



Controle de Versão com Subversion e TortoiseSVN

Erick Sasse



O que vamos ver?



- Exemplos práticos, slides apenas para referência.
- Como usar o Subversion através do TortoiseSVN.
- Como organizar o repositório.
- Um repositório para tudo ou repositórios separados?
- Como compartilhar código entre projetos.
- Como manter código de terceiros no repositório.
- Branching e Merging, quando e como usar.
- Instalando um servidor SVN na sua rede local em poucos cliques.
- Diferenças entre controle de versão centralizado como Subversion e distribuído como Mercurial e Git.





- Comecei profissionalmente no Clipper
- Delphi desde a versão 1
- Um pouco de Xcode para iPhone e iPad
- Palestrante BorCon, Delphi Developers Day, Firebird Conference 2006 em Praga, FDD, entre outros.
- Gerente das equipes de desenvolvimento na Cadena Sistemas e na Monde Sistemas.
- 5 anos usando Subversion.



Apache Subversion



- Surgiu em 2000
- Sistema de controle de versão centralizado
- Um dos mais usados no mundo
- Em 2010 graduou-se como um projeto da Apache Software Foundation
- <http://subversion.apache.org/>





- Cliente Subversion implementado como uma extensão do shell do Windows (Explorer).
- Compilado com código do próprio SVN, se o repositório estiver na mesma máquina, nem precisa de servidor.
- O melhor cliente SVN que eu conheço.
- <http://tortoisesvn.net/>



Comandos Básicos



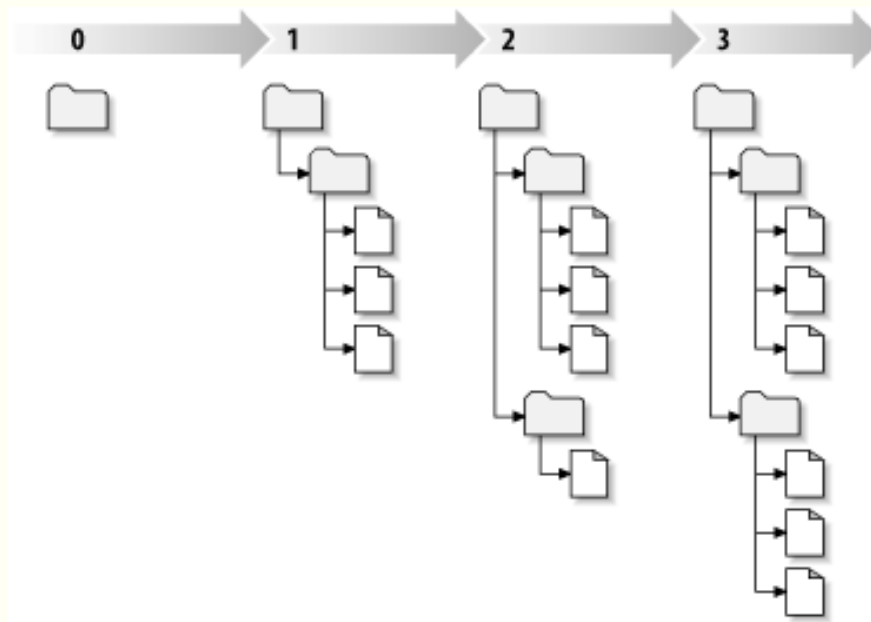
- Create repository
- Import
- Checkout
- Update
- Commit
- Log



Revisões



- As revisões são globais e representam a situação de todo o repositório após um commit.
- O mesmo arquivo em revisões diferentes não necessariamente foi modificado.
- Não existe versão por arquivo.





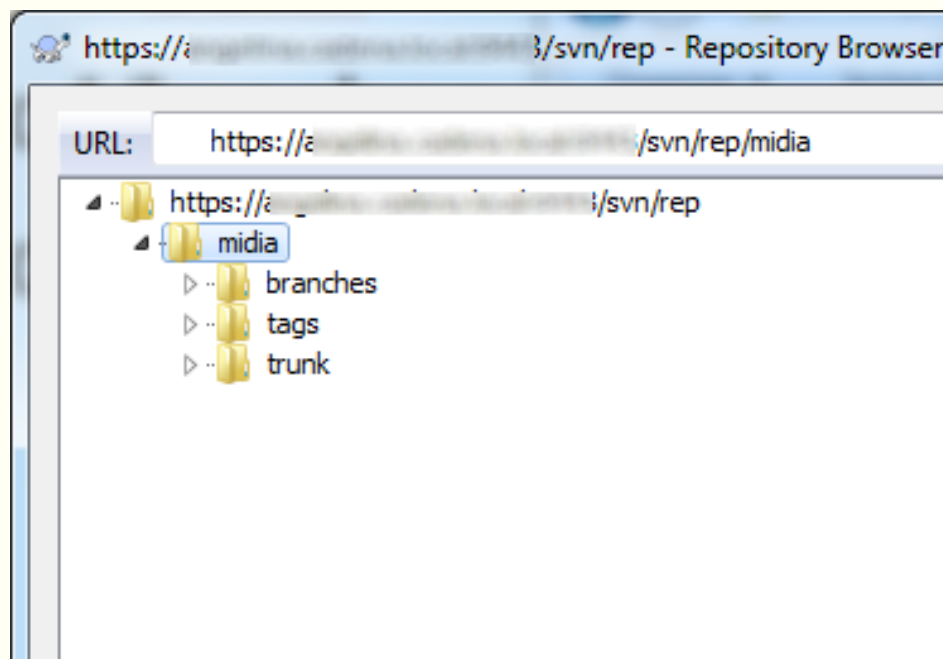
- Atualizar sua working copy
 - Update
- Fazer suas modificações
 - Alterar arquivos, incluir novos, excluir, etc.
- Revisar suas modificações
 - Check for modifications
- Receber alterações dos outros
 - Update
- *Commitar* suas modificações
 - Commit



Estrutura do Repositório



/projeto1/trunk
/projeto1/braches
/projeto1/tags
/projeto2/trunk
/projeto2/braches
/projeto2/tags



Trunk



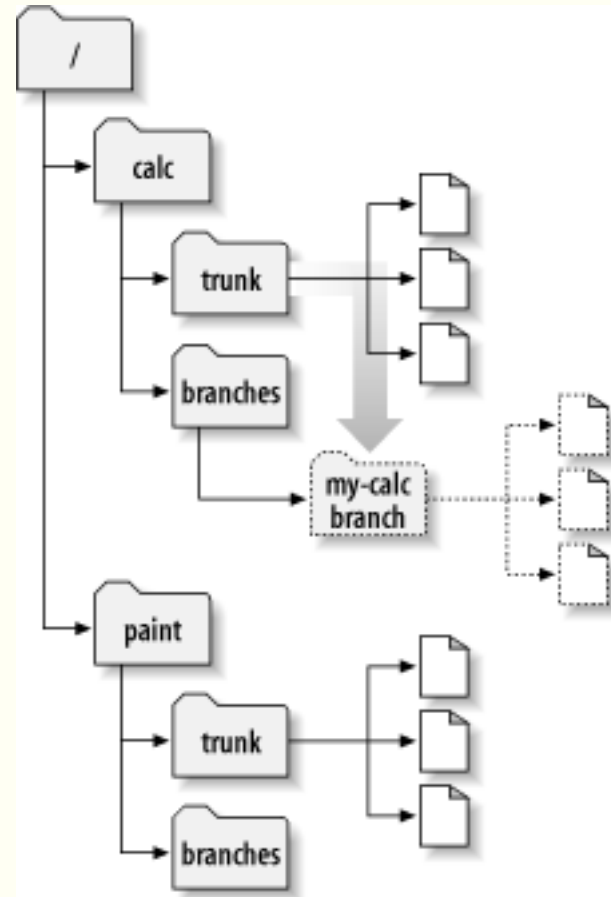
- O trunk é a linha de desenvolvimento principal. É normalmente onde fica sendo desenvolvida a próxima versão do projeto.



Branches



- Linhas paralelas de desenvolvimento.
- SVN faz “cheap copies”, ou seja, não copia os arquivos de verdade até que sejam alterados, apenas faz “hard links”.



Release Branch



- Criada a partir da trunk quando se chega perto da finalização de um release (todos os novos features já implementados, bugs conhecidos corrigidos).
- Cria-se essa branch para liberar o trunk para implementação das novas features para o próximo release.
- Bugs encontrados no release são corrigidos na RB e depois feito merge para trunk.



Feature/Experimental Branch



- Normalmente criada a partir do trunk.
- Usada para desenvolver novos recursos de maior impacto no sistema, que deixariam o build quebrado por dias.
- Também para quando não se sabe quando o recurso vai ser introduzido no sistema. (no trunk fica sempre o próximo release)
- Para recursos experimentais, que podem não ser incluídos no projeto.
- Exemplo:
 - /projeto/branches/FB-NFE



Mantendo Código de Terceiros



- Coloque todo código de terceiros no seu repositório.
- Muito útil para automatizar os builds, para manter histórico, etc.
- Colocar todos abaixo de um caminho específico (/vendor).
- A última versão fica no caminho /current e a cada nova versão comitada, criar uma tag com o número da versão.

`/vendor/devexpress/current`

`/vendor/devexpress/v1`

`/vendor/devexpress/v2`



Compartilhar Código Entre Projetos



- External permite que sua working copy seja composta por vários checkouts de locais diferentes.
- Você pode incluir outros caminhos do seu próprio repositório ou de outros repositórios totalmente diferentes localizados na sua rede ou fora dela.

Exemplo:

```
^/lib/trunk/ lib
```

Um external como esse cria uma pasta lib dentro da pasta do projeto no momento do checkout e traz o código que está no caminho /lib/trunk do repositório.



Compartilhar Código Entre Projetos



- Externals também são usados para incluir código de terceiros no projeto:

```
^/vendor/devexpress/current vendor/devexpress
```

```
^/vendor/corelab/dbxida/current vendor/corelab/dbxida
```



Um repositório ou vários?



- Minha recomendação é se possível usar um único repositório para todos os projetos.
- Uma das principais facilidades é poder com um único número de revisão saber exatamente a situação de qualquer arquivo, seja ele de uma biblioteca compartilhada, de um componente de terceiro ou do seu próprio projeto.
- Considere a possibilidade de usar a revisão do repositório na própria versão do seu projeto, exemplo:
 - Hello World 1.0.0.1234
 - Nesse caso o 1234 seria a revisão do seu repositório usada para este build.





- Servidor Subversion gratuito para Windows
- Já vem com webserver (Apache)
- Integra-se com Active Directory para controle de acesso.
- Roda como serviço.
- Instalação NNF (Next, Next, Finish).
- <http://www.visualsvn.com/server/>



Centralizado x Distribuído



- Controle de versão distribuído é uma evolução do modelo centralizado como o Subversion.
- Esta na moda.
- Principais diferenças:
 - Todos os desenvolvedores tem o repositório completo na máquina.
 - Os commits são todos locais e você decide quando enviar as modificações para um repositório central.
 - Desenvolvedores podem trocar modificações entre seus repositórios sem tocar o repositório central.
- Principais opções: Git e Mercurial
- Ainda não migrei por não sentir que os benefícios valem o trabalho envolvido.





- Livro online gratuito sobre Subversion
 - <http://svnbook.red-bean.com/>
- TortoiseSVN Add-In Delphi
 - <http://tinyurl.com/svn-delphi>
- Plug-in para Visual Studio:
 - <http://www.visualsvn.com/visualsvn/>
- Cliente para MacOS:
 - <http://versionsapp.com/>



Obrigado!



- www.ericksasse.com.br
- esasse@gmail.com
- twitter.com/esasse
- Xbox Live: Jogador76

