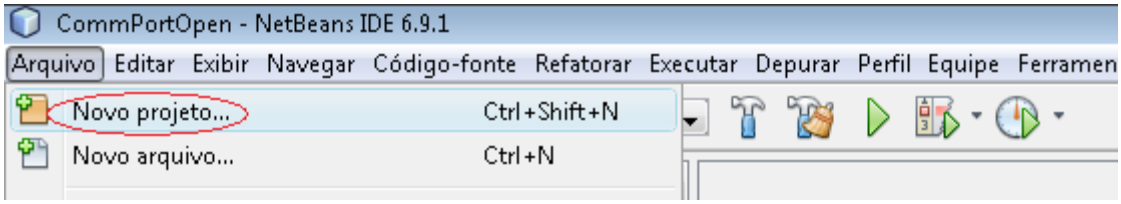
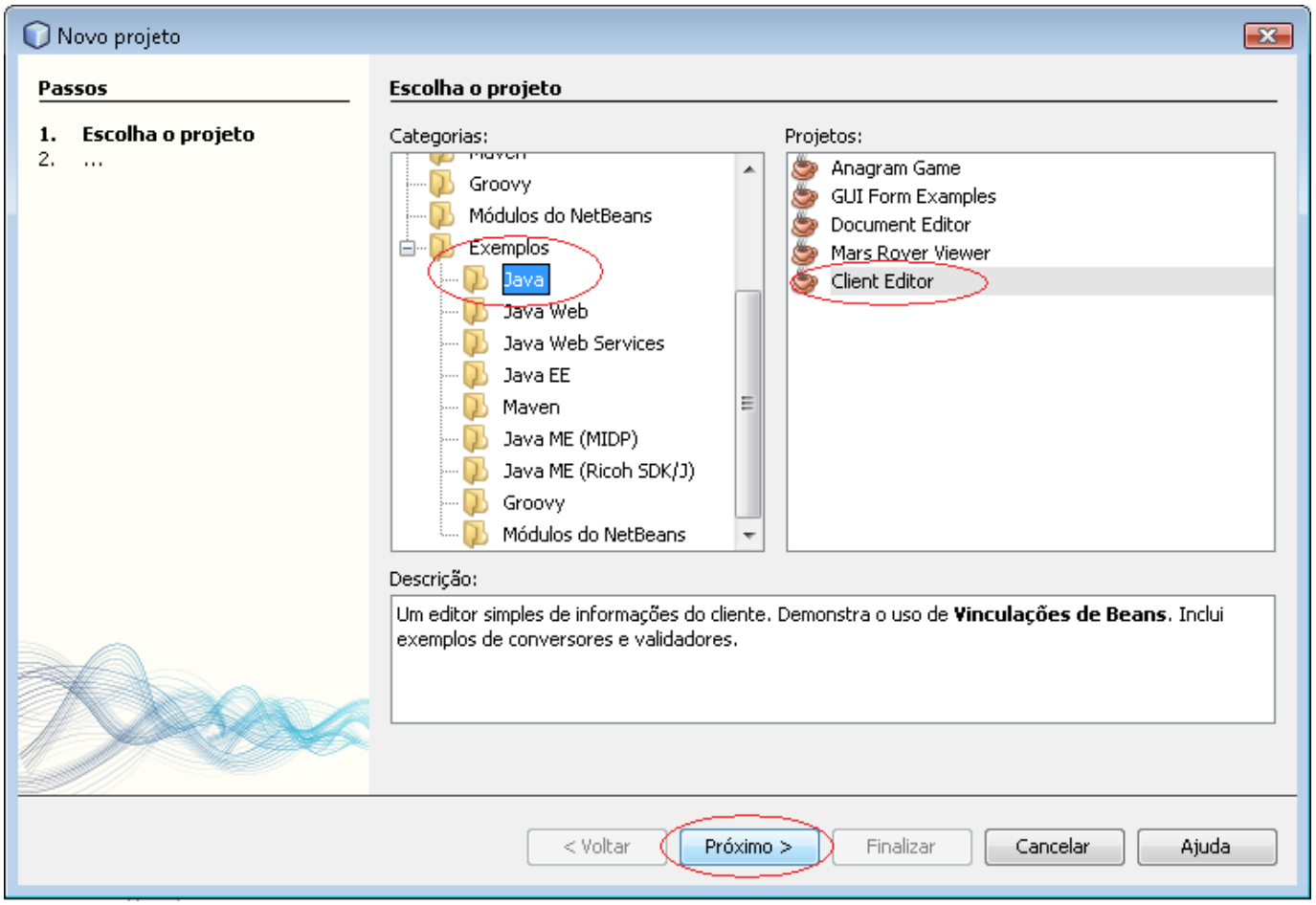
Configurando seu projeto no netbeans para gerar um único pacote jar

Neste post pretendo mostrar passo a passo como configurar seu projeto java desenvolvido no netbeans para que ele gere um único pacote JAR contendo todas as bibliotecas extras que você tenha adicionado ao seu projeto. Nem sempre precisamos de um único JAR, por default o netbeans gera apenas o JAR da aplicação, deixando pra fora, embora dentro do diretório de distribuição, as bibliotecas utilizadas no aplicativo.
Se o seu aplicativo não faz uso de nenhuma biblioteca extra então não existe o que se preocupar, a configuração padrão atende tranquilamente. O problema na distruição de um JAR e mais um conjunto de bibliotecas (um ou mais arquivos JARs) surge quando precisamos enviar, instalar, armazenar e principalmente dar suporte aos clientes. O ideal, pra manter a coisa extremamente simples (minha filosofia dentro da engenharia) é distribuir para o seu cliente apenas um único arquivo. Esse post demonstra extamente como distribuir apenas um JAR para o seu cliente. Este post foi inspirado pelo artigo deixado na referência, diferenciando ligeiramente em alguns aspectos, por exemplo, demonstrando como adicionar a splash screen no arquivo manifest.mf do pacote JAR gerado

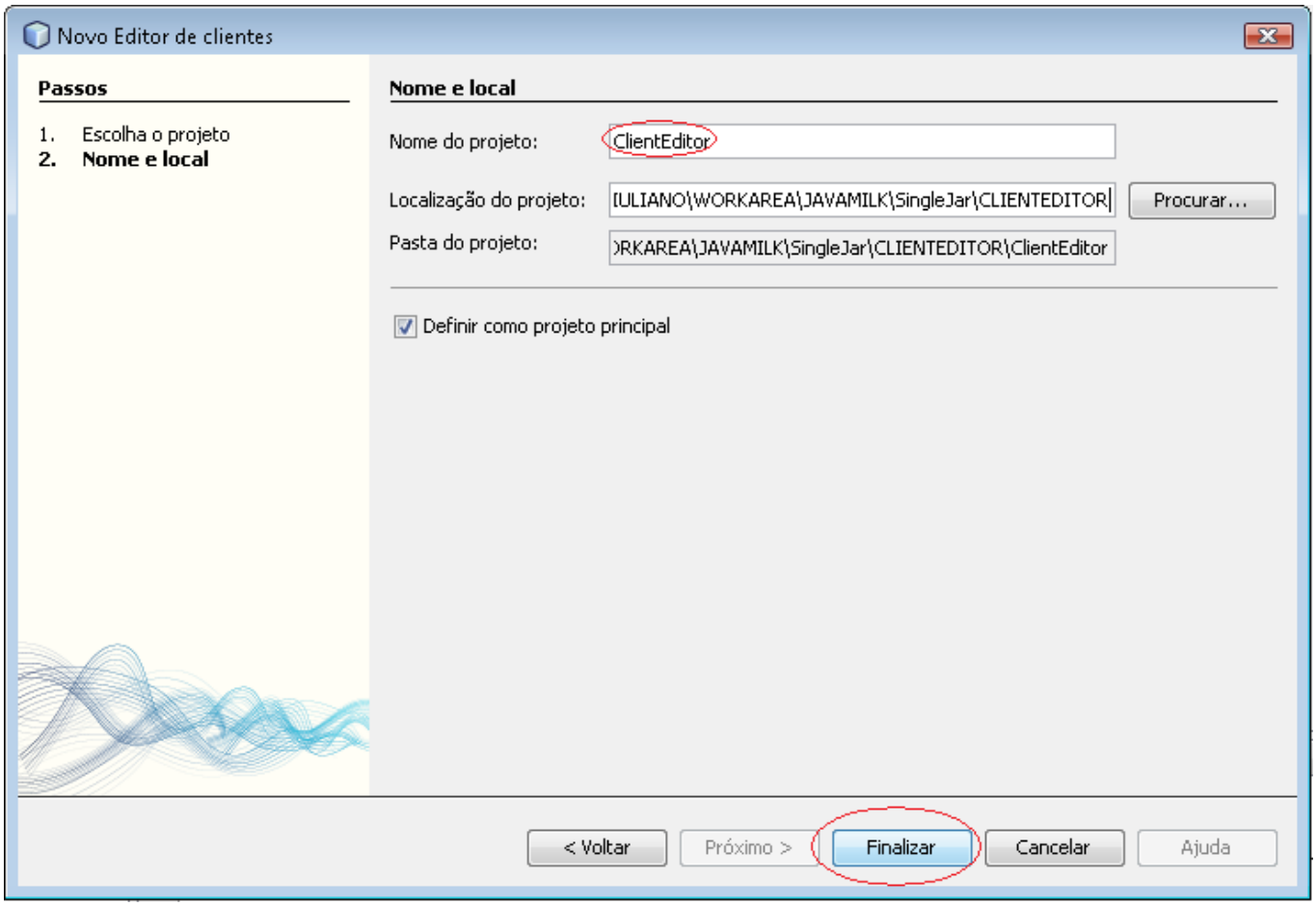
O primeiro passo é criar um aplicativo que faça uso de alguma biblioteca externa, não disponível no pacote default do Java SE 6. Para tal, vamos utilizar um recurso bacana do netbeans, a construção de um projeto exemplo, que faça uso de alguma(s) biblioteca (s) externa (s). No menu "Arquivo" do netbeans, escolha "Novo projeto"



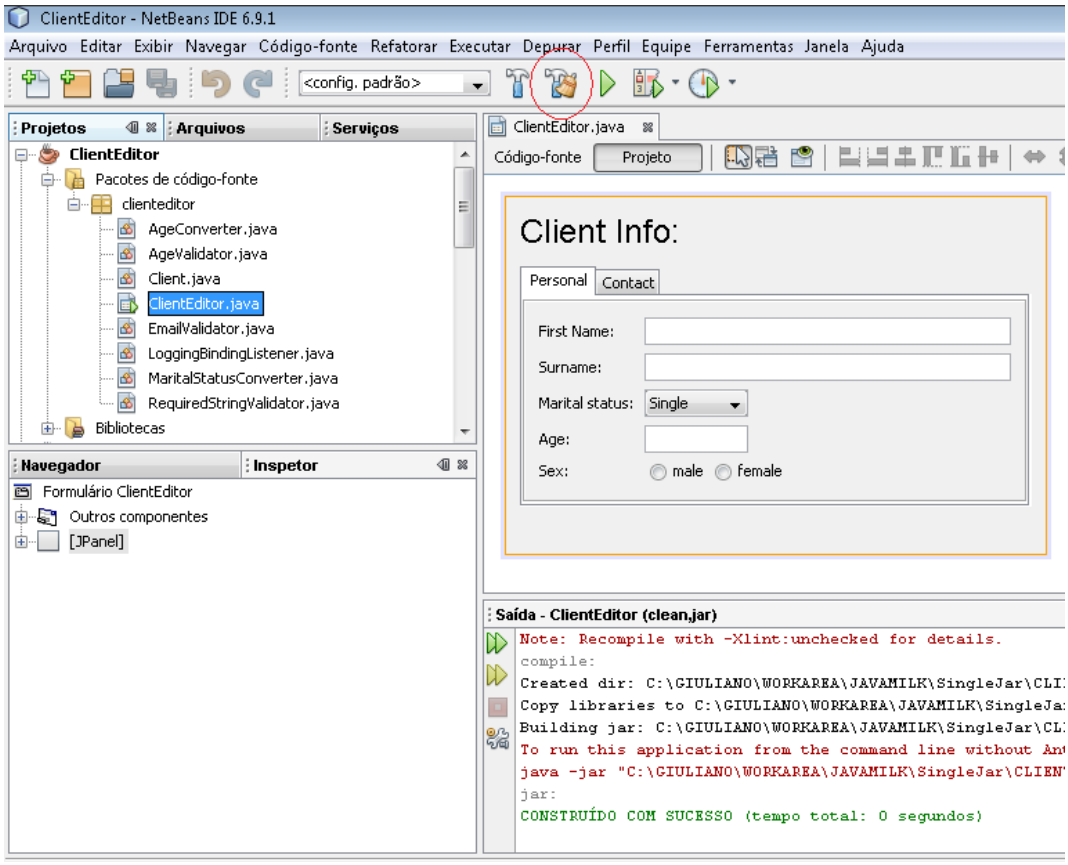
Selecione a categoria "Exemplos", "Java" e o projeto "Client Editor", em seguida clique em "Próximo".



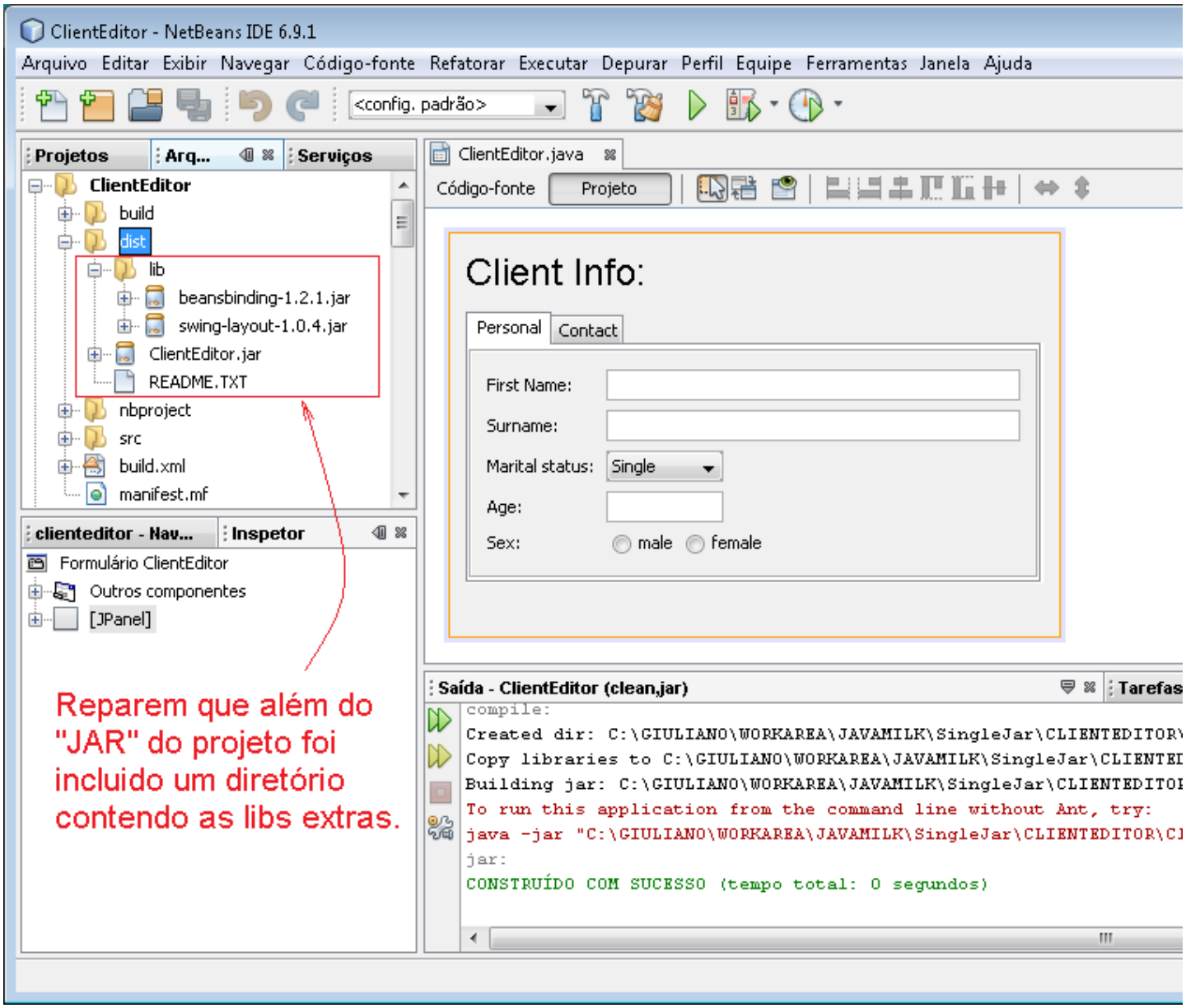
Escolha um local e um nome pra este projeto e clique em "Finalizar"



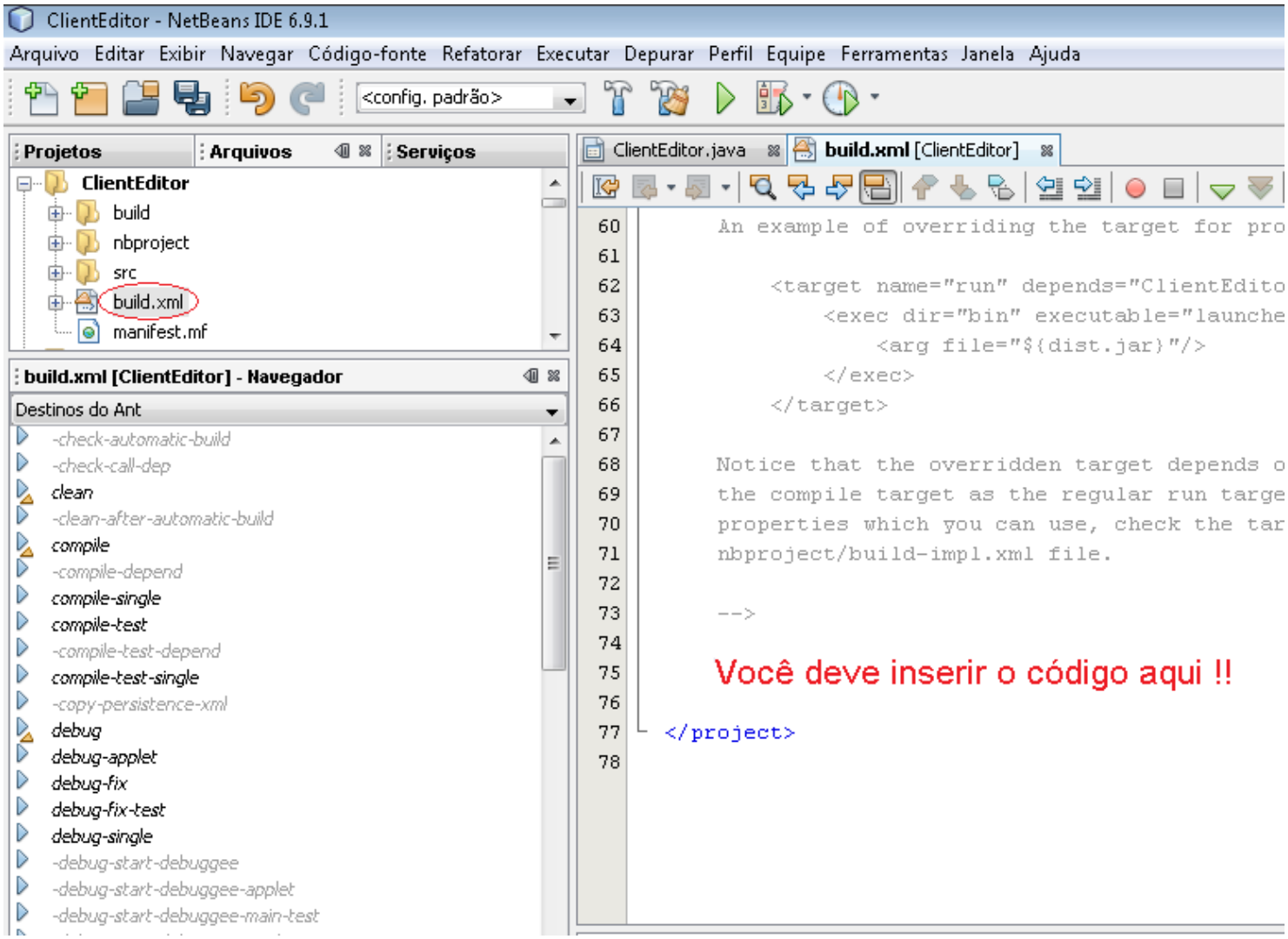
O netbeans constroi todo o projeto. Pra construirmos os arquivos de execução, neste caso o diretorio de distribuição (Jar do projeto + arquivos dependentes), basta clicar na ferramenta marcada abaixo.
Para executar o projeto, basta clicar no icone ao lado (play) ou então apertar F6.



Na aba "Arquivos" é possivel verificar que o passo anterior criou uma pasta chamada "dist", onde foi colocado o JAR do projeto e uma outra pasta de nome "lib" onde foram deixadas as bibliotecas extras utilizadas por este projeto. O netbeans fez tudo isso sozinho sem que precisassemos digitar uma linha sequer de código. Este é procedimento default do netbeans para criar o pacote de distribuição do aplicativo. Ele coloca tudo que você precisa enviar pro seu cliente na pasta "dist". O grande problema disso é se o seu cliente mover apenas o "jar" do projeto, ou então, por engano apagar alguma das bibliotecas utilizadas pelo projeto. Os passos a seguir demostram como gerar apenas um "jar" contendo tudo que for necessário pro projeto funcionar, sem a necessidade de enviar junto uma pasta contendo outros arquivos. Simples e fácil!



Para conseguirmos isso, vamos editar o arquivo "build.xml" do projeto (na aba "arquivos" do netbeans). Abra o arquivo no editor e posicione como mostrado na figura abaixo.

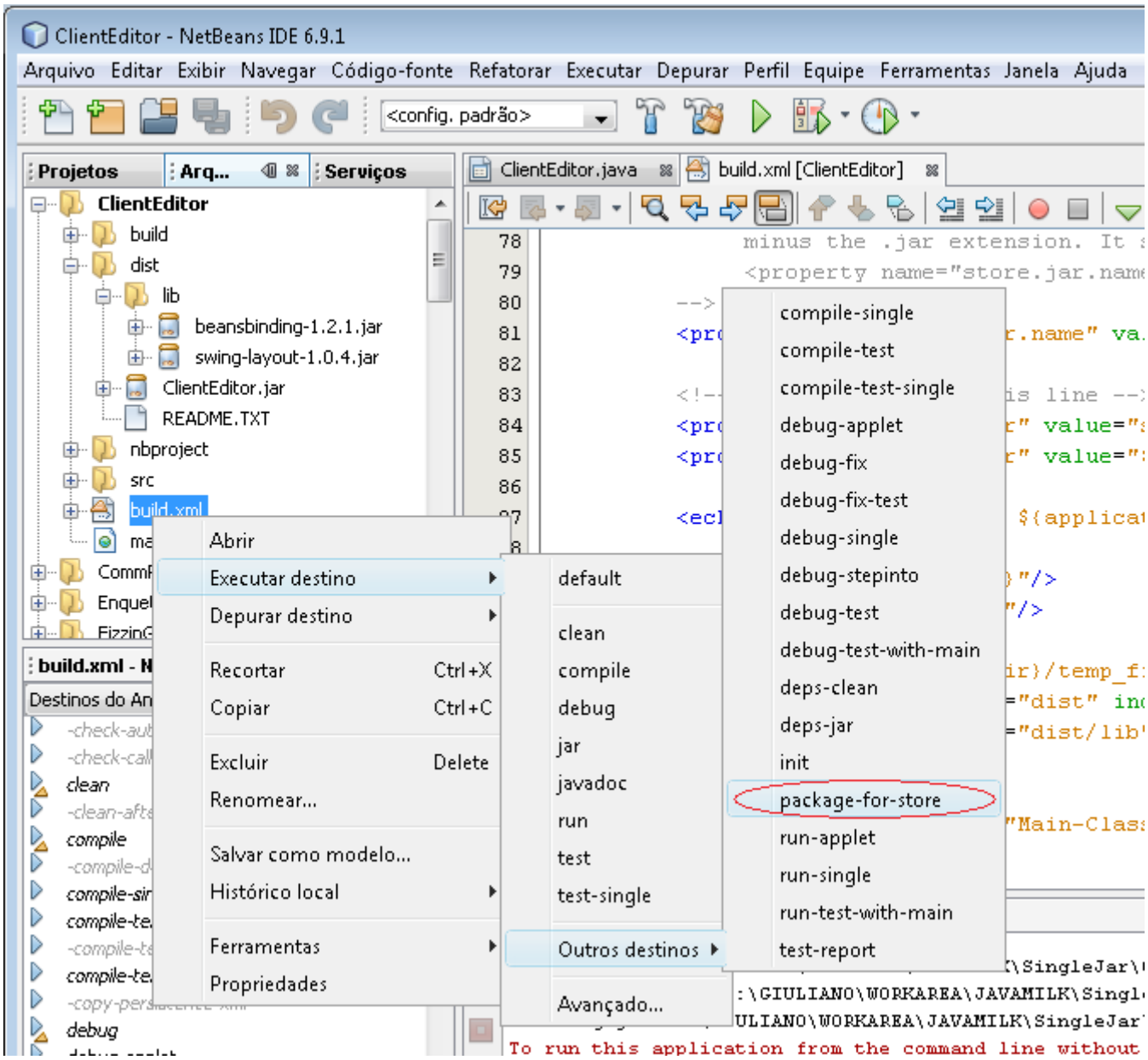


Agora basta inserir o código abaixo, ficando atendo APENAS quanto ao nome dado ao arquivo que será gerado. Lembre-se de modifica-lo para seu projeto. No meu exemplo use o mesmo nome do projeto "ClientEditor".

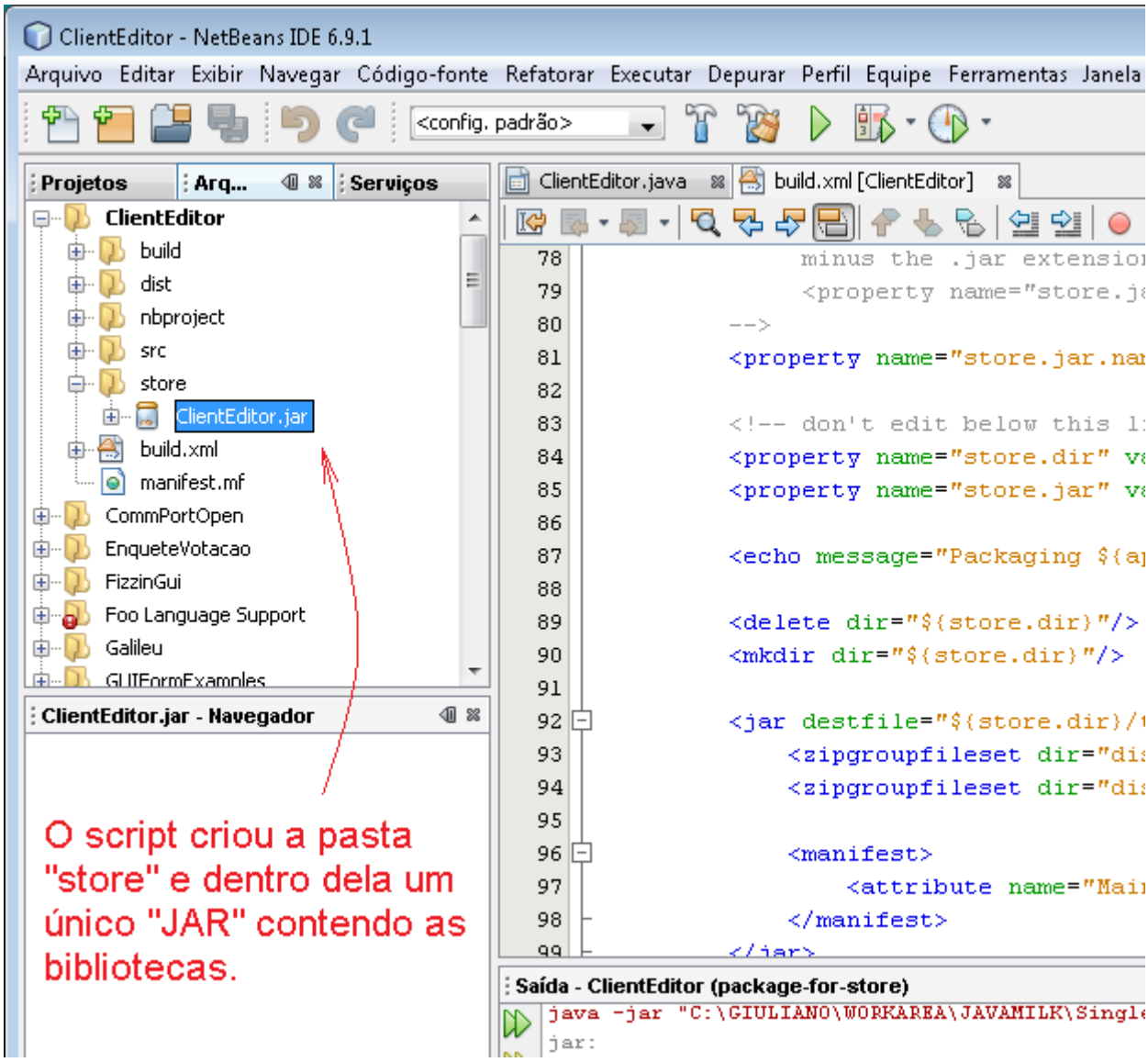
```
<target name="package-for-store" depends="jar">

<!-- Change the value of this property to be the name of your JAR,
minus the .jar extension. It should not have spaces.
<property name="store.jar.name" value="MyJarName"/>
-->
<property name="store.jar.name" value="ClientEditor"/>

<!-- don't edit below this line -->
<property name="store.dir" value="store"/>
<property name="store.jar" value="${store.dir}/${store.jar.name}.jar"/>

<echo message="Packaging ${application.title} into a single JAR at ${store.jar}"/>

<delete dir="${store.dir}"/>
<mkdir dir="${store.dir}"/>
```

```
<jar destfile="${store.dir}/temp_final.jar" filesetmanifest="skip">
<zipgroupfileset dir="dist" includes="*.jar"/>
<zipgroupfileset dir="dist/lib" includes="*.jar"/>

<manifest>
<attribute name="Main-Class" value="${main.class}"/>
</manifest>
</jar>

<zip destfile="${store.jar}">
<zipfileset src="${store.dir}/temp_final.jar"
excludes="META-INF/*.SF, META-INF/*.DSA, META-INF/*.RSA"/>
</zip>

<delete file="${store.dir}/temp_final.jar"/>

</target>
```

Para gerar o "arquivo JAR único do projeto", clique com o botão direito no arquivo "build.xml" e navegue até a opção "package-for-store" (como demostrado abaixo). A opção "package-for-store" é justamente o nome do script criado pelo código que adicionamos no passo anterior.



Executado o passo anterior teremos uma nova pasta "store" contendo um único "JAR" para o projeto. Essa pasta contém todas as bibliotecas extras utilizadas pelo projeto.



Apenas para concluirmos o post e agregar um pouco no artigo original, vale a pena ressaltar que o script anterior cria o arquivo manifest.mf do pacote gerado do zero, ou seja, não utiliza o arquivo manifest.mf deixado na raiz do projeto, como é o caso feito pelo procedimento default do netbeans. Neste caso, se por alguma razão você esteja modificando o arquivo manifest.mf do seu projeto (por exemplo, para adicionar uma splashscreen), você precisa modificar o script anterior, adicionando o(s) parâmetro(s) que você deseja que apareça(m) no arquivo manifest.mf do deu pacote JAR. A imagem abaixo demostra exatamente isso, a adição de um parâmetro no arquivo manisfest.mf. No meu exemplo, adicionei o parâmetro para chamada da imagem splash screen. Basta adicionar o trecho de código abaixo no script apresentado acima.

```
<manifest>
    <attribute name="Main-Class" value="${main.class}"/>
    <attribute name="SplashScreen-Image" value="ClientEditor/SplashScreenImg.png"/>
</manifest>
```

O post apresentado é bastante simples, porém pode nos ajudar a resolver uma dúvida que acabamos deixando de lado por não ter algo rápido. Na realidade a solução é muito simples e rápida e o resultado é fantástico. Um único arquivo JAR contendo tudo que o seu projeto precisa, sem dúvida simplificará seu processo de distribuição do projeto e garantirá um futuro mais tranquilo no que diz respeito ao suporte pro cliente. Uma boa semana a todos!