**IGITA**

17

**Trabalhando com o SVN no Eclipse**

**Igor Ivanoff Takats**

Configurando o repositório de Eclipse 3

Introdução **3**

Objetivos **3**

Pré-Requisitos 3

**Introdução**

Neste pequeno tutorial vou apresentar uma descrição básica de como se trabalhar com o Subversion (SVN) no Eclipse.  
Embora outros servidores de controle de versão estejam sendo utilizados cada vez mais pelo mercado, como o Git, por exemplo, o SVN ainda é, sem dúvida, utilizado por uma grande quantidade de usuários nas empresas.  
Apesar da grande utilização, nem sempre as melhores práticas são adotadas quando se trabalha com este tipo de ferramenta, nem mesmo recursos como merging e branching são utilizados na maioria das vezes (ou pela maioria dos usuários), o que acaba por provocar a  subutilização da ferramenta, afetando consequentemente a produtividade das equipes.

**Objetivos**

Apresentar orientações básicas de como trabalhar com o SVN no Eclipse, que é um dos ambientes de desenvolvimento Java mais utilizados do mercado.  
Serão apresentados exemplos de como criar um branch, realizar um merge entre o branch e o trunk, e resolver problemas de conflito durante o merge.  
Os conceitos apresentados são válidos para outros ambientes de desenvolvimento, mas os procedimentos podem mudar de acordo com os plugins utilizados em cada um.

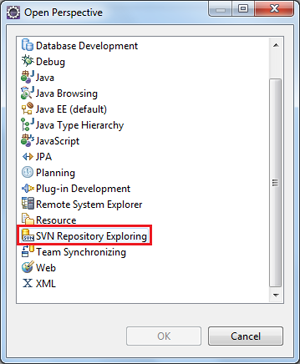
**Pré-Requisitos**

Como não é o foco do tutorial apresentar a instalação do servidor SVN, é necessário ter um repositório disponível para os testes. Uma dica, para se testar em uma máquina local, é instalar o VisualSVN Server Manager.  
É necessário também ter o plugin do SVN instalado no Eclipse. Um exemplo de como a instalação do plugin pode ser feita pode ser encontrado no seguinte endereço: <https://luizgustavoss.wordpress.com/2009/12/13/preparando-um-ambiente-de-desenvolvimento-java-ee-baseado-em-eclipse/>

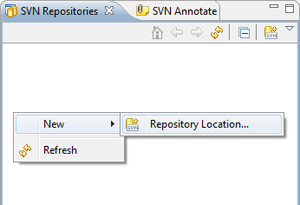
**Mão na Massa!**

**Configurando o repositório no Eclipse**

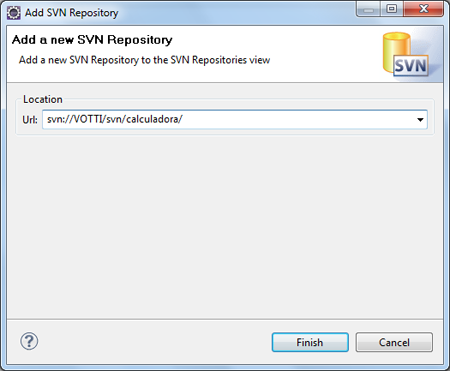
Para começar a utilizar um repositório do SVN é necessário configurá-lo n o Eclipse. Para isso é preciso ter a URL do repositório, o que pode ser conseguido com o administrador do repositório.  
Com a URL em mãos, abra a perspectiva do repositório SVN no Eclipse:

[](https://luizgustavoss.files.wordpress.com/2010/11/img1.png)

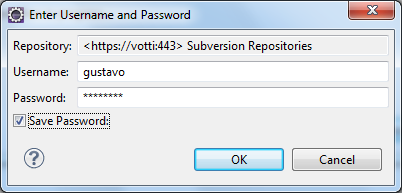
Clicando com o botão auxiliar do mouse, selecione a opção para adicionar um novo local:

[](https://luizgustavoss.files.wordpress.com/2010/11/img2.png)

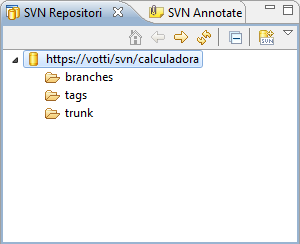
Adicione a URL do repositório de testes no campo indicado:

[](https://luizgustavoss.files.wordpress.com/2010/11/img3.png)

Pode ser necessário informar o usuário e senha para acesso. Esta informação pode ser conseguida com o administrador do servidor:

[](https://luizgustavoss.files.wordpress.com/2010/11/img4.png)

Após estes passos, o repositório será apresentado na perspectiva, conforme abaixo:

[](https://luizgustavoss.files.wordpress.com/2010/11/img5.png)

A estrutura sugerida pelo SVN possui três diretórios: branches, tags e trunk.

**/trunk** –  o diretório onde o desenvolvimento principal do projeto ocorre, ou seja, onde sempre está a versão mais recente e estável dos artefatos oficiais;  
**/branches** – o diretório onde pode-se criar ramificações da linha principal de desenvolvimento (trunk);  
**/tags** – o diretório onde se pode armazenar ramificações da linha principal que são criadas, e talvez destruídas, mas nunca modificadas.

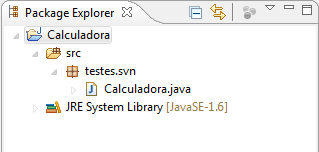
**Criando uma aplicação Java**

Crie, no C:\, um diretório chamado workspace, e nele crie dois diretórios chamados work1 e work2. Estes serão os workspaces usados para os testes.

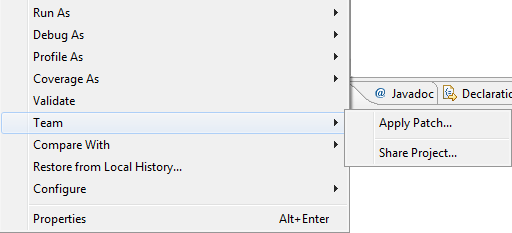
Abra o Eclipse apontando para o workspace work1. Após configurar o ambiente (adicionar um JDK apropriado) crie um projeto Java simples, chamado Calculadora. Neste projeto crie um pacote chamado testes.svn, e nele crie uma classe chamada Calculadora.java, com os seguintes métodos:

|  |  |
| --- | --- |
|  | public class Calculadora {    public BigDecimal somar(BigDecimal param1, BigDecimal param2) throws Exception{  return null;  }    public BigDecimal subtrair(BigDecimal param1, BigDecimal param2) throws Exception{  return null;  }    public BigDecimal multiplicar(BigDecimal param1, BigDecimal param2) throws Exception{  return null;  }    public BigDecimal dividir(BigDecimal param1, BigDecimal param2) throws Exception{  return null;  }  } |
|  |  |

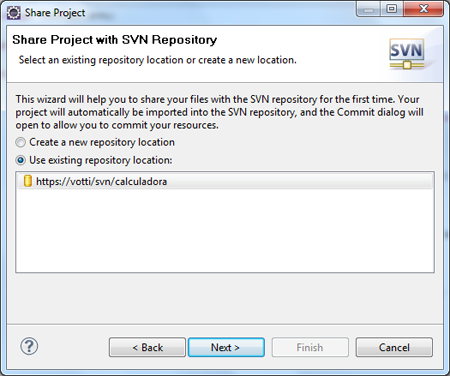
Esta é a estrutura do projeto após a criação da classe:

[](https://luizgustavoss.files.wordpress.com/2010/11/img6.png)

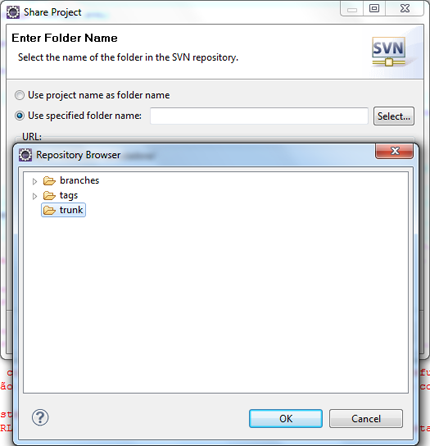
Vamos agora colocar o projeto no repositório que configuramos anteriormente. Clique com o botão auxiliar do mouse sobre o projeto, vá até a opção de menu “Team” e escolha a opção “Share Project”:

[](https://luizgustavoss.files.wordpress.com/2010/11/img7.png)

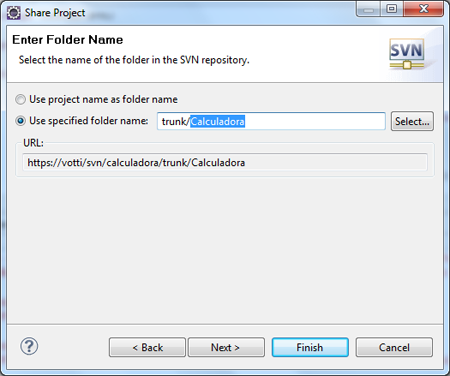
Na próxima tela, escolha a opção SVN e prossiga. Na tela seguinte, escolha o repositório configurado anteriormente e prossiga:

[](https://luizgustavoss.files.wordpress.com/2010/11/img8.png)

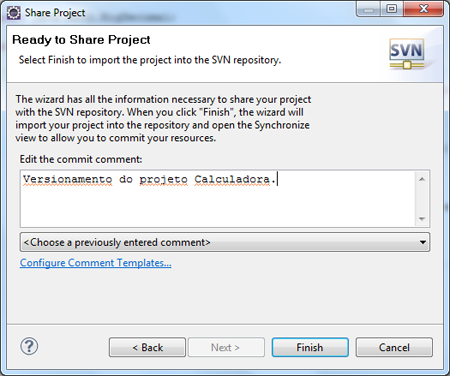
Na próxima tela, temos a opção de configurar o nome sobre o qual o projeto será armazenado no repositório, e sua localização. Marque a segunda opção, e clicando no botão “Select…” informe o diretório trunk:

[](https://luizgustavoss.files.wordpress.com/2010/11/img9.png)

Ao retornar da tela, esta deverá ser a configuração da tela:

[](https://luizgustavoss.files.wordpress.com/2010/11/img10.png)

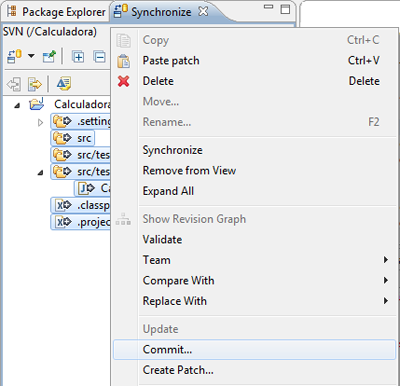
Prossiga, e na tela final digite uma mensagem inicial para o versionamento e confirme:

[](https://luizgustavoss.files.wordpress.com/2010/11/img11.png)

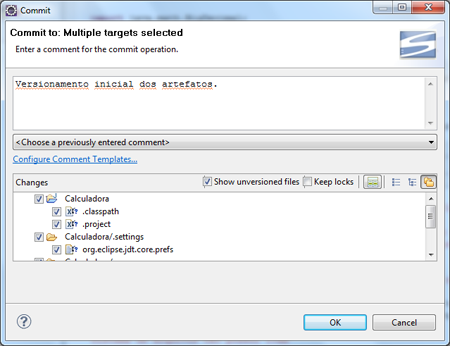
O seguinte comando será executado contra o servidor, adicionando o projeto ao repositório:

*mkdir –parents -m “Versionamento do projeto Calculadora.”* [*https://votti/svn/calculadora/trunk/Calculadora*](https://votti/svn/calculadora/trunk/Calculadora)  
*checkout* [*https://votti/svn/calculadora/trunk/Calculadora*](https://votti/svn/calculadora/trunk/Calculadora) *-r HEAD –depth=immediates –force*

Logo em seguida, será apresentada a perspectiva de sincronização, onde poderemos “commitar” o projeto no repositório, ou seja, enviar o conteúdo do projeto para o repositório, literalmente.  
As setas cinza sobre os arquivos e diretórios indicam que estes arquivos devem ser comitados, então selecione os mesmos e, clicando com o botão auxiliar do mouse sobre os itens selecionados, escolha a opção “commit”:

[](https://luizgustavoss.files.wordpress.com/2010/11/img12.png)

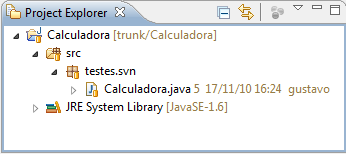
Na tela seguinte, inclua um comentário inicial e confirme:

[](https://luizgustavoss.files.wordpress.com/2010/11/img13.png)

Os seguintes comandos serão executados, indicando que os artefatos foram “commitados” com sucesso:

*add -N C:\workspace\work1\Calculadora\.settings*  
*A         C:/workspace/work1/Calculadora/.settings*  
*add -N C:\workspace\work1\Calculadora\src*  
*A         C:/workspace/work1/Calculadora/src*  
*add -N C:\workspace\work1\Calculadora\src\testes*  
*A         C:/workspace/work1/Calculadora/src/testes*  
*add -N C:\workspace\work1\Calculadora\src\testes\svn*  
*A         C:/workspace/work1/Calculadora/src/testes/svn*  
*add -N C:\workspace\work1\Calculadora\.settings\org.eclipse.jdt.core.prefs*  
*A         C:/workspace/work1/Calculadora/.settings/org.eclipse.jdt.core.prefs*  
*add -N C:\workspace\work1\Calculadora\.project*  
*A         C:/workspace/work1/Calculadora/.project*  
*add -N C:\workspace\work1\Calculadora\src\testes\svn\Calculadora.java*  
*A         C:/workspace/work1/Calculadora/src/testes/svn/Calculadora.java*  
*add -N C:\workspace\work1\Calculadora\.classpath*  
*A         C:/workspace/work1/Calculadora/.classpath*  
*commit -m “Versionamento inicial dos artefatos.” (8 paths specified)*  
*Adding         C:/workspace/work1/Calculadora/.classpath*  
*Adding         C:/workspace/work1/Calculadora/.project*  
*Adding         C:/workspace/work1/Calculadora/.settings*  
*Adding         C:/workspace/work1/Calculadora/.settings/org.eclipse.jdt.core.prefs*  
*Adding         C:/workspace/work1/Calculadora/src*  
*Adding         C:/workspace/work1/Calculadora/src/testes*  
*Adding         C:/workspace/work1/Calculadora/src/testes/svn*  
*Adding         C:/workspace/work1/Calculadora/src/testes/svn/Calculadora.java*  
*Transmitting file data …*

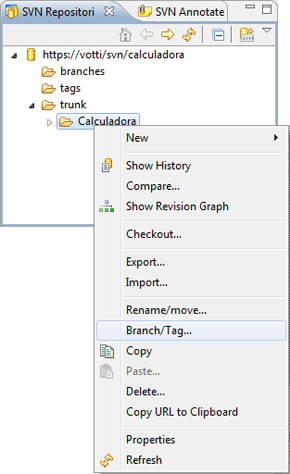
Voltando à perspectiva Java, podemos perceber que o projeto está no repositório, pela notação à frente do nome do projeto, e podemos ver ainda qual usuário realizou o “commit” da classe Calculadora:

[](https://luizgustavoss.files.wordpress.com/2010/11/img14.png)

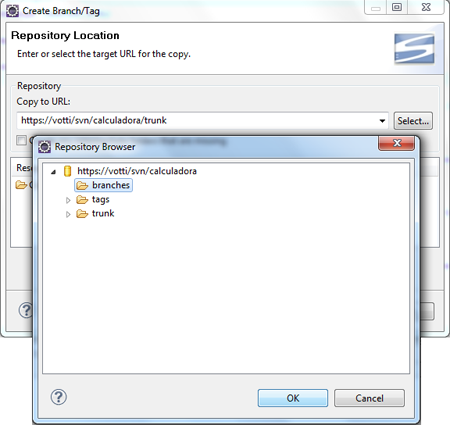
Pois bem, temos o projeto versionado no trunk. Como dito anteriormente, no trunk deveriam ser mantidos os códigos estáveis de um projeto. Para a realização de testes ou implementações que podem ser instáveis por um longo período, pode ser preferível criar um branch. No branch colocamos uma cópia do projeto que está no trunk, mas sem perder a referência. A equipe que trabalha no código do branch pode ainda manter uma sincronização com o trunk, trazendo deste as atualizações pertinentes, evitando que o código do branch fique desatualizado com relação ao trunk,  mas evitando que o código instável do branch influencie negativamente o código do trunk.  
Para nosso tutorial, vamos supor que a implementação dos métodos da classe Calculadora serão  realizados no branch. No dia-a-dia tal implementação poderia ser feita no trunk, mas como nosso objetivo é aprender a criar branches, vamos supor que esta seja uma necessidade real.

**Criando um Branch**

Para criar um branch vá até a perspectiva do repositório SVN e, expandindo os diretórios, clique com o botão auxiliar do mouse sobre o projeto Calculadora, e no menu de contexto escolha a opção “Branch/Tag…”:

[](https://luizgustavoss.files.wordpress.com/2010/11/img15.png)

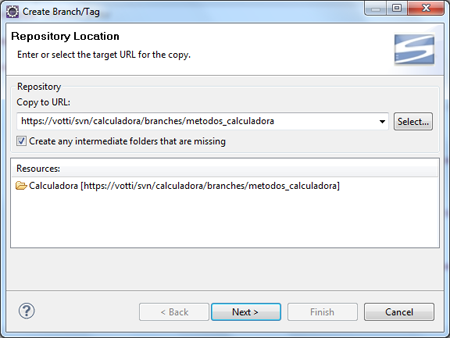
Na tela seguinte precisamos informar o diretório destino, no repositório, para o projeto. Como estamos criando um branch, o diretório destino será o branch. Porém, como podem existir vários branches de um mesmo projeto ao mesmo tempo, é conveniente criar um subdiretório com um nome que identifique o objetivo do branch.  
Para fazer isso, selecione o branch através do botão “Select…”:

[](https://luizgustavoss.files.wordpress.com/2010/11/img16.png)

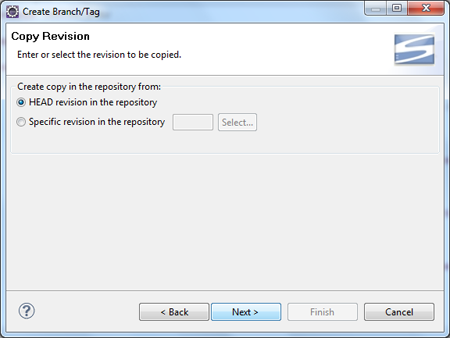
Ao voltar para a tela anterior, dê um nome ao subdiretório. Isso pode ser feito adicionando o nome à URL resultante do processo anterior. Para este exemplo, o diretório será “metodos\_calculadora”. Eis a URL final:

<https://votti/svn/calculadora/branches/metodos_calculadora>

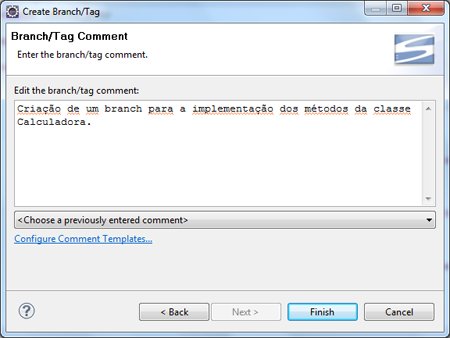
Lembre-se de marcar na tela a opção que permite que o diretório adicional seja criado (*Create any intermediate folders that are missing*.):

[](https://luizgustavoss.files.wordpress.com/2010/11/img17.png)

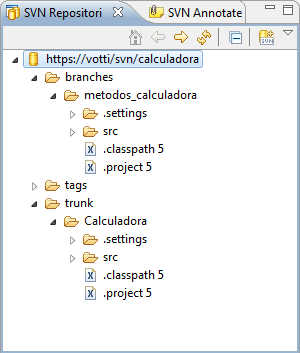
Na próxima tela podemos escolhar qual revisão queremos para o branch. No nosso caso vamos escolhar a opção HEAD, que indica que será enviada para o branch a versão mais atual existente no trunk. Porém, se necessário, uma versão específica poderá ser selecionada:

[](https://luizgustavoss.files.wordpress.com/2010/11/img18.png)

Na tela seguinte adicione um comentário pertinente, e faça a confirmação:

[](https://luizgustavoss.files.wordpress.com/2010/11/img19.png)

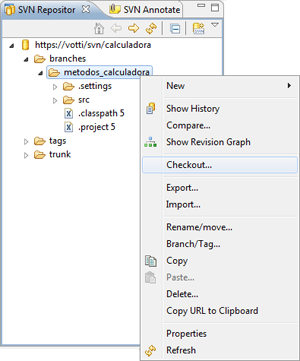
Se formos agora à perspectiva do repositório, poderemos ver que o conteúdo do trunk foi copiado para o branch configurado:

[](https://luizgustavoss.files.wordpress.com/2010/11/img20.png)

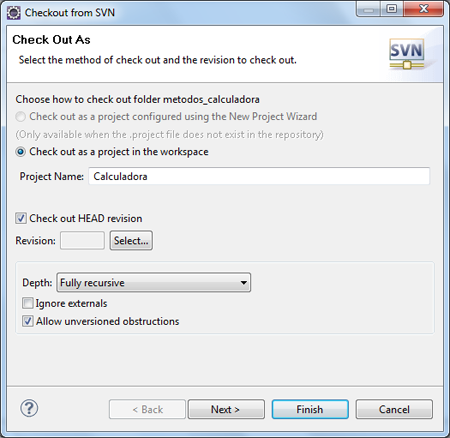
Veja que até o momento copiamos o conteúdo do projeto para o trunk e a partir do trunk criamos um branch para realizar a implementação, mas no nosso workspace temos apenas o projeto que está “ligado” ao trunk. Isso significa que se fizermos qualquer modificação, e fizermos um commit, estas modificações serão enviadas ao trunk.

**Realizando modificações no branch**

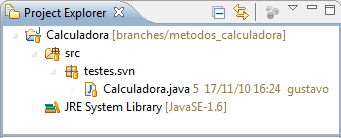
Se quisermos realizar modificações no branch criado, teremos que configurar um projeto que esteja “ligado” a ele.  
Para fazer isso, vamos reiniciar o Eclipse e como workspace vamos selecionar o diretório work2, criado no início do tutorial. Assim que tiver aberto o Eclipse no novo workspace faça as configurações necessárias (configuração de uma JDK padrão, por exemplo).  
Abra a perspectiva do SVN e, assim como no início do tutorial, insira uma novo local de repositório, informando a URL do repositório de exemplo.  
Ao terminar o processo de configuração do repositório, abra o diretório branch. Clique com o botão auxiliar do mouse sobre o diretório “metodos\_calculadora”, e no menu de contexto escolha a opção “Checkout…”:

[](https://luizgustavoss.files.wordpress.com/2010/11/img21.png)

Com isso vamos copiar o conteúdo do branch para o repositório, para trabalhar nele.  
A próxima tela apresenta os dados para o projeto qe será criado:

[](https://luizgustavoss.files.wordpress.com/2010/11/img22.png)

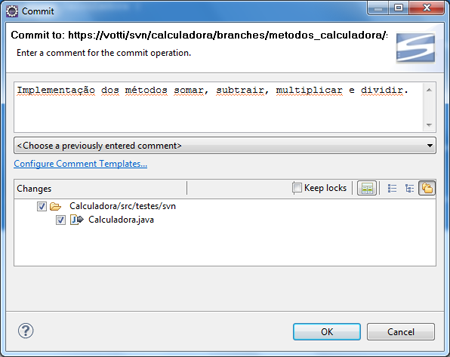
Repare que o nome do projeto foi recuperado e dado como sugestão. Também há a possibilidade de escolher uma revisão específica, ou então a última (HEAD). Vamos manter as configurações sugeridas.  
Na próxima tela será sugerido como workspace o workspace atual (work2). Mantenha o valor sugerido e prossiga, clicando em “Finish”.  
Ao voltar para a perspectiva Java, é possível verificar que o projeto foi configurado, e que referencia o branch criado:

[](https://luizgustavoss.files.wordpress.com/2010/11/img23.png)

Agora vamos simular a situação real de ter que alterar o código em ambos os locais, trunk (work1) e branch (work2) e ter de mantê-los sincronizados.  
Vamos começar codificando a classe Calculadora no workspace work2, que referencia o branch, no qual estamos:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | public class Calculadora {        public BigDecimal somar(BigDecimal param1, BigDecimal param2) throws Exception{          return param1.add(param2);      }        public BigDecimal subtrair(BigDecimal param1, BigDecimal param2) throws Exception{          return param1.subtract(param2);      }        public BigDecimal multiplicar(BigDecimal param1, BigDecimal param2) throws Exception{          return param1.multiply(param2);      }        public BigDecimal dividir(BigDecimal param1, BigDecimal param2) throws Exception{          return param1.divide(param2);      }  } |

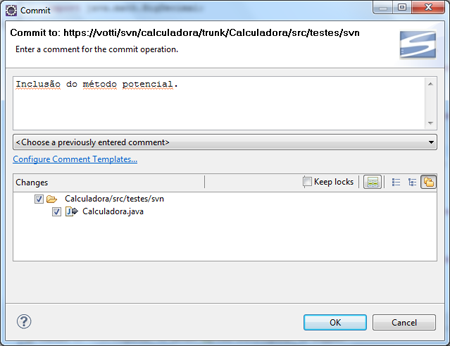
Depois de codificar a classe, faça o commit das modificações no branch, descrevendo as modificações:

[](https://luizgustavoss.files.wordpress.com/2010/11/img24.png)

É extremamente importante realizar o commit de todas as alterações do workspace no brach/trunk antes de realizar qualquer operação de merge. Aliás, é uma exigência!  
Vamos agora realizar uma modificação no projeto do trunk, para ver como recuperar esta modificação, depois, no branch. Mude de workspace, voltando para a workspace work1.  
Você verá que, como esperado, os métodos na classe Calculadora do workspace work1 estão como antes, sem implementação.  
Adicione então, à classe Calculadora, um método a mais:

|  |  |
| --- | --- |
| 1  2  3 | public BigDecimal potencial(BigDecimal param1, BigDecimal param2) throws Exception{  return null;  } |

Faça o commit da modificação, adicionando um comentário pertinente:

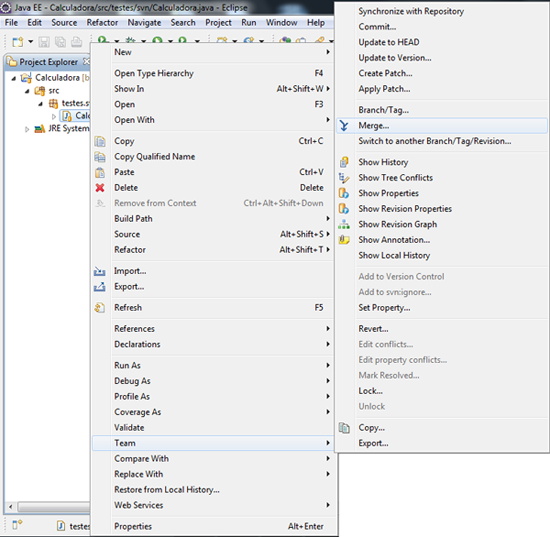
[](https://luizgustavoss.files.wordpress.com/2010/11/img25.png)

Retorne para o workspace work2.

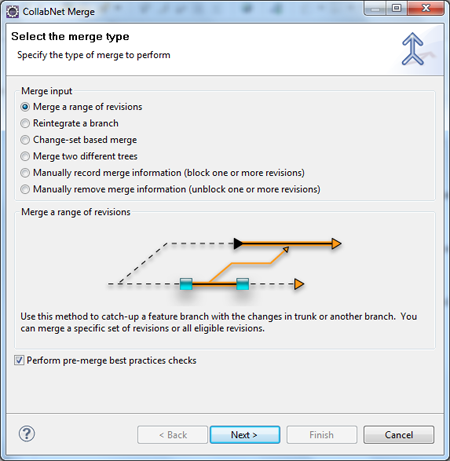
**Realizando um merge do trunk para o branch**

Agora, no workspace work2, vamos realizar um merge para receber as atualizações do trunk, ou seja, o novo método adicionado à classe Calculadora.  
Este procedimento deveria ser, como comentado anteriormente, uma prática realizada frequentemente para que o branch não fique muito tempo desatualizado em relação ao trunk. Isto minimiza problemas de conflito na hora de mover para o trunk as modificações do branch, procedimento este que realizaremos depois.

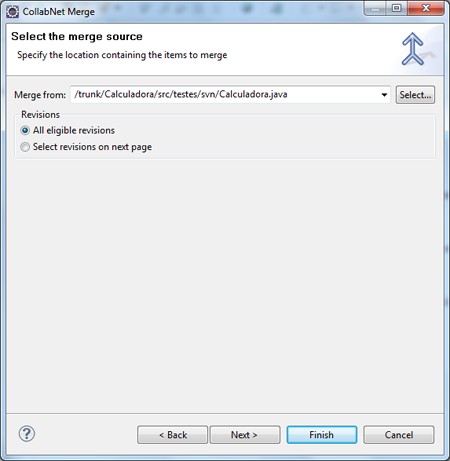
Selecione a classe Calculadora no projeto, e clicando com o botão auxiliar do mouse sobre ela escolha a opção de “Merge…” no menu “Team”:

[](https://luizgustavoss.files.wordpress.com/2010/11/img26.png)

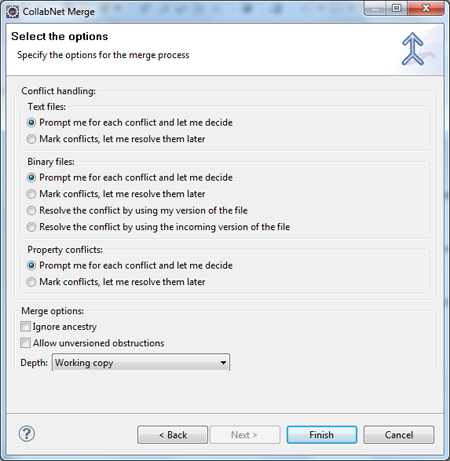
A tela seguinte nos apresenta algumas opções para a operação de merge:

[](https://luizgustavoss.files.wordpress.com/2010/11/img27.png)

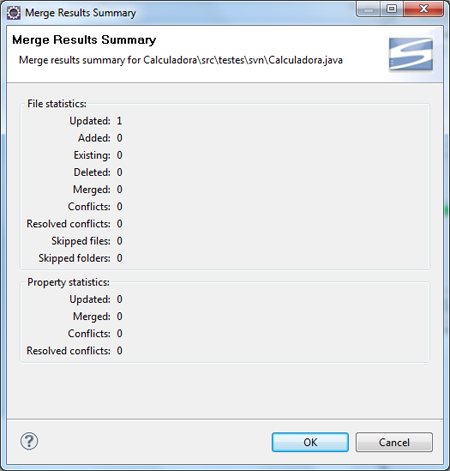
Para este tutorial as duas primeiras opções são as que interessam.  
Para esta operação que realizaremos agora, mantenha selecionada a primeira opção. Esta opção é usada para capturar mudanças que foram realizadas no trunk ou outro branch, e trazê-las ao branch no qual estamos trabalhando. Confirme a operação, clicando em “Next”:

[](https://luizgustavoss.files.wordpress.com/2010/11/img28.png)

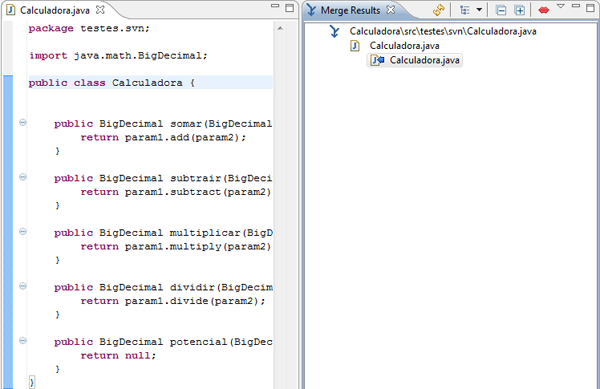
A próxima tela apresenta uma sugestão do artefato a partir do qual desejamos realizar o merge, ou seja, a classe Calculadora  presente no projeto do trunk. Neste ponto você poderia apontar para a mesma classe presente em algum outro branch, caso existisse, ou seja, não precisa ser necessariamente um merge a partir do trunk. Mas no nosso caso manteremos a sugestão, pois realmente precisamos realizar o merge com o trunk.  
Na próxima tela são apresentadas opções para a operação. Basicamente, as sugestões marcadas por padrão são melhores, pois indicam que a cada problema encontrado, o usuário deverá ser consultado sobre o que fazer. Mantenha as sugestões apresentadas por padrão:

[](https://luizgustavoss.files.wordpress.com/2010/11/img29.png)

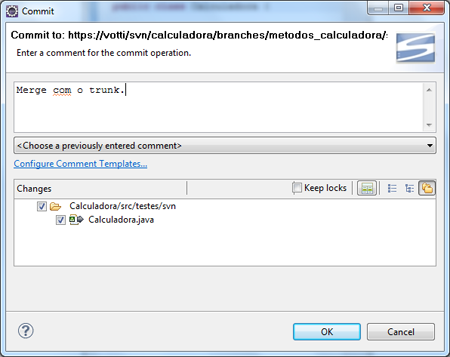
Ao confirmar, a seguinte tela é apresentada, indicando que uma atualização foi encontrada:

[](https://luizgustavoss.files.wordpress.com/2010/11/img30.png)

Após a conclusão da operação, você verá que o método adicionado na classe Calculadora do trunk apareceu na classe Calculadora do branch:

[](https://luizgustavoss.files.wordpress.com/2010/11/img31.png)

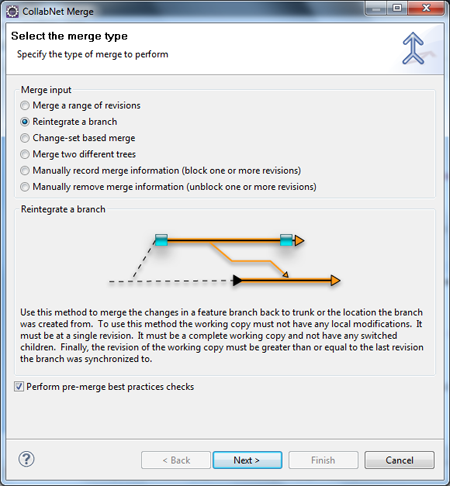
Vale lembrar que, apesar de termos realizado esta operação para apenas uma classe, ela pode ser realizada para vários artefatos ao mesmo tempo.  
Agora que a classe foi modificada, faça o commit das alterações no branch, informando um comentário pertinente para a mudança:

[](https://luizgustavoss.files.wordpress.com/2010/11/img32.png)

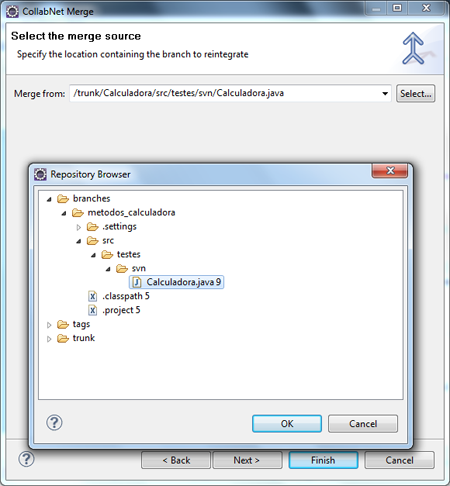
**Realizando um merge do branch para o trunk**

Depois do trabalho realizado no branch, precisamos reintegrá-lo ao trunk, uma hora ou outra.  
Muitas vezes o trabalho em um branch pode ser tão rápido quanto algumas horas, ou longo a ponto de levar vários dias ou meses. O tamanho do trabalho a ser realizado é que irá determinar isso.  
Independente de quanto tempo ele leve, o fato de realizar periodicamente o merge do trunk com o branch, como feito anteriormente, deve minimizar os problemas de conflitos quando chegar a hora de trazer o branch para o trunk, uma vez que, idealmente, os conflitos existentes já terão sido resolvidos no momento em que se levou as mudanças do trunk para o branch.  
Vamos agora trazer o que foi feito no branch para o trunk. Mude para o workspace work1.  
Selecione a classe Calculadora no projeto, e clicando com o botão auxiliar do mouse sobre ela escolha a opção de “Merge…” no meu “Team”

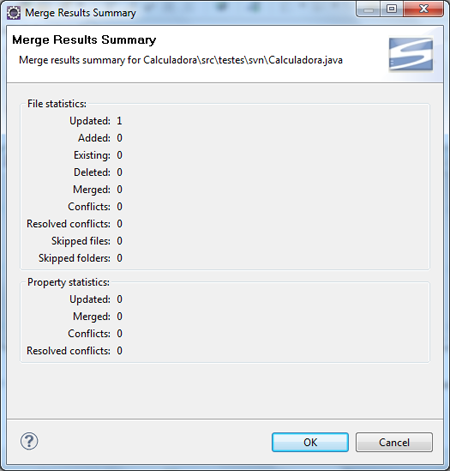
Na tela que aparece em seguida, mantenha selecionada a segunda opção, que nos permitirá trazer as mudanças do branch para o trunk:

[](https://luizgustavoss.files.wordpress.com/2010/11/img33.png)

Na tela seguinte, na qual informamos a partir de onde será feito o merge, selecione a classe Calculadora do branch, como indicado:

[](https://luizgustavoss.files.wordpress.com/2010/11/img34.png)

Na próxima tela confirme as opções oferecidas como padrão, e ao finalizar a tela seguinte será apresentada, indicando que uma atualização será realizada:

[](https://luizgustavoss.files.wordpress.com/2010/11/img35.png)

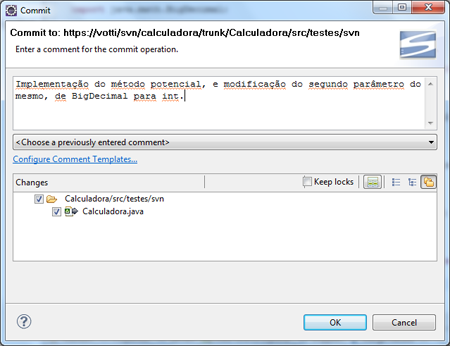
Confirme clicando no botão “OK”. Ao término da operação as atualizações do branch serão aplicadas ao trunk.

**Lidando com conflitos entre o trunk e o branch**

Até agora realizamos o merge entre o trunk e o branch sem nenhum problema de conflito, mas isto está longe de ser a realidade do dia-a-dia de trabalho das equipes de desenvolvimento. Na verdade os conflitos são muito comuns. As boas práticas procuram melhorar a produtividade das equipes, manter a estabilidade dos repositórios (principalmente do trunk) e minizar os conflitos, mas eliminar os conflitos, principalmente em um ambiente onde muitos desenvolvedores atualizam os mesmos arquivos, é algo muito difícil. Por esta razão é bom se acostumar com estes conflitos, e se acostumar a resolvê-los também.  
Para simular este cenário, vamos modificar o último método adicionado à classe Calculadora. Perceba que ele ainda não foi implementado, e que recebe dois parâmetros. Na verdade precisamos modificar um de seus parâmetros, e também precisamos implementá-lo.  
Ainda no workspace work1 modifique o método como abaixo:

|  |  |
| --- | --- |
| 1  2  3 | public BigDecimal potencial(BigDecimal param1, int param2) throws Exception{  return param1.pow(param2);  } |

Realize o commit das modificações, e sim, coloque um comentário! (não é à toa que estou batendo nesta tecla):

[](https://luizgustavoss.files.wordpress.com/2010/11/img36.png)

Agora mude para o workspace work2 e também faça estas modificações no métod, exatamente como no trunk, realizando o commit ao final.  
Depois de realizado o commit, faça um merge, trazendo do trunk as modificações para o branch.  
Apesar de termos alterado o mesmo método em ambos os locais, nenhuma atualização é encontrada, uma vez que as modificações foram exatamente as mesmas, ou seja, nenhum conflito é encontrado.  
Vamos agora simular um conflito real, quando o mesmo local é modificado, porém com valores diferentes.

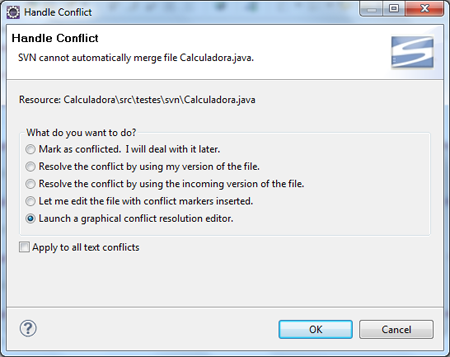
Mude para o workspace work1 e altere o método dividir conforme apresentado abaixo, realizando o commit ao final:

|  |  |
| --- | --- |
| 1  2  3 | public BigDecimal dividir(BigDecimal param1, BigDecimal param2) throws Exception{  return param1.divide(param2, BigDecimal.ROUND\_UP);  } |

Agora mude para o workspace work2 e altere o método dividir conforme apresentado abaixo, realizando o commit no final:

|  |  |
| --- | --- |
| 1  2  3 | public BigDecimal dividir(BigDecimal param1, BigDecimal param2) throws Exception{  return param1.divide(param2, BigDecimal.ROUND\_HALF\_EVEN);  } |

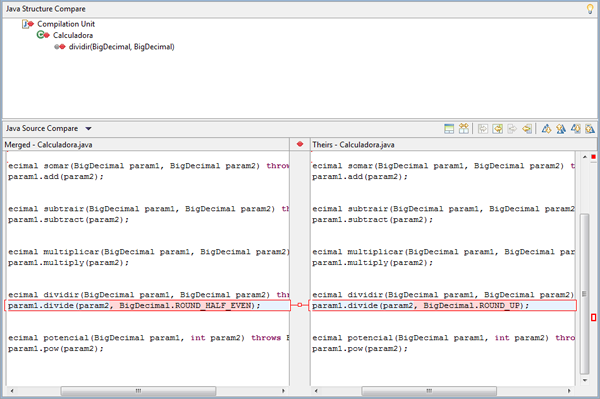
Agora tente novamente trazer do trunk para o branch as modificações na classe Calculadora e veja o resultado.

[](https://luizgustavoss.files.wordpress.com/2010/11/img37.png)

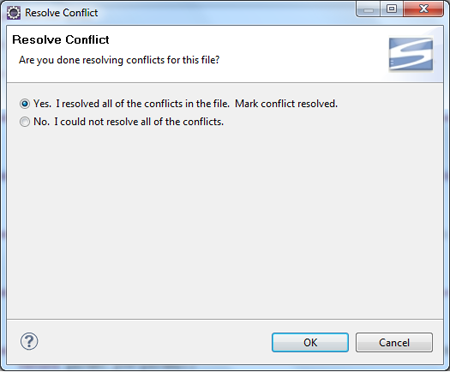
Desta vez há um conflito, pois o método foi modificado em mais de um local (repositório), e no mesmo ponto.  
As opções apresentadas na tela que é apresentada são:

– Marcar como em conflito. Resolverei mais tarde.  
– Resolver o conflito usando a minha versão do arquivo.  
– Resolver o conflito usando a versão do arquivo que está chegando.  
– Me deixe editar o arquivo com os marcadores de conflito inseridos.  
– Abrir um editor gráfico de solução de conflitos.

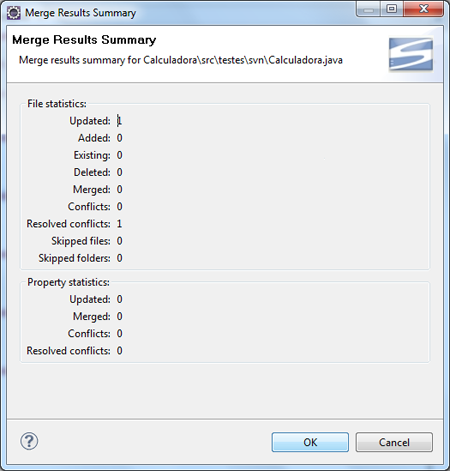
Particularmente recomendo lidar o quando antes com os conflitos, portanto, escolha a última opção e confirme. O Editor apresenta os conflitos a serem resolvidos:

[](https://luizgustavoss.files.wordpress.com/2010/11/img38.png)

Suponhamos que nossa versão do branch está correta. Então feche o editor, e uma mensagem será apresentada, perguntando se o conflito foi resolvido:

[](https://luizgustavoss.files.wordpress.com/2010/11/img39.png)

Marque que o conflito foi resolvido (primeira opção) e confirme:

[](https://luizgustavoss.files.wordpress.com/2010/11/img40.png)

Na tela que indica que uma atualização será feita faça a confirmação. Depois realize o commit da classe Calculadora no branch.  
Desta forma, cada implementação do método, no trunk e no branch, ficou de uma forma. Se quiséssemos deixá-las iguais, bastaria ter aceito a diferença vinda do trunk (caso esta fosse a correta) ou então, a partir do trunk, puxar as alterações do branch e aceitá-las.

**Conclusão**

Neste tutorial foram apresentados os passos necessários para se configurar um repositório SVN no Eclipse e para se adicionar um projeto ao repositório.  
Foram apresentados também os passos necessários para se criar um branch, como realizar o merge entre o branch e o trunk (em ambas as direções) e como resolver conflitos de versionamento.  
Com isso concluímos este tutorial!  
Espero que tenha sido de bom proveito, e que com o que foi apresentado aqui você possa melhorar sua rotina de trabalho aplicando mais algumas boas práticas no seu dia-a-dia de utilização do SVN.