02

Compilar e executar testes com ANT

Downloads

Você pode baixar o ANT (zip) no site da Apache Software Foundation: aqui (http://ant.apache.org/bindownload.cgi)

Também segue o link para baixar o projeto que será utilizado nessa aula: Projeto agenda a importar no Eclipse (https://s3.amazonaws.com/caelum-online-public/PM-88/exercicios/agenda.zip)

Instalação do ANT no Windows, Mac e Linux

Para realizar a instalação do ANT no Windows ou sistemas Unix é preciso baixar a distribuição no site (http://ant.apache.org/bindownload.cgi)http://ant.apache.org/bindownload.cgi (http://ant.apache.org/bindownload.cgi). Escolha o arquivo, por exemplo apache-ant-1.8.x-bin.zip. Extraia o arquivo em uma pasta adequada. Dentro da pasta apache-ant-1.8.x existe a documentação, bibliotecas e, na pasta `bin`, os arquivos para executar o ANT na linha de comando. Para usar o ANT no sistema operacional Windows existe o executável `ant.bat`, para os sistemas UNIX o executável se chama `ant` apenas, ambos encontrados na pasta `bin`.

Configuração no Windows

É conveniente configurar a pasta `bin` da distribuição ant para fazer parte da variável de ambiente `PATH`. Assim, podemos executar o `ant.bat` a partir de qualquer pasta, não apenas dentro da pasta `bin`.

Por exemplo, vamos assumir que o Ant foi instalado na raiz da partição `C:\`, então é preciso adicionar o caminho `C:\apache-ant-1.8.3\bin` na variável PATH.

Para isso abra um terminal: Programas -> Acessórios -> Prompt de Comando

No terminal digite para modificar o PATH:

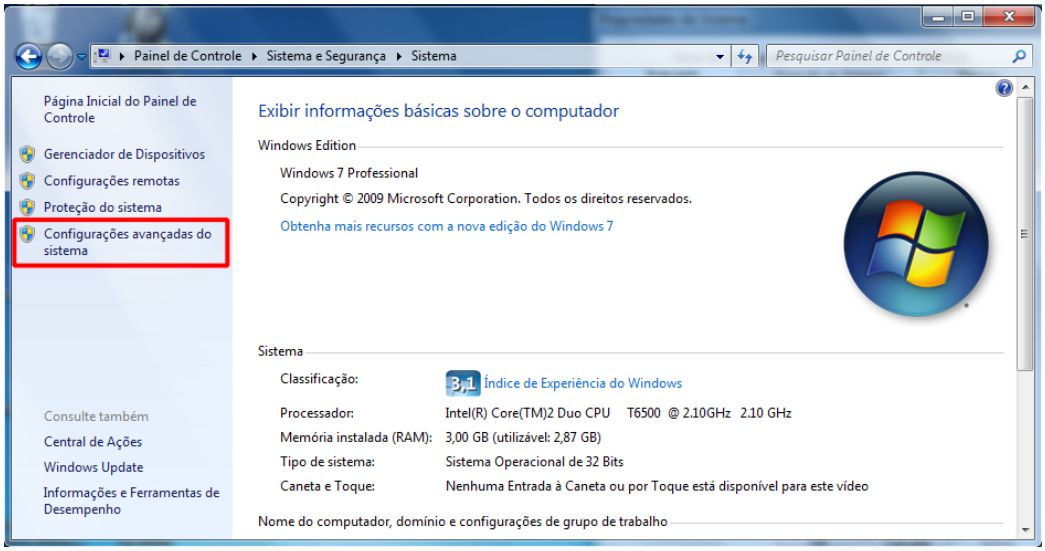
```
cd C:\ set PATH=%PATH%;C:\apache-ant-1.8.3\bin
```

E verifique a versão do ANT:

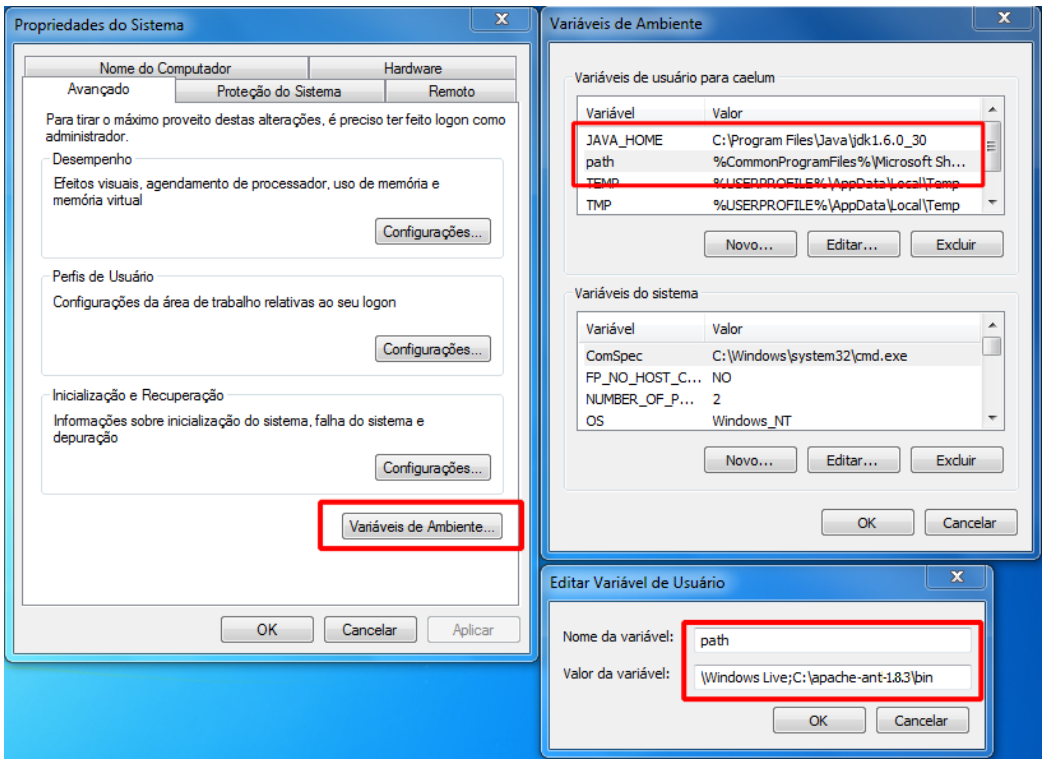
```
C:\Users\caelum>ant -version Apache Ant(TM) version 1.8.2 compiled on August 19 2011
```

A mudança da variável PATH pelo comando `set` só é valida durante da sessão da terminal. Abrindo um novo terminal temos que redigitar o comando. Para adicionar a pasta `bin` no PATH permanente altere a variável pela janela `Propriedades do Sistema`. No Windows 7 chegamos a ele através de:

Panel de Controle -> Sistema e Segurança -> Sistma -> Configurações avançadas do sistema.



Dentro da janela `Propriedades do Sistema` é preciso alterar a variável `path` :



Instalação no Mac ou Unix Based

Após ter descompactado a distribuição do ANT precisamos adicionar a pasta `bin` no path para poder executar de qualquer lugar:

```
cd echo "export PATH=\$PATH:/Users/caelum/apache-ant.../bin" >> ~/.profile
```

Feche seu terminal! Abra novamente e confirma:

```
ant -version Apache Ant(TM) version 1.8.2 compiled on August 19 2011
```

Instalação com APT-GET no Linux

A instalação mais simples ocorre nos sistemas que possuem suporte ao APT-GET, como o Ubuntu. Execute no terminal o comando:

```
apt-get install ant
```

Nesse caso o ANT automaticamente fará parte do `PATH` do sistema operacional. Basta conferir:

```
ant -version Apache Ant(TM) version 1.8.2 compiled on August 19 2011
```

Compilação das classes de teste

É preciso garantir que as classes de teste foram compiladas antes de executar testes. Como já vimos, a tarefa `javac` tem esse papel que utilizaremos novamente para compilar as classes de teste. Vamos criar um novo target `compilar-teste` dentro de uma target:

```
<target name="compilar-teste" depends="compilar">
        <javac destdir="build/classes-teste" srcdir="src-teste" >
            <classpath>
            <!-- definição do classpath-->
        </classpath>
       </javac>
</target>
```

Repare que as classes são compiladas para uma pasta separada `build/classes-teste` . Dentro da tag `classpath` é preciso definir as dependências das classes de testes. As dependências aqui também são, além das classes da aplicação, as bibliotecas da aplicação (fileset `WEB-INF/lib`) e as bibliotecas de testes (fileset `lib-teste`). Veja a configuração:

```
<classpath>
    <pathelement path="build/classes" />

    <fileset dir="WebContent/WEB-INF/lib">
         <include name="*.jar" />
    </fileset>

    <fileset dir="lib-teste">
         <include name="*.jar" />
    </fileset>
</classpath>
```

Executando testes no build do software

Executar testes é uma fase valiosa no nosso processo de build, no ANT já existe uma tarefa `junit` definida para este objetivo. A tarefa possui algumas propriedades para mostrar um relatório no final (`printsummary`) e configurar a saida no console (`showoutput`). Além disso, podemos definir que `junit` deve parar a execução caso aconteça alguma falha durante os testes.

Para executar a tarefa ela deve estar dentro de um target que chamaremos `executar-testes` . O esboço inicial do target fica assim:

```
<target name="executar-testes">
  <junit haltonfailure="yes" showoutput="true"  printsummary="on" >
    ...
```

```
    </junit>
  </target>
```

A tarefa `junit` deve achar o código fonte dos testes. Para isso definimos o caminho das classes de teste usando a tag `fileset` como já vimos anteriormente. Nele indicamos em qual pasta o código está (`src-teste`) e qual é o sufixo (`*Test.java`):

```
    <fileset dir="src-teste">
        <include name="**/*Test.java" />
    </fileset>
```

Quando os testes são executados um relatório final deve mostrar o resultado. Para isso podemos definir um formato de exibição através da tag `formatter` . Existem 4 tipos de formatos (xml, plain, brief e failure), iremos usar aqui o plain através do atributo `type` :

```
    <formatter type="plain" />
    <fileset dir="src-teste">
        <include name="**/*Test.java" />
    </fileset>
```

O `formatter` e o `fileset` são configurados pela tag `batchtest` que informa a tarefa `junit` em qual pasta queremos gerar o relatório, isso pelo atributo `todir` .

Fora do `batchtest` é preciso definir o `classpath` para `junit` rodar os testes. O `classpath` é composto das bibliotecas e classes da aplicação E dos testes. O target completo com dependência do target `compilar-teste` fica como:

```
  <target name="executar-testes"  depends="compilar-teste">
    <junit haltonfailure="yes" showoutput="true"  printsummary="on" >
      <batchtest todir="relatorio-teste">
          <formatter type="plain" />
          <fileset dir="src-teste">
              <include name="**/*Test.java" />
          </fileset>
      </batchtest>
      <classpath>
          <pathelement location="build/classes"/>
          <pathelement location="build/classes-teste"/>

          <fileset dir="WebContent/WEB-INF/lib">
              <include name="*.jar" />
          </fileset>

          <fileset dir="lib-teste">
              <include name="*.jar" />
          </fileset>
          </classpath>
    </junit>
  </target>
```

Repare que existem 4 definições dentro da `classpath` , duas relacionadas com a aplicação e duas relacionadas com os testes. Também não podemos esquecer criar a pasta `relatorio-teste` que é preciso para disponibilizar os resultados

dos testes :

```
<mkdir  dir="relatorio-teste" />
```

Assim podemos executar o target `executar-testes` com ANT, por exemplo:

```
ant executar-testes
```

Com uma possível saída no console:

```
... compilar-testes: [echo] Compilando classes de teste


executar-testes: [junit] Running br.com.caelum.vraptor.controller.ContatosControllerTest [junit] Running
br.com.caelum.vraptor.controller.PessoasControllerTest [junit] Tests run: 3, Failures: 0, Errors: 0, Time elapsed:
0,777 s ...
```

Uma cadeia de targets muito comum é:

executar-teste (junit) -> compilar-teste (javac) -> compilar (javac) -> limpar (delete, mkdir)