Péricles: Roadmap

Guia de desenvolvimento

Um não muito pequeno guia para os membros do projecto Péricles que explica como obter a árvore de desenvolvimento do projecto e como efectuar as suas próprias contribuições.

Martinho Fernandes

Abril 2009

Péricles: Roadmap

Guia de desenvolvimento

Índice

Começar

Instalar o cliente de *Subversion*

A primeira coisa a fazer é instalar o *TortoiseSVN*. É um cliente de *Subversion* que funciona como uma extensão do *Explorador do Windows*. É bastante fácil de usar. Façam o download, instalem e reiniciem o *Windows*.

Password do *Google Code*

O projecto está alojado no *Google Code*. Para poderem submeter alterações é necessária uma conta do *Google* e uma *password*. Esta *password* não é a *password* que usam para fazer *login* no *Google*. É uma *password* gerada automaticamente e o *Google Code* não permite que seja alterada[1]. Podem ver a vossa password acedendo à página do projecto (depois de fazerem *login* no *Google*), no separador *Source* > *Checkout*.

Checkout

Para obterem a árvore do projecto devem começar por navegar para a pasta que vão usar para trabalhar localmente. Depois basta fazer clique com o botão direito do rato num espaço vazio e seleccionar a opção *SVN Checkout*.

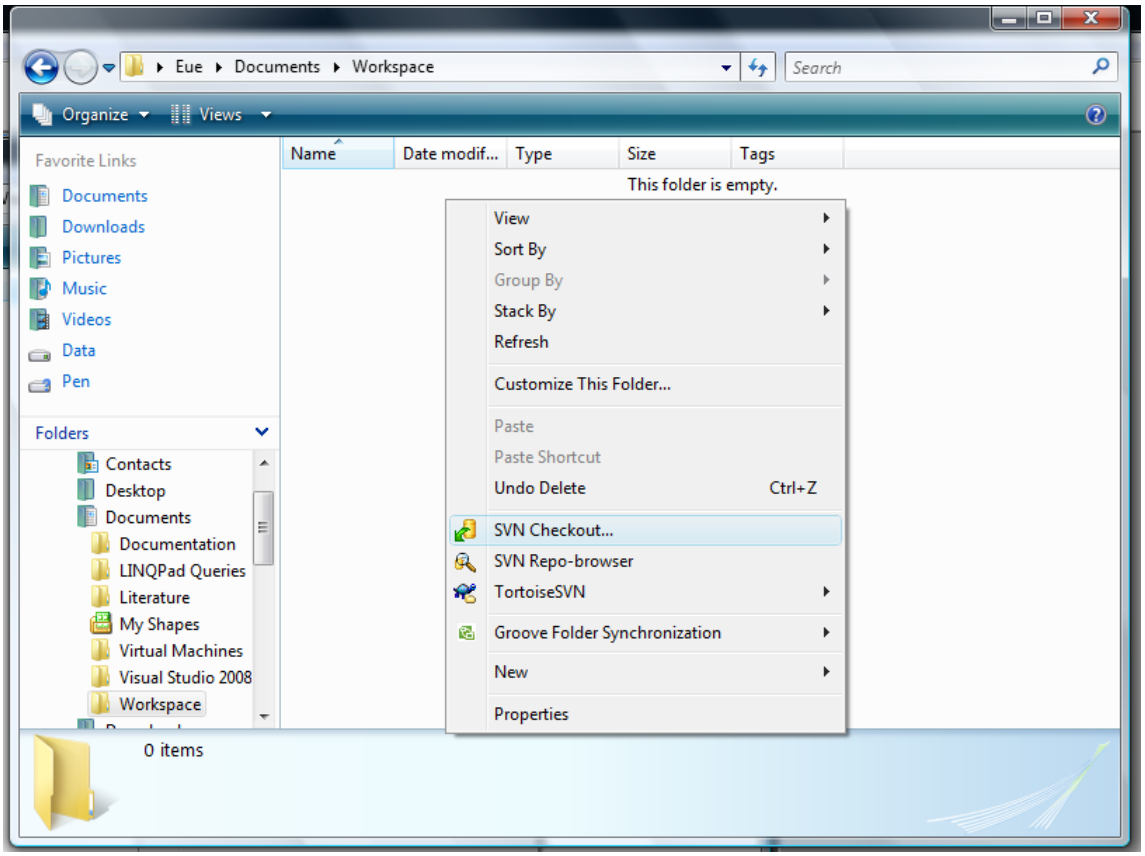

Figura 1

[1] Pode ser gerada novamente de forma aleatória, só não pode é ser alterada para algo à vossa escolha.

O URL do repositório é *https://pericles.googlecode.com/svn/trunk*. O único outro campo importante no diálogo que surge é a pasta para fazer o *checkout*. É nessa pasta que vai ficar a árvore do projecto.

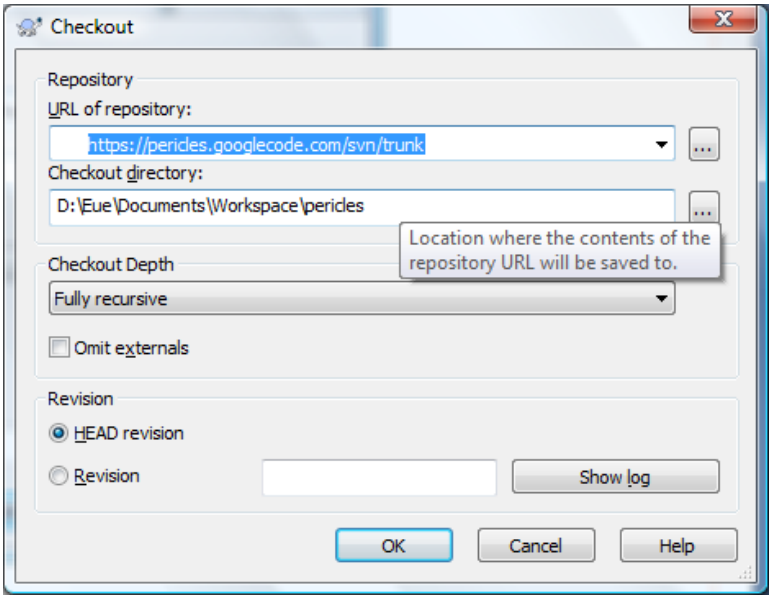

Figura 2

Agora é *OK*, introduzir o *username* e a *password* e esperar. Se quiserem podem seleccionar a opção para guardar as credenciais para não terem de reintroduzi-las sempre que fizerem *update* ou *commit*. O repositório é um pouco grande (quase 50 Mb) porque contém várias ferramentas e bibliotecas que serão úteis. Depois do primeiro *checkout* apenas as alterações têm de ser descarregadas e por isso será ***muito*** mais rápido.

A árvore

No final, a estrutura da vossa pasta deve ter um aspecto semelhante a este:

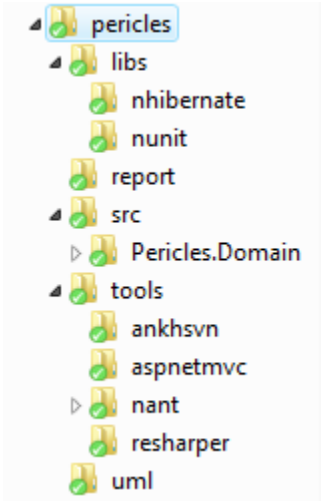

Figura 3

Ferramentas

Na pasta *tools* encontram-se ferramentas úteis para o projecto. Neste momento podem encontrar as seguintes:

- *AnkhSVN*: um cliente de *Subversion* que é um *addin* para o *Visual Studio*. Serve para aceder ao repositório no *Google Code* directamente a partir do *Visual Studio*.
- *ASP .NET MVC*: um update que permite usar o padrão *MVC* (*Model-View-Controller*) para criar sites *web*. Este padrão torna o código mais fácil de manter e testar.
- *FxCop*: uma ferramenta que analisa o código e procura potenciais erros e más práticas. É desejável que nada seja apontado pelo *FxCop*. Se esta ferramenta indicar muitos erros e *warnings* significa que o código é pouco saudável.
- *GhostDoc*: um pequeno *addin* para o *Visual Studio* que ajuda a documentar o código.
- *NAnt*: o *make*, *on steroids*, para *.NET*. É usado para correr testes e análises ao código automaticamente.
- *ReSharper*: outro *addin* para o *Visual Studio*. É um *must-have*. Ajuda a manter o estilo de código, detecta muitos erros e falhas prematuramente e ajuda a fazer *refactorings*. Também serve para correr testes directamente a partir do *Visual Studio*.

Bibliotecas

A pasta *libs* é para as bibliotecas que serão usadas no projecto. Todas as bibliotecas que forem usadas devem estar nesta pasta. A única excepção são as bibliotecas *.NET standard*. Neste momento podem encontrar duas bibliotecas nesta pasta:

- *NUnit*: uma biblioteca para escrever *unit tests*. Estes testes podem depois ser executados no *Visual Studio* usando o *ReSharper*.
- *NHibernate*: uma biblioteca para abstrair o código de acesso à base de dados. Com esta biblioteca a camada de dados torna-se muito mais simples.

Código

A pasta *src* é onde ficará todo o código do projecto. Nesta pasta encontra-se a solução do *Visual Studio 2008* do Péricles. A solução será dividida em vários projectos: um para cada camada. Não é obrigatório que cada camada fique sempre num projecto separado, mas assim é mais fácil garantir que as camadas estão bem separadas. Também deve existir um projecto de teste para cada camada.

Para além do ficheiro da solução, existem outros ficheiros nesta pasta:

- *Bevonn.snk*: uma chave para assinar digitalmente as *DLLs* do projecto. A razão principal para isto é porque dá um ar mais oficial à coisa.
- *build.bat*: este *script* compila o Péricles pela linha de comandos (invoca o *nant*), sem ser necessário abrir o *Visual Studio*.
- *CommonAssemblyInfo.cs*: um ficheiro C# com *metadata* comum a todos os projectos.
- *default.build*: uma espécie de *Makefile* para *.NET*. É para ser usado com o *nant*. Serve para compilar, analisar e testar o código.
- *Pericles.4.5.resharper*: ficheiro de estilo para o *ReSharper*. Serve para manter o estilo do código consistente em todo o lado.

- *Pericles.FxCop*: projecto do *FxCop*. Serve para analisar o código e encontrar possíveis erros.

A maior parte destes ficheiros serão modificados apenas raramente, ou mesmo nunca.

UML

Na pasta *uml* fica o modelo UML do projecto, um ficheiro do *Visio* com todos os diagramas criados.

Relatório

Como o nome indica, o relatório do projecto fica na pasta *report*.

Usando *Subversion*

Agora que já têm a vossa cópia do projecto, já estão prontos para trabalhar.

Overlays

Devem ter reparado que os ficheiros e as pastas do projecto têm uma marca verde. Isto significa que vocês não efectuaram alterações. Quando alterarem um ficheiro (ou uma pasta) esse ficheiro vai ficar com um ponto de exclamação vermelho. Isso significa que têm alterações que ainda não foram aplicadas ao repositório.

Commit

Para aplicar essas alterações ao repositório devem efectuar um *commit*. Naveguem para a pasta raiz do projecto e cliquem com o botão direito num espaço vazio. No menu que surge, seleccionem a opção *SVN Commit*.

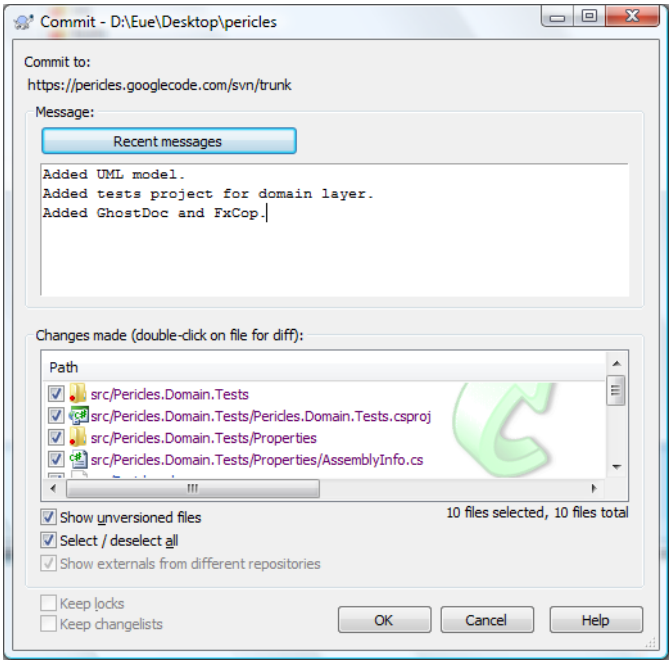

Figura 4

Na caixa de diálogo que surge devem preencher caixa de texto do topo com uma mensagem curta mas descriptiva das alterações que efectuaram. Na parte de baixo podem confirmar quais foram os ficheiros modificados.

Depois é *OK* e esperar que as alterações sejam enviadas para o repositório. Como só as alterações é que são enviadas não deve levar muito tempo.

Esta operação pode falhar por a vossa cópia estar desactualizada. Se for esse o caso devem actualizá-la primeiro, resolver quaisquer conflictos que surjam e depois repetir o *commit*.

Update

Devem actualizar a vossa cópia periodicamente para receber as alterações efectuadas pelos outros. Para isso devem seleccionar a opção *SVN Update* no menu de contexto na pasta raiz do projecto. Surge uma janela que mostra o progresso do *download*. Quando terminar podem clicar em *OK*. A vossa cópia está agora actualizada.

Após um *update* devem sempre confirmar que as vossas alterações são "compatíveis" com as alterações que receberam. Por exemplo, se fizeram alterações no código devem confirmar que ainda compila e funciona conforme esperado. Se tal não acontecer devem resolver o problema antes de fazerem *commit*.

Podem surgir conflitos ao fazer um *update*. Isto acontece porque as vossas alterações locais não podem ser fundidas automaticamente com as alterações efectuadas pelos outros. Nesta situação devem resolver os conflitos antes de fazer *commit*. Se precisarem de ajuda para resolver o conflicto devem discutir a situação com o responsável pelas outras alterações para o resolverem em conjunto.

Se quiserem saber que alterações foram efectuadas pelos outros devem seleccionar a opção *View log* no submenu do *TortoiseSVN*.

Outras operações

Há outras operações que podem efectuar. Estas são as mais comuns:

- *Show log*: ver o histórico de alterações;
- *Add*: adicionar um ficheiro ou pasta;
- *Diff*: comparar um ficheiro com uma versão anterior para ver as alterações;
- *Revert*: anular alterações efectuadas num ficheiro ou numa pasta.

As outras operações disponíveis ou são óbvias, ou não são muito úteis, ou são operações avançadas que requerem alguma experiência. Destas últimas destacam-se as operações de *branch*, *switch* e *merge*. Explicarei estas se e quando surgir a necessidade de usá-las.

Recomendações

Há apenas uma regra básica que devem seguir a todo o custo: não fazer *commits* com erros. Se o fizerem, esses erros vão ser propagados para toda a gente quando fizerem *update*. Embora seja possível reverter o repositório para o estado anterior, é sempre chato quando isso acontece.