Conflict detection and resolution

Since module version:

1.4.0

This page is based on 'Conflict detection and resolution - research'.

Introduction

Conflict scenarios:

- Update-update
  Occurs when the same object was updated at more than one node.
- Update-delete
  Occurs if a row was updated at one node, but the same row was deleted at another node.

Conflict Detection

Conflict detection is based on hash codes. Every object has its own Sync hash code which can be generated basing on that object. During pull or push operation a child instance compares hash codes of a local object, parent's object and recently saved parent's object. The hash code of recently saved parent's object is persisted in the database. We have to establish some shortcuts before we explain when the conflict is detected:

- #C : a hash code of the child instance object
- #P : a hash code of the parent instance object
- #lastP : a last saved hash code of the parent instance object.

The conflict is detected when #lastP != NULL && #lastP != #P && #lastP != #C.

Note! We have added a new row in a database sync_parent_object_hashcode table for each object pulled from its **Parent** instance. We keep only the most current hash code of each object.

Conflict Resolution

Conflict resolution can be handled automatically or manually. After the conflict is detected it is passed to object/service which implements MergeBehaviour interface. The MergeBehaviour class defines a strategy of dealing with conflicts automatically. (In fact, you can add your own implementation. All hints can be found here.) All unresolved conflicts will be saved in order to be solved by the users.

If the same object causes another conflict, then an old one will be replaced with the new one. So there is only one, the most recent, conflict regarding any object.

The used MergeBehaviour is determined by global property "sync2.mergeBehavior". For instance when it's set to "sync2.newIsTheBestMergeBehaviour" then the NewIsTheBestMergeBehaviour will be used.

With Sync 2 version 1.4.0 was provided two example MergeBehaviour:

1) RestrictConflictMergeBehaviourImpl (link) - "sync2.restrictConflictMergeBehaviour"

This is the default strategy which using hash codes check if the both objects were changed then create unresolved conflict in order to be solved by the users. If only one object changed then it will be choose as a newest version.

2) NewIsTheBestMergeBehaviourImpl (link) - "sync2.newIsTheBestMergeBehaviour"

In this strategy used the date of changed to determine the newest version. If the object hasn't the date of change then the strategy will chose the source object.