svn 版本主干分支管理详解

1.	创建分支的意义	2
	创建分支的原理	
	如何创建分支	
	版本的合并	
	4.1. 合并的类型	
	4.1.1. 合并一个范围的版本	
	4.1.2. 复兴合并	
	4.1.3. 合并两个不同的树	4
	4. 2. 合并的方式	5
	4. 2. 1. 主干合并到分支	5
	4. 2. 2. 分支合并到主干	

1. 创建分支的意义

创建分支的意义,比如接到一个子项目,需要做大量代码修改,如中山交警项目,而主项目也需要继续修改,如修改 bug。如果不创建分支,两个项目之间会互相影响,影响最大的就是测试和版本的发布。所以需要从主项目(trunk)中创建分支版本(branch),从而隔离互相的影响。

2. 创建分支的原理

在 svn 中,创建分支,实际上就是一个版本 copy,但不是真正意义上的 copy,而是在仓库中以软连接的方式关联。版本发生更改版本号会增加,比如:目前主干版本是 100,分支版本是 101,主干中增加一个文件,版本为 102,分支中再增加一个文件,版本就为 103 了。两边的版本号是一套,不会重复。

3. 分支策略

保留

4. 如何创建分支

下面以 TortoiseSVN 为例,myeclipse 中也相似。

首先要到 working copy 目录下找到要创建分支的项目,如图 3.1 所示打开创建分支的界面。

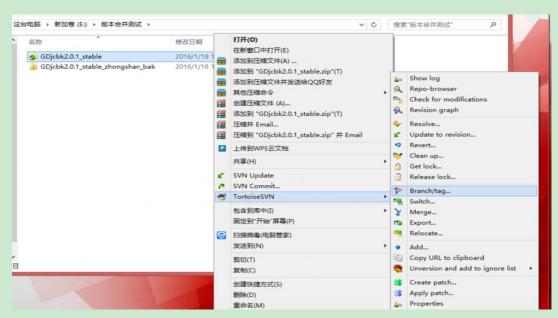


图 3.1 打开创建分支界面 如图 3.2,指定目标路径,版本号就选 HEAD,点击 ok 完成创建

To path:		and the second s	
	/branches/GDjcbk2.0.1	_stable_zhongshan	<u> </u>
Destination URL: https://lenovo-pc4:4	444/svn/MySvnTest/branches/	GDjcbk2.0.1_stable_zhong	ıshan
			? I .
Log message			
Recent mess	sages		
	repository from:		
HEAD revision in Specific revision Working copy	n the repository n in repository	52	Show Log
HEAD revision in Specific revision Working copy	n the repository n in repository n for these externals:	52	Show Log
HEAD revision in Specific revisior Working copy Set explicit revisior	n the repository n in repository n for these externals:	52 Fixed at r	
HEAD revision in Specific revision Working copy Set explicit revision Check: All No	n the repository n in repository n for these externals: ne		

图 3.2 创建分支界面

完成创建后,我们就可以将该分支 check out 下来开发了

	Checkout		
Repository			
URL of repository:			
vo-pc4:444/svn/MySvnTe	est/branches/GDjcbk2.	0.1_stable_zhongsha	<u> </u>
Checkout directory:			
E:\版本合并测试\GDjcbk2.0.	1_stable_zhongshan		
Multiple, independent work	ing copies		
Checkout Depth			
Fully recursive			~
Omit externals		Choose iter	m <mark>s</mark>
Revision			
HEAD revision			
○ Revision		Show	log

5. 版本的合并

5.1. 合并的类型

5.1.1. 合并一个范围的版本

此类型应用最为广泛,主要是把分支中的修改合并到主干上来。在主干上点击右键选择合并,然后选择合并类型:合并一个范围的版本。合并的源 URL 填写的是要合并的分支的 URL,待合并的版本范围如果为空,则指的是合并分支上所有的版本,即自从分支创建以来到分支当前最新版本的所有演变。如果只是选择其中一个版本,或者几个版本,那么就表示只是将制定的 n 个版本的变化合并到主干上。如果只是选择其中一个版本,那么表示只是选择那个版本的修改,之前或之后的修改将不被采纳。

5.1.2. 复兴合并

复兴合并可以理解为是第一种合并类型的一种特例,在复兴合并中,主干可以理解为是 自从开创分支之后没有任何修改,而分支是经过修改的,而且合并中分支是没有版本选择的。 经过复兴合并,分支中所有的修改都会合并到主干中,合并的结果将使得分支和主干一模一 样,从而可以删除分支。

5.1.3. 合并两个不同的树

此类型与前两种类型不同,第一种类型可以选择分支合并的版本,主干不能选择版本;第二种类型是主干和分支都不能选择合并的版本;而这种类型则是无论是主干还是分支都可以选择合并的版本,即可以选择过去的一个主干版本与分支的某个版本进行合并。合并的时候以选择的分支版本为主,如果选择的主干版本与分支版本有不同的地方,合并时主干部分将被放弃。

5.2. 合并的方式

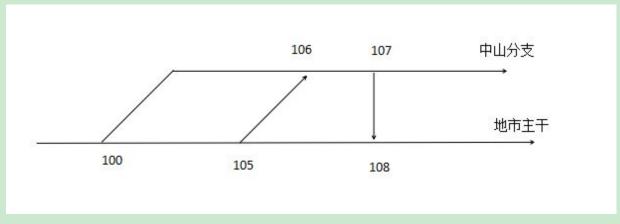


图 4.1 分支示意图

例如现在有一个主干和分支在并行开发,当到了 105 版本时,需要将分支第一次合并到主干,需要以下步骤:

首先要将主干上的更改合并到分支上,若有冲突,需要在分支上解决。在分支上提交完成后再合并回主干。

5.2.1. 主干合并到分支

我们就用上面提到的合并类型中的第一种,将主干特定的版本合并到分支上,这样即使有不同的文件也不会被删掉。

步骤如下:

右键点击我们的分支版本 GDjcbk2.0.1_stable_zhongshan -->Tortoies-->Merge ,如图 4.2。

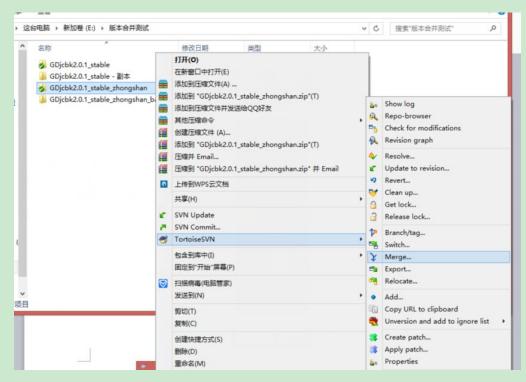


图 4.2 打开合并界面

打开合并界面如图 4.3。

点击第一个,合并范围中的版本,Next;

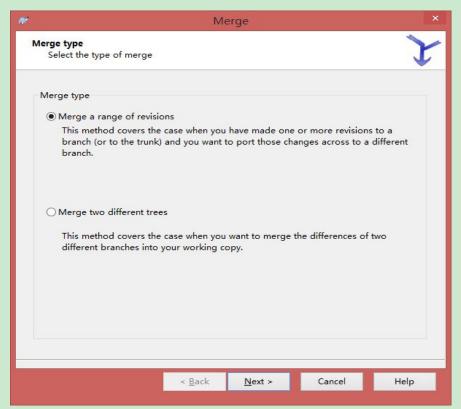


图 4.3 合并界面 1

图 4.4 中目标 URL 选择主干版本, 然后选择要合并进来的目标版本。

选择版本范围最简单的方法是,点击**显示日志**,列出最近的修改和日志。如果你要合并单个版本的修改,直接选取那个版本。如果你要合并多个版本,就选择范围(使用通常的 Shift 键)。点击**确认**,就会为你填写要合并的版本号列表。

100000	ge		
lerge revision range Select the revisions to merge			
URL to merge from			
https://lenovo-pc4:444/svn/MySvnTest/true	nk/GDjcbk2.0.	1_stable	·
Revision range to merge			
all revisions			
specific range			Show log
Reverse merge			
Use the log dialog to select the revisions you merge, separated by commas. A revision rang			
Example: 4-7,9,11,15-HEAD@pegrevision			
To merge all revisions (reintegrate/automatic	merge), leave	the box em	pty.
To merge all revisions (reintegrate/automatic	merge), leave	the box em	pty.
To merge all revisions (reintegrate/automatic		the box em	pty. Show log
To merge all revisions (reintegrate/automatic		the box em	

图 4.4 合并界面 2

到最后一个界面,图 4.5,全部按默认设置。点击 Merge,建议合并时,使用 test merge来查看有多少冲突,如果冲突较多,可分多次合并,减少合并的复杂性

合并完成后可能会有冲突,编辑完冲突后,提交代码,希望你在提交日志中注明最后一个合并的版本号。这时,你可以在工作副本上使用显示日志对话框跟踪日志。记住,我们将版本号视作修改集,你应该使用最后合并的版本之后的版本作为本次合并的开始版本。例如,上次你已经合并了版本 37 到 39,那么本次合并你应该从版本 40 开始。

完成。

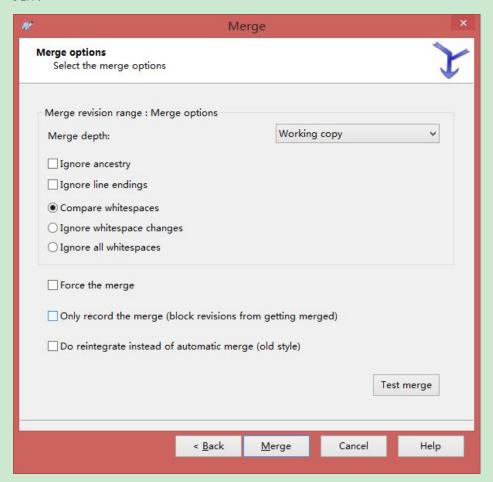


图 4.5 合并界面 3

5.2.2. 分支合并到主干

分支合并到主干是一样的操作,只是把 URL 换成分支版本,这里的情况算是复兴合并,图 4.4 中选择 all revisions, Merge 后如图 4.6 所示,然后赶紧提交,删除掉分支,大功告成。



图 4.6 合并完成

6. 分支使用的几点原则及注意事项

- ▶ 使用有意义的分支名称:有意义的分支名称方便管理,便于开发人员选择正确的分支 存取代码。
- ▶ 使用多分支而不是冻结代码:例如使用主线集成开发分支的变更,而不是在主线上开发。否则版本发布时需要冻结主线代码进行验证和修复,浪费开发时间。
- ▶ 尽早并经常集成:分支中新的变更一旦准备好,就应该进行变更合并
- ▶ 为不兼容的开发创建分支:使用分支作不兼容开发,能够很好的避免相互影响,提高质量和开发速度。
- 创建适当数量的分支:不要期望分支可以解决项目的所有问题,需要的情况下才使用分支。避免分支数量超出项目的管理能力。
- ➤ 只对完整的变更做合并:不要在变更没有全部完成的时候将部分变更合并到主线中
- ▶ 一般一个分支解决完问题后就删除了,若有其他问题再另建分支,尽量避免一个分支合并多次。