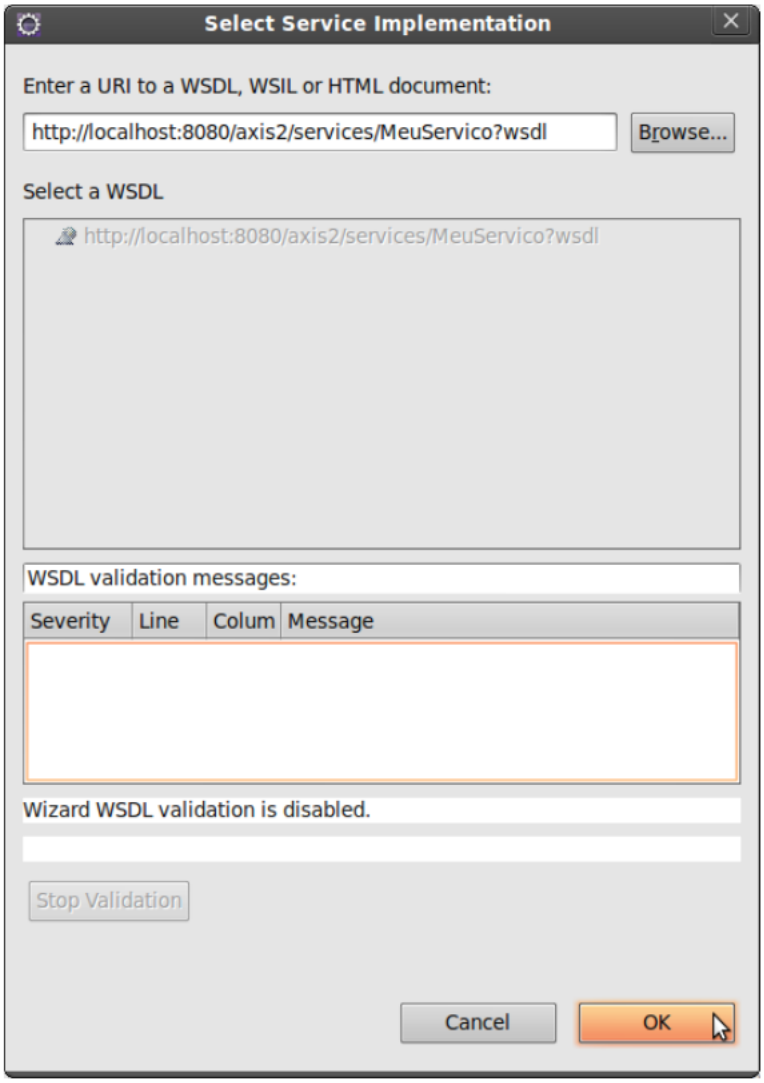

Figura 32. Informar a WSDL.

6. Após informar a WSDL, o *wizard* estará configurado e pronto para ser finalizado. O botão "*Finish*' deverá ser pressionado, como exibe a Figura 33.

Uma vez que o *wizard* for finalizado, ele criará os arquivos de *stubs* dentro do projeto do cliente. A Figura 34 exibe os dois arquivos de *stubs* que foram criados pelo *wizard*. Os arquivos *stubs* foram criados pelo *wizard* com auxílio do Axis2. O *wizard* já se encarregou de adicionar as bibliotecas dependentes. O desenvolvedor deve apenas adicionar o código com a lógica do cliente para conseguir invocar a operação/método desejada no *Web Service* utilizando as classes dos arquivos *stubs*. Para adicionar o cliente, deve-se clicar com o botão direito do mouse sobre qualquer parte do projeto do cliente, ir à opção "*New*" e depois em "*Class*".

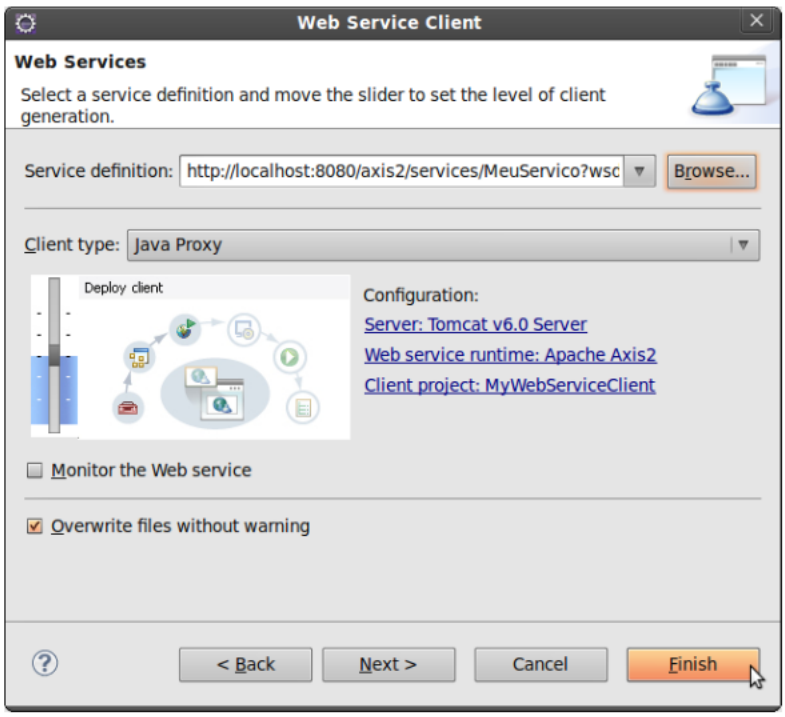

Figura 33. Finalizar o *wizard* do Cliente.

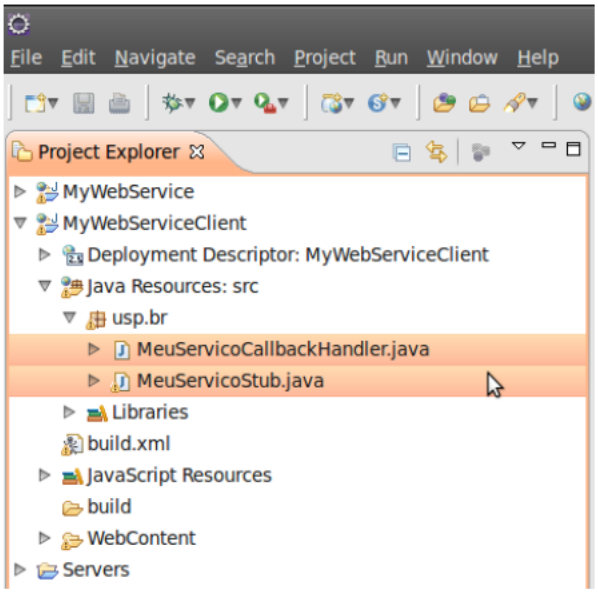

Figura 34. Arquivos *stubs* criados pelo *wizard*.

A janela da Figura 35 é aberta, para que o desenvolvedor consiga:

- ◇ Informar, opcionalmente, o pacote (*package*) da nova classe, que neste exemplo é "*usp.br*";
- ◇ Informar o nome da classe, neste caso "*Cliente*";
- ◇ Marcar a opção "*public static void main(String[] args)*", que indica que o método principal será público e estático, permitindo que a classe seja executada.

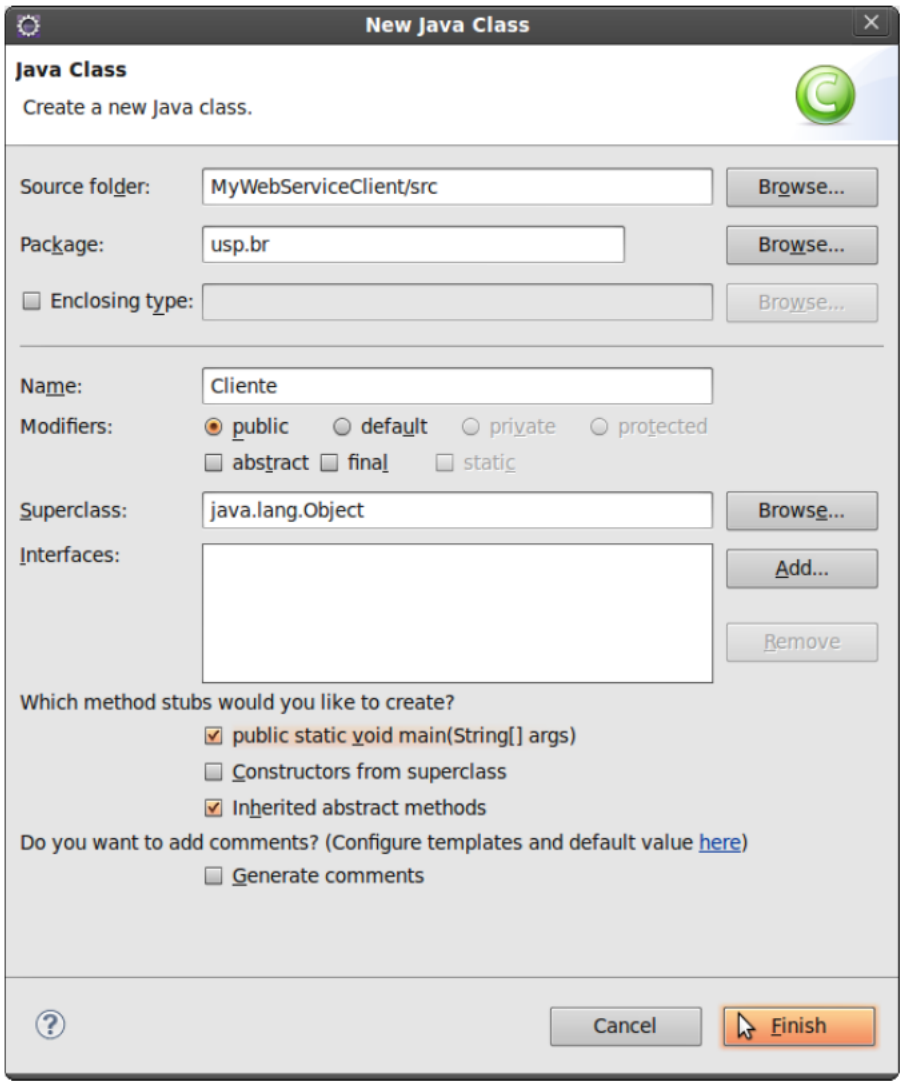

Figura 35. Criar a classe Cliente.

Com todos estes dados preenchidos, o botão "*Finish*" deve ser pressionado, e esta ação vai gerar um novo arquivo ("*Cliente.java*"), que contém a nova classe. O desenvolvedor poderá, então, programar a lógica do cliente e utilizar as classes *stubs* criadas pelo *wizard*. O ANEXO 2 contém o código de implementação do cliente, que invoca a operação/método "*HelloWorld()*" do *Web Service* utilizando as classes *stubs*.

Finalizada a implentação do Cliente será possível realizar um teste. Para isso, deve-se clicar com o botão direito do mouse sobre o arquivo "*Cliente.java*", escolher a opção "*Run As*" e selecionar "*Java Application*". O servidor Tomcat tem que estar em execução, e o resultado que aparecerá na aba "*Console*" será: "**Resposta : Hello World, ICMC**".

Uma vez que o Cliente passou no teste corretamente, é interessante que ele seja transformado em um arquivo JAR executável. Para criar um JAR executável, é preciso exportar o projeto para este tipo de arquivo. O primeiro passo é clicar com o botão direito do mouse sobre o pacote ("*usp.br*") e escolher a opção "*Export...*" (Figura 36).

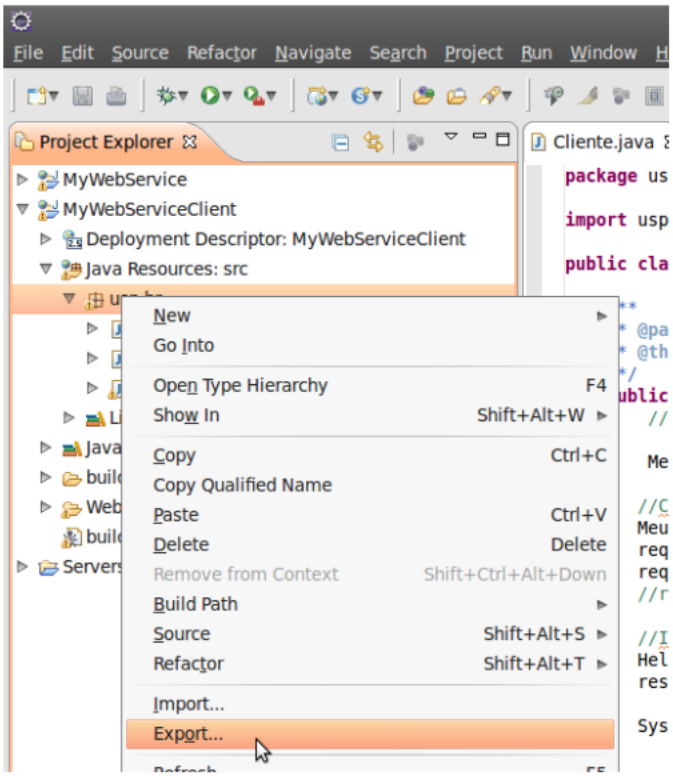

Figura 36. Iniciar a criação de um JAR.

Uma janela igual à Figura 37 é apresentada. O desenvolvedor deve escolher a pasta "*Java*", selecionar o item "*Runnable JAR file*" e pressionar o botão "*Next*".

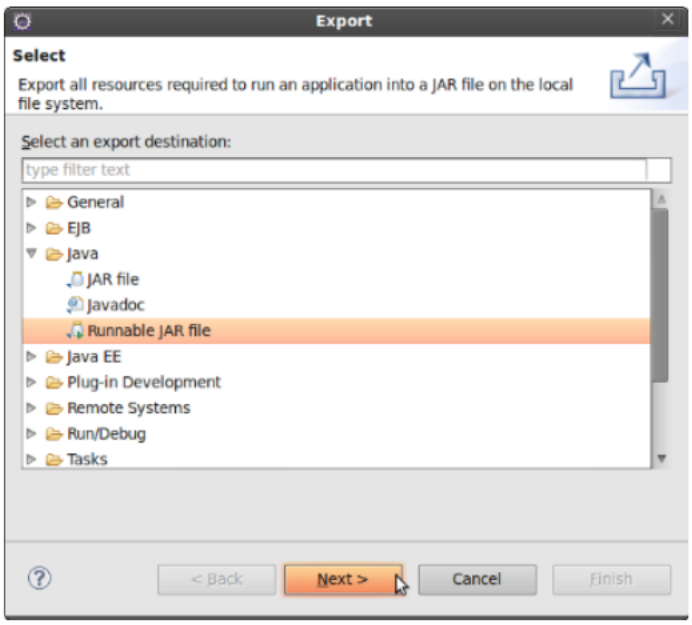

Figura 37. Selecionar o item JAR executável.

O último passo da criação do arquivo JAR será a apresentação de uma janela igual à da Figura 38.

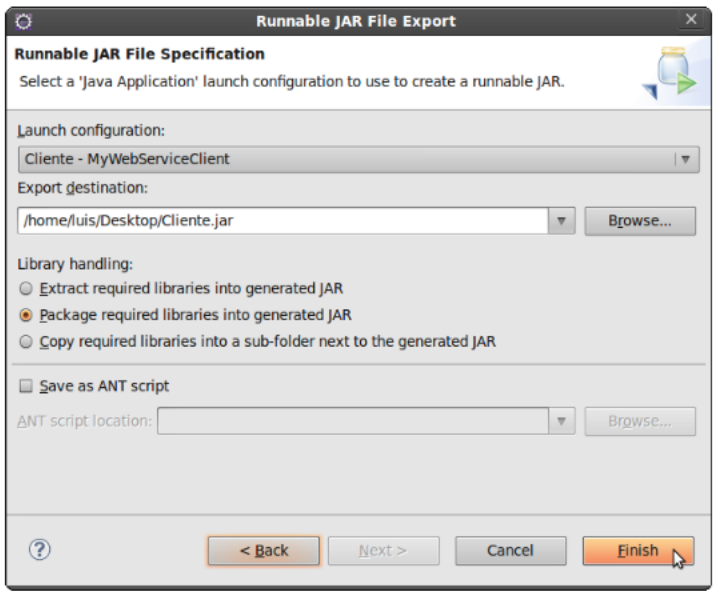

Figura 38. Configurar e finalizar a criação do JAR.

Na janela de criação do arquivo JAR, é posspivel realizar as seguintes ações:

- Definir qual a configuração de lançamento que, no caso do exemplo, é o Cliente do projeto "***MyWebServiceClient***";
- O diretório de destino do arquivo JAR, especificando o nome do arquivo, para este caso: "*/luis/home/Desktip/Cliente.jar*";
- Deverá ser escolhida a opção "***Package required libraries into generated JAR***", para que os pacotes de bibliotecas dependentes sejam criados dentro do arquivo JAR, eliminando assim as dependências no momento da execução;
- Por fim, o desenvolvedor deverá pressionar o botão "*Finish*", e então o arquivo JAR será criado no caminho que foi especificado.

Para testar o arquivo JAR criado anteriormente, deve-se primeiramente verificar se o Apache Tomcat está em execução; em seguinda, é preciso acessar o diretório onde o arquivo JAR foi criado, e então executar o comando:

- `java -jar Cliente.jar`

O resultado será o seguinte: "***Resposta : Hello World, ICMC***", conforme ilustrado na Figura 39.

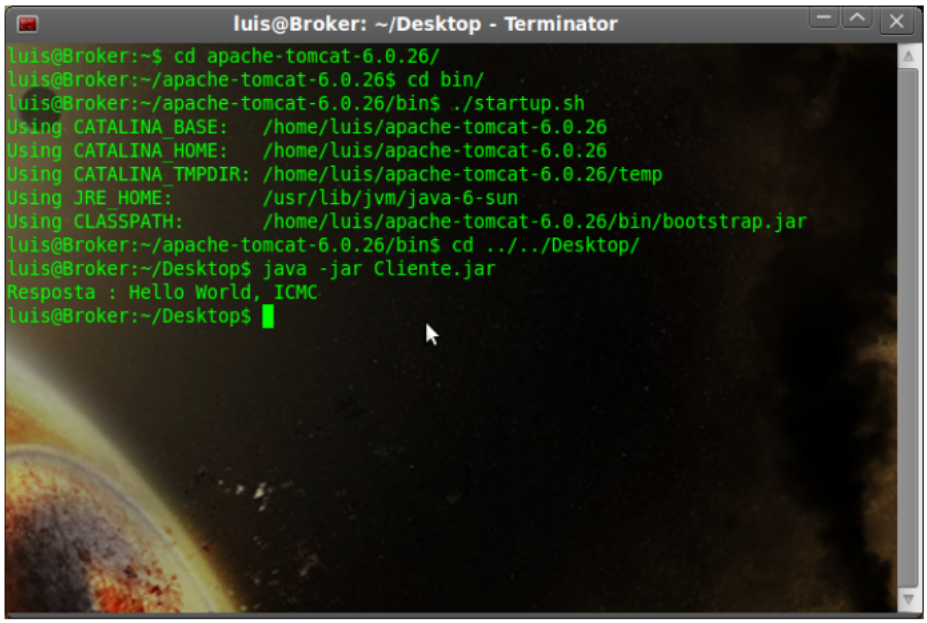

Figura 39. Resultado da Execução do JAR.

6 Conclusões

A crescente utilização de *Web Services* demanda que o seu desenvolvimento seja realizado de forma produtiva. Um modo de agilizar o desenvolvimento de *Web Services* é por meio da utilização de ambientes integrados de desenvolvimento (IDE). Este relatório em forma de tutorial procurou exemplificar sistematicamente os passos necessários para criação de um *Web Service* simples em Java, utilizando Eclipse e WTP. Além disso, este relatório apresentou o desenvolvimento de um cliente em Java utilizado para consumir o *Web Service* criado. Foram apresentadas ferramentas e programas recentes, que estão disponíveis gratuitamente na Internet, possibilitando o aprendizado básico de desenvolvimento de *Web Services*.

A abordagem adotada neste documento é apenas uma amostra entre diversas outras formas de desenvolvimento de *Web Services* existentes. Outras formas de desenvolvimento com outras IDEs e linguagens de programação podem ser discutidas em futuros relatórios. Um exemplo é a criação de *Web Services* com o Visual Studio (ambiente de desenvolvimento da Microsoft), utilizando-se a linguagem de programação C#.

ANEXO 1 - “MeuServico”

Este anexo exibe o código fonte utilizado para a criação do *Web Service* “MeuServiço”, que consta na Listagem 2.

Listagem 2 Arquivo MeuServico.java

```
package usp.br;

public class MeuServico {

  public String HelloWorld (String nome) {
    return "Hello World, " + nome;
  }

}
```

ANEXO 2 - “Cliente”

Este anexo exibe, na Listagem 3, o código fonte utilizado para a criação do Cliente do *Web Service* “MeuServiço”.

Listagem 3 Arquivo Cliente.java

```
package usp.br;

import usp.br.MeuServicoStub.HelloWorldResponse;

public class Cliente {

  /**
  * @param args
  * @throws Exception
  */
  public static void main(String[] args) throws Exception {

    MeuServicoStub stub = new MeuServicoStub();

    //Cria a requisicao para o servico
    MeuServicoStub.HelloWorld request;
    request = new MeuServicoStub.HelloWorld();
    request.setNome("ICMC");
    //request.setParam0("ICMC");

    //Invoca o servico
    HelloWorldResponse response;
    response = stub.HelloWorld(request);

    System.out.println("Resposta : " + response.get_return());

  }

}
```

Referências

1. Apache. Service archive generator wizard guide for eclipse plug-in, 2011. Disponível em: `http://axis.apache.org/axis2/java/core/tools/eclipse/servicearchiver-plugin.html`. Último acesso: 01/05/2011.
2. Coulouris, J. Dollimore, and T. Kindberg. *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.
3. Eclipse. About the eclipse foundation, 2011. Disponível em: `http://www.eclipse.org/org/#about`. Último acesso: 01/05/2011.
4. Eclipse. Eclipse web tools platform project, 2011. Disponível em: `http://www.eclipse.org/projects/project_summary.php?projectid=webtools`. Último acesso: 01/05/2011.
5. A. Erradi and O. Maheshwari. A broker-based approach for improving web services reliability. In *IEEE International Conference on Web Services (ICWS'05)*, 2005.
6. J. C. Estrella, R. H. C. Santana, and M. J. Santana. Web services tutorial, 2009. (Desenvolvimento de material didático) - Notas Didáticas do ICMC-USP). Disponivel em: http://www.icmc.usp.br/ biblio/BIBLIOTECA/not_did/ND_77.pdf.
7. P. Farkas and H. Charaf. Web services planning concepts. *Journal Of .net Technologies*, pages 9–12, 2003.
8. S. Lee and D. Shin. Web service qos in multi-domain. In *Proceedings of Advanced Communication Technology, - ICACT*, 2008.
9. L. H. V. Nakamura. *Utilização de Web Semântica para a seleção de informações de Web Services no Registro UDDI Uma Abordagem com Qualidade de Serviço.* Qualificação de mestrado, Universidade de São Paulo, São Carlos, SP, 2010.
10. M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. *Communications of the ACM. October 2003/Vol. 46, No. 10*, 2003.
11. L. Sandakith. Eclipse wtp tutorials - creating top down web service via apache axis2, 2011. Disponível em: `http://www.eclipse.org/webtools/community/tutorials/TopDownAxis2WebService/td_tutorial.html`. Último acesso: 01/05/2011.
12. J. P. Thomas and M. G. G. Thomas. Modeling of web services flow. In *Proceedings of the IEEE International Conference on E-Commerce (CEC'03)*. IEEE Computer Society, 2003.
13. Tomcat. Apache tomcat, 2011. Disponível em: `http://tomcat.apache.org`. Último acesso: 01/05/2011.