11 业务对象生命周期及其状态

11.6 Save 业务对象

11.6.1可跨物理域提交业务数据的方法

一般情况下，应该调用 Root 业务对象的 Save()函数来提交整棵“业务结构”树（见“12.业务结构对象模型”章节），除非从来不会编辑到 Root 业务对象才允许提交其中的某棵子树，而且务必自始至终仅能编辑这颗子树，否则整棵树的编辑状态将会被打乱，这是 CSLA 不允许的。

被提交的“业务结构”树中所有的业务对象，都将在一个新事务中被持久化到数据库中。

11.6.1.1 Phenix.Business.BusinessBase<T>提供的提交函数

```
    /// <summary>
    /// 保存
    /// </summary>
    /// <param name="needCheckDirty">校验数据库数据在下载到提交期间是否被更改过，一旦发现将报错：CheckDirtyException；如果ClassAttribute.AllowIgnoreCheckDirty = false本功能无效，必定报错：CheckSaveException </param>
    /// <param name="firstTransactionData">参与事务处理前端的业务队列</param>
    /// <param name="lastTransactionData">参与事务处理末端的业务队列</param>
    /// <returns>成功提交的业务对象</returns>
    public T Save(bool needCheckDirty, IBusiness[] firstTransactionData, IBusiness[] lastTransactionData)
```

11.6.1.2 Phenix.Business.BusinessListBase<T, TBusiness>提供的提交函数

```
    /// <summary>
    /// 保存
    /// </summary>
    /// <param name="needCheckDirty">校验数据库数据在下载到提交期间是否被更改过，一旦发现将报错：CheckDirtyException；如果ClassAttribute.AllowIgnoreCheckDirty = false本功能无效，必定报错：CheckSaveException </param>
    /// <param name="onlySaveSelected">仅提交被选择的业务对象</param>
    /// <param name="firstTransactionData">参与事务处理前端的业务队列</param>
    /// <param name="lastTransactionData">参与事务处理末端的业务队列</param>
    /// <returns>成功提交的业务对象集合</returns>
    public T Save(bool needCheckDirty, bool? onlySaveSelected, IBusiness[] firstTransactionData, IBusiness[] lastTransactionData)
```

11.6.2嵌入到现有事务中提交业务数据的方法，不可跨物理域

这些需提交的业务对象，来源于：

- 作为业务对象上可序列化的一个字段，寄生在客户端的业务对象上，随着业务结构的提交一起被带到服务端；
- 在服务端的执行代码里 New、Fetch 产生；

当它们被编辑后，要和提交上来的业务结构在一个事务中被持久化到数据库。

11.6.2.1 Phenix.Business.BusinessBase<T>提供的提交函数

```
/// <summary>
/// 保存
/// </summary>
/// <param name="transaction">数据库事务，如果为空则将重启新事务</param>
/// <param name="needCheckDirty">校验数据库数据在下载到提交期间是否被更改过，一旦发现将报错：CheckDirtyException；如果ClassAttribute.AllowIgnoreCheckDirty = false本功能无效，必定报错：CheckSaveException </param>
/// <param name="firstTransactionData">参与事务处理前端的业务队列</param>
/// <param name="lastTransactionData">参与事务处理末端的业务队列</param>
/// <returns>成功提交的业务对象</returns>
public T Save(DbTransaction transaction, bool needCheckDirty, IBusiness[] firstTransactionData, IBusiness[] lastTransactionData)
```

前文的示例中已使用到了本功能，_linkAssemblyInfo 是寄生在 AssemblyClassInfo 类上的一个字段，可以在更新 AssemblyClassInfo 对象的时候，一起被持久化：

```
/// <summary>
/// 更新本对象集合之前
/// 在运行持久层的程序域里被调用
/// </summary>
/// <param name="transaction">数据库事务</param>
/// <param name="limitingCriteriaExpressions">限制保存的条件(not exists 条件语句)</param>
protected override void OnUpdatingSelf(DbTransaction transaction, ref List<CriteriaExpression> limitingCriteriaExpressions)
{
  if (_linkAssemblyInfo != null && _linkAssemblyInfo.IsDirty)
    _linkAssemblyInfo.Save(transaction);
}
```

11.6.2.2 Phenix.Business.BusinessListBase<T, TBusiness>提供的提交函数

```
    /// <summary>
    /// 保存
    /// </summary>
    /// <param name="transaction">数据库事务，如果为空则将重启新事务</param>
    /// <param name="needCheckDirty">校验数据库数据在下载到提交期间是否被更改过，一旦发现将报错：CheckDirtyException；如果ClassAttribute.AllowIgnoreCheckDirty = false本功能无效，必定报错：CheckSaveException </param>
    /// <param name="onlySaveSelected">仅提交被选择的业务对象</param>
    /// <param name="firstTransactionData">参与事务处理前端的业务队列</param>
    /// <param name="lastTransactionData">参与事务处理末端的业务队列</param>
    /// <returns>成功提交的业务对象集合</returns>
    public T Save(DbTransaction transaction, bool needCheckDirty, bool? onlySaveSelected, IBusiness[] firstTransactionData, IBusiness[] lastTransactionData)
```