精灵代码分支管理

# 标准分支管理

Release

Trunk

Branch

V1.0

V1.1

V2.0

FixBranch

V1.1\_fix

tag

tag

tag

merge…

merge

merge

merge

reintegrate

reintegrate

reintegrate

merge

reintegrate

tag

branch

branch

branch

branch

V1.0\_release

V1.1\_release

V1.1\_fixed\_release

V2.0\_release

* Branch 开发以及一轮测试分支；
* Trunk主干分支，测试人员进行测试的专用分支，除非是修复测试人员提出的错误，否则开发人员不能修改该分支；
* Release发布专用的分支，不能进行任何修改；
* FixRelease紧急bug修复分支；
* branch操作版本开始，从trunk拉出开发分支；
* merge操作版本开发过程中，或定期或不定期多次从trunk进行代码同步到当前开发分支（至少在reintegrate操作之前进行一次merge）；
* reintegrate操作从分支将代码合并到trunk，通常合并后Branch上的分支需要进行冻结，之后的bug修复在trunk进行。虚线表示可以不进行；
* tag操作测试完成后，将从trunk拷贝代码（tag）到发布分支，并根据这个发布分支进行发布；
* 注意在多版本并行开发合并到trunk时，需要明确版本的上线先后顺序，先上先合；

# 精灵分支管理

由于系统部署系统原因，需要对以上分支管理模型进行一些修改

Trunk

Baseline

Branch

V1.0

V1.1

V2.0

FixBranch

V1.1\_fix

merge

merge

merge

merge…

merge

merge

merge

reintegrate

reintegrate

reintegrate

merge

merge

reintegrate

branch

branch

branch

branch

V1.0\_release

V1.1\_release

V1.1\_fixed\_release

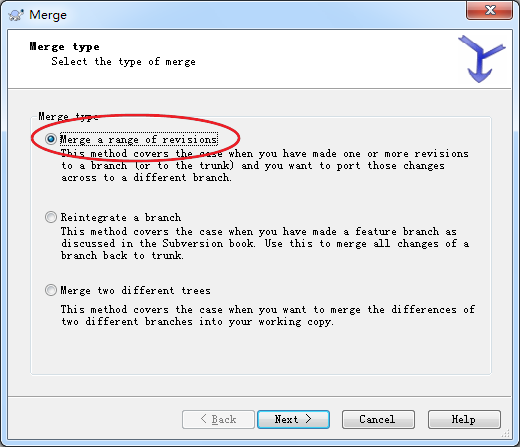
V2.0\_release

merge

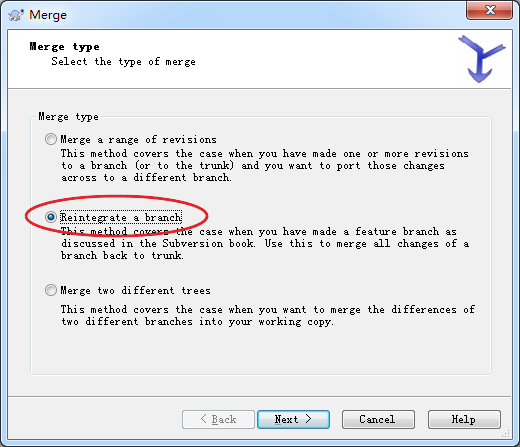
* Branch 开发以及一轮测试分支；
* Baseline开发基线分支，主要是用于版本开始时从这里拉取分支代码，以及开发完成后进行测试的主要分支，除非是修复测试人员提出的错误，否则开发人员不能修改该分支；；
* Trunk发布主干分支，在没有开发任务在进行时，baseline和trunk代码应该是保持一致的。最好不要在这个分支进行直接代码修改；
* FixRelease紧急bug修复分支，类似Branch但是FixRelease测试后将直接合并Trunk而不是Baseline，之后根据需求决定是否将紧急bug的修复从Trunk同步（merge操作）到Baseline；
* branch操作版本开始，从Baseline拉出开发分支；
* merge操作版本开发过程中，或定期或不定期多次从Baseline进行代码同步到当前开发分支（至少在reintegrate操作之前进行一次merge）。此外，Baseline和Trunk之间的代码同步也是使用merge操作（不能是reintegrate操作）；
* reintegrate操作从分支将代码合并到Baseline，通常合并后Branch上的分支需要进行冻结，之后的bug修复在Baseline进行。虚线表示可以不进行；
* 注意在多版本并行开发合并到Baseline时，需要明确版本的上线先后顺序，先上先合；

# SVN 操作

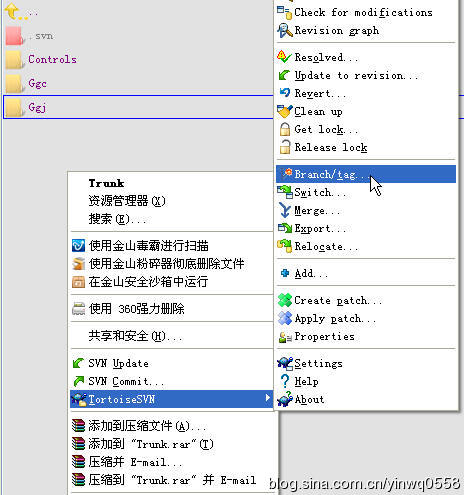
* 从baseline合并到branch



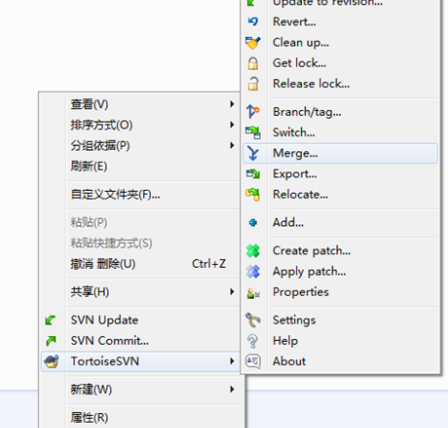
* 从branch到baseline



* 拉分支或打tag



* 切换分支



# 命名规范

分支命名：v版本号-时间-说明

其中时间和说明可不要（最好加上时间，时间格式yyyyMMdd）

例如：v2.7.1-20160219-workreport v2.7.1-20160219 v2.7.1

修复分支命名：需要修复的分支名+fixed后缀

例如：v2.7.1-20160219-workreport-fixed

# 建议本地开发分支管理

1. 应该有一个固定的baseline的workspace，比如workspace\_qiye\_baseline；
2. 最好固定一个branches的workspace，比如workspace\_qiye\_branch。每次开发新版本，切换到新的分支（使用svn switch，或者删除当前workspace的分支checkout新分支）；
3. 从baseline到分支代码合并使用merge；
4. 尽量不定期的将baseline代码merge到分支，但合并到baseline之前必须要进行一次branches到baseline的merge操作；
5. 一轮测试通过后才能将代码合并到baseline，并且必须使用reintegrate；
6. 上了baseline之后，尽量在baseline上进行修改，分支丢弃。若还需要将分支代码合并baseline，则只能使用merge；