



EfsFrame 企业级框架设计原理分析

| | | | |
|---|----------|---------------|--|
| 版本号：V1.0 版 | | EfsFrame 框架团队 | |
| 联系方式： | | | |
| 网址： http://www.efsframe.cn/ | | | |
| 联系人： | 郭志军 | Email： | enjsky@163.com |
| Q Q： | 68098375 | 联系电话： | 13297990437 |

目 录

| | |
|---------------------------------------|----|
| 1、EfsFrame 框架整体说明 | 3 |
| 1.1、Efs 总体设计思想..... | 3 |
| 1.2、Efs 总体架构图..... | 5 |
| 1.3、EfsFrame 框架的适用开发平台..... | 5 |
| 1.4、学习 EfsFrame 框架的前提 | 6 |
| 2、EfsFrame 框架相关概念说明 | 7 |
| 2.1、事务类型、事件类型 | 7 |
| 2.2、单位、用户管理 | 8 |
| 2.3、角色管理 | 8 |
| 2.4、字典管理 | 9 |
| 2.4、编码规则管理 | 11 |
| 2.5、系统日志 | 12 |
| 2.6、错误日志 | 13 |
| 2.7、汉字管理 | 13 |
| 3、EfsFrame 框架用户体验篇 | 14 |
| 3.1、 总体标准页面布局 | 14 |
| 3.2、 标准样式按钮（支持快捷键、小图标） | 14 |
| 3.3、 多样式的 Menu 菜单（支持快捷键、小图标） | 14 |
| 3.4、 Window 窗体..... | 14 |
| 3.5、 分页查询列表 | 15 |
| 3.6、 多页签布局 | 15 |
| 3.7、 统一的表单域样式 | 15 |
| 3.8、 全键盘事件 | 15 |
| 3.9、 字典快速索引选择 | 16 |
| 3.10、 表单域输入的及时验证及输入提醒 | 17 |
| 3.11、 Ajax 技术实现的各种异步提交 | 18 |
| 4、EfsFrame 框架业务操作流程分析 | 19 |
| 4.1、添加、修改、删除操作流程分析 | 19 |
| 4.2、查询列表功能操作流程分析 | 22 |

1、EfsFrame 框架整体说明

1.1、Efs 总体设计思想

Efs 是一套基础的企业级开发解决方案，整个框架体系中包含了 Web 表现层开发包，组件开发包，基础数据库设计一整套完整的基于 B/S 架构应用程序设计开发的完整方案。

EfsFrame 框架从研发到时间，历时近 10 年，积累了大量实战软件工程专家、数学专家的心血不断完善而成，已应用的大大小小的项目几十个，从小项目的开发管理维护设计到大项目的负载均衡设计，Efs 逐渐形成了一整套完整的基于 B/S 架构的设计解决方案。

EfsFrame 框架设计目标：

- l 整体提升企业的项目管理水平；
- l 整体提升企业的研发人员的研发水平；
- l 整体提升企业的项目研发效率；
- l 整体提升企业的项目研发的健壮性；
- l 最大限度减少企业的项目维护成本；

EfsFrame 框架特点如下：

- 1、完善的 Web 表现层开发包：为企业 Web 表现层开发人员提供的一套完整、高效、美观的 B/S 结构设计表现层解决方案。
 - a) JS + DIV + CSS 的表现层设计，与语言无关，支持各种编程语言环境；
 - b) 完善的 JS 类库，让各种优美的 Web 表现能轻松按照配置实现，极大提高企业的项目 Web 表现层的开发效率；
 - c) 美观的布局，全局的键盘事件，快速的数据检索设计，最大限度提升产品的用户体验；
 - d) 统一企业的项目 UI 设计，统一的框架结构，能迅速规范企业的 Web 表现层代码设计规范，最大限度的减轻企业后期的项目管理、维护、升级成本；
 - e) 完善的表现层 API 帮助，减少企业 Web 表现层开发人员的培训投入；

2、完善的组件开发包：为企业组件开发人员提供的一套完整、稳定、高效的 B/S 结构设计业务逻辑层解决方案。

- a) 完善的基础类库的封装，极大提高企业的项目业务逻辑层组件开发效率，最大限度让业务接口组件简洁、高效；
- b) 统一的接口规范，能迅速规范企业的业务逻辑层组件代码设计规范，最大限度的减轻企业后期项目管理、维护升级成本；
- c) 完善的组件层 API 帮助，减少企业业务逻辑层组件开发人员的培训投入；

3、完善的基础数据库设计：

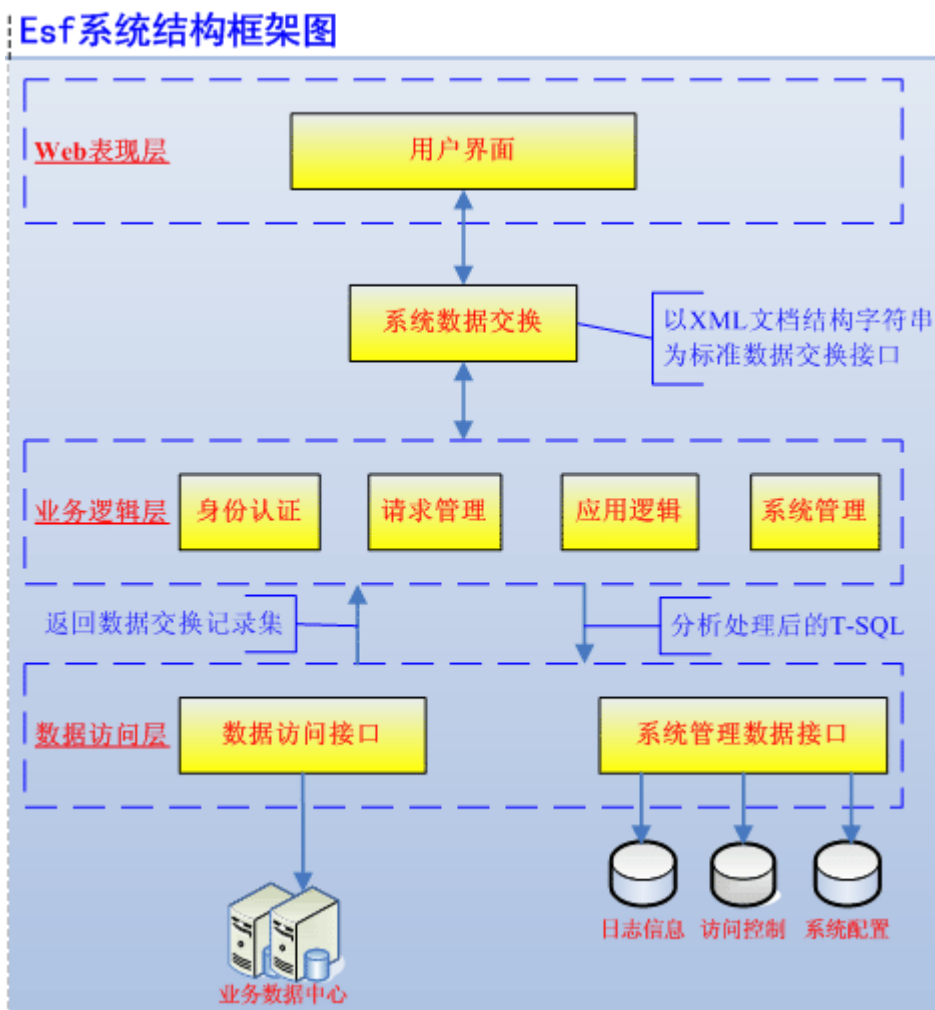
- a) 完整的事务、事件管理、用户、单位、角色、权限管理设计，能快速帮助企业在不同的项目中快速完成用户、单位、角色、权限的分配，迅速投入到项目本身的业务系统开发中。
- b) 完整的字典管理功能，能方便的对业务系统的全部字典文件进行维护。
- c) 分页查询存储过程设计，为业务系统开发过程中的分页查询提升效率。
- d) 编码分配设计，只需要通过配置即可快速实现可满足各种要求的唯一编码。
- e) 汉字拼音管理，收录了常用的 2 万多汉字的全拼与简拼，能迅速完成对汉字的全拼与简拼的翻译处理。

4、分层结构设计：EfsFrame 框架严格按照 MVC 模式设计开发。

- a) 能帮助企业迅速发挥团队开发优势，合理分工协作（能迅速将 Web 表现层开发，业务逻辑组件开发，系统设计合理分离）。
- b) 标准的三层结构模型，为系统的稳定、高效运行打下坚实基础。

1.2、EfsFrame 总体架构图

对于 EsfFrame 框架的应用结构，按照标准的多层结构要求设计如图：



1.3、EfsFrame 框架的适用开发平台

EfsFrame 框架目前已提供的应用平台解决方案包含：

- 1、基于 Windows 平台的 Asp + Com Plus + SQL Server/ Oracle 应用模型；
- 2、基于 Windows 平台的 .Net(C#) + SQL Server 应用模型；
- 3、基于 Java 环境的 Jsp + Java 中间组件 + Oracle/SQL Server/MySql 应用模型；

其他应用平台组合，我们可为您快速定制开发。

1.4、学习 EfsFrame 框架的前提

EfsFrame 框架是一套完整的企业级开发应用平台，不是一套入门级编程教程，所以对学习 EfsFrame 框架的开发人员有一定的要求，具体如下：

- 1、有一定 B/S 架构项目开发经验；
- 2、作为平台的 Web 表现层开发人员，要熟悉 HTML, JavaScript, 了解 XML 编程，熟悉一种 Asp、C#、Jsp 至少一种表现层开发脚本语言，适当了解数据库设计原理；
- 3、作为业务逻辑层组件开发人员，至少熟悉一种高级编程语言（如：pascal，VB，C#，C++、Java 等），熟悉 XML 编程，熟悉数据库设计原理（对表设计、视图、存储过程、自定义函数等有一定的认识）。

2、EfsFrame 框架相关概念说明

2.1、事务类型、事件类型

事务类型：是指我们业务开发过程中，对某一些事件的分组名称，也可以叫业务组名称，如：学生档案管理、订单管理等；

事件类型：是指我们业务开发过程中，具体的某一个操作的事件名称，如：学生档案管理里面的添加学生、修改学生、删除学生、查询学生档案等，或者如订单管理中的添加订单、修改订单、审核订单、删除订单、订单查询等；

事件类型是包含在事务类型下面的子信息，事务类型是对事件类型的分组管理，EfsFrame 框架中所有的事件类型在添加的过程中，会要求指定对应的 url 地址，即该业务操作的具体业务实现页面的地址，这样，我们在不同角色用户登录系统后，根据其事件类型权限，就能实现二级的功能树菜单，如下图：

|  增加事务(A)  编辑事务(E)  生成字典文件(T)  返回(B) | | |
|---|--------|--------|
| 事务类型名称 | 事务类型模式 | 事务类型描述 |
| 1 开发配置 | 系统类 | 系统开发 |
| 2 系统管理 | 系统类 | 系统管理 |
| 3 用户消息 | 消息类 | 消息 |
| 4 查询 | 业务类 | 查询组 |
| 5 学生档案管理 | 业务类 | 学生档案管理 |

| 事件类型编号 | 事件类型名称 | 事务类型名称 | 是否禁用 | 是否显示 |
|-----------|--------|--------|------|------|
| 1 000101 | 事务类型管理 | 开发配置 | 否 | 是 |
| 2 000102 | 事件类型管理 | 开发配置 | 否 | 是 |
| 3 000201 | 设置系统参数 | 系统管理 | 否 | 是 |
| 4 000202 | 角色管理 | 系统管理 | 否 | 是 |
| 5 000203 | 用户管理 | 系统管理 | 否 | 是 |
| 6 000204 | 普通字典管理 | 系统管理 | 否 | 是 |
| 7 000205 | 编码管理 | 系统管理 | 否 | 是 |
| 8 000206 | 汉字管理 | 系统管理 | 否 | 是 |
| 9 000207 | 发布公告 | 系统管理 | 否 | 是 |
| 10 000208 | 系统日志 | 系统管理 | 否 | 是 |
| 11 000209 | 错误日志 | 系统管理 | 否 | 是 |
| 12 000210 | 区域定义 | 系统管理 | 否 | 是 |
| 13 000211 | 单位管理 | 系统管理 | 否 | 是 |
| 14 300101 | 添加学生 | 学生档案管理 | 否 | 是 |
| 15 300102 | 管理学生信息 | 学生档案管理 | 否 | 是 |

第 1 页 共 1 页 显示第 1 - 15 条 共 15 条 Design By:efsframe.cn

 增加事件类型(A)
  编辑事件类型(E)
  生成字典文件(T)
  返回(B)

2.2、单位、用户管理

任何一个应用系统都离不开单位、用户的管理，所有的权限分配都围绕不同的单位级别的不同类型用户展开，所以在 EfsFrame 框架中，我们将此部分默认到系统中，并给了一些常规的字段，但是在实际的项目应用中，可能需要对这部分进行相应的修改，特别是单位类型、单位级别、用户类型这些字段，可以根据自身的业务需要来进行调整，以便更加灵活的实现各种权限的分配。

2.3、角色管理

角色管理：权限分配是应用系统中最基本的功能，所以我们设计了一个简单的权限分配体系在 EfsFrame 框架中，便于提升大家的项目开发效率，在角色中，我们可以给角色分配用户，分配事件类型，在同一个角色中的用户，拥有相同的事件类型权限，同一个用户可以隶属于不同的角色，同一个事件可以隶属于不同的角色，这和 Windows 的角色管理有类似之处。用户登录后，功能树的出现即是以所对应的角色的事件类型的集合类展现的。

图例展示：

角色属性

角色事件权限

角色用户

角色事件权限列表

+

添加事件类型

✖

删除事件类型

| <div><div></div><div></div></div> | 事件类型编号 | 事件类型名 | 是否禁用 | 是否快捷键 | 是否显示 |
|-------------------------------------|--------|--------|------|-------|------|
| 1 <div><div></div><div></div></div> | 300101 | 添加学生 | 否 | 否 | 是 |
| 2 <div><div></div><div></div></div> | 300102 | 管理学生信息 | 否 | 否 | 是 |

角色属性

角色事件权限

角色用户

角色用户列表

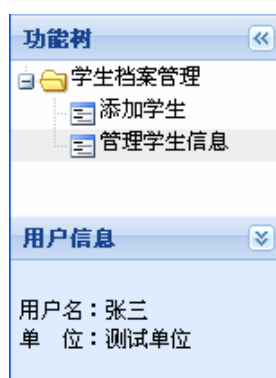
+

添加用户

✖

删除用户

| <div><div></div><div></div></div> | 用户编号 | 用户名称 | 用户姓名 | 用户单位 | 是 |
|-------------------------------------|------------|----------|------|------|---|
| 1 <div><div></div><div></div></div> | 0000000108 | zhangsan | 张三 | 测试单位 | |



2.4、字典管理

字典管理：字典管理作为 EfsFrame 框架的亮点，开始不容易为人所理解，但是掌握后又能让人耳目一新。

这里说的字典是指我们在项目中经常用到的一些可枚举项的集合，表现的常规的 Web 上就是下拉框，比如说：性别字典、学历字典、籍贯字典、是否字典、婚姻状态字典等，这些属性的内容都是可以枚举的项，所以我们统称他们为字典。

往往在不同的行业，不同的应用系统中，会有很多字典需要管理与维护，为了便于统一管理，我们将该管理放在了 EfsFrame 框架中，以后大家在业务开发过程中，可以轻松的将所有的字典项纳入到系统管理平台中，添加新的字典后，只要生成字典文件即可在页面调用。

在字典管理中的注意事项：

- 1、字典名称，我们一般以大写的 **DIC_** 开头，这是一种编码规范；
- 2、字典条目中的字典编码，一律要以数字表示，如：001、002 等，便于页面的通过编码快速检索；
- 3、添加完成的字典，生成字典文件，会统一以 UTF-8 编码模式的 XML 文件存放在应用系统项目的 **dic** 目录下，以便页面调用；
- 4、生成的字典 XML 文件会自动完成字典条目描述的全拼、简拼的构造，方便页面在快速字典选择过程中，通过编码、简拼、全拼的方式来检索字典数据；
- 5、所有的字典信息，我们将在系统启动时，将其缓存到应用服务器内存中，在做各种字典翻译的时候，特别是在详细信息查询时，就不需要做关联查询，自动从内存中获得字典翻译内容，从而大大提升系统的运行效率；

图例展示：

字典简单查询字典列表

字典列表字典操作字典条目维护(E)生成字典文件(C)返回(B)

| 字典名称 | 字典物理名称 | 编码长度 | 字典类型 |
|---------------------|--------|------|------|
| 1 DIC_ABLE | 能否 | 1 | 开发字典 |
| 2 DIC_ACCUSAL | 罪名 | 8 | 标准字典 |
| 3 DIC_AFFAIRTYPE... | 事务类型模式 | 1 | 开发字典 |
| 4 DIC_AFFAIRTYPE... | 事务类型分类 | 1 | 开发字典 |
| 5 DIC_CESHI | 测试用的 | 2 | 普通字典 |
| 6 DIC_CLAN | 政治面貌 | 2 | 开发字典 |
| 7 DIC_CODE | 行政区划 | 6 | 标准字典 |
| 8 DIC_DICEDITABLE | 字典修改权限 | 1 | 开发字典 |
| 9 DIC_DUTY | 职务 | 2 | 标准字典 |
| 10 DIC_EDUCATION | 文化程度 | 2 | 标准字典 |
| 11 DIC_GENDER | 性别 | 1 | 标准字典 |
| 12 DIC_HAVEORNO | 有无 | 1 | 开发字典 |
| 13 DIC_HEALTH | 健康状况 | 1 | 标准字典 |
| 14 DIC_IDPARA | 编码类型规则 | 2 | 普通字典 |
| 15 DIC_LOGOUT_ST... | 注销状态表 | 1 | 普通字典 |
| 16 DIC_MARRIAGE | 婚姻状况 | 1 | 标准字典 |

第 1 页 共 2 页

显示第 1 - 25 条 共 26 条 Design By:efsframe.cn

添加新的字典

确定(O)

字典名称 DIC_USERTYPE

字典描述 用户类型

字典编码长度 2

字典修改权限 普通字典

字典面板

混 请输入查询关键字 查询

| 编码 | 字典值 |
|----|-------|
| 0 | 开发人员 |
| 1 | 实施人员 |
| 2 | 系统管理员 |
| 3 | 普通管理员 |

第 1 页 共 1 页

处理字典条目: DIC_USERTYPE

增加条目(A) 编辑条目(E) 删除条目(D) 生成字典文件(C)

| 字典编码 | 字典内容 | 是否有效 |
|------|-------|------|
| 1 0 | 开发人员 | |
| 2 1 | 实施人员 | |
| 3 2 | 系统管理员 | |
| 4 3 | 普通管理员 | |

修改字典条目

确定(O)

字典编号 1

字典内容 实施人员

第 1 页 共 1 页 显示第 1 - 4 条 共 4 条

2.4、编码规则管理

编码规则管理：在业务开发过程中，我们往往要给业务表的主键来快速生成唯一的编码，以避免主键唯一冲突，显然自动增长列是不太灵活的，很多时间无法满足我们的需求，我们所设计的编码规则管理就是为了解决此问题。

| 属性 | 说明 |
|------|--------------------------------------|
| 编码编号 | 一个应用系统中，可能有很多个唯一编码，所以我们给每个编码规则分配一个编号 |
| 编码名称 | 给不同的编码规则取名 |
| 编码长度 | 每次根据该规则生成的编码要多长 |
| 编码规则 | 是否将编码的种子加在新编码的前面 |
| 是否循环 | 当一个种子编码达到了最大编码后，之后重新从最小编码开始编号 |
| 最小值 | 编码的最小起始值 |
| 最大值 | 编码能达到的最大值 |

备注：

所有的编码生成，都调用统一的数据库存储过程来完成编码分配，在存储过程中设计了行级锁（同时只能有一个操作在该行），所以能绝对保证编码的唯一性，同时对该编码生成的存储过程也写在相应的类 NumAssign 中，每次只需要调用该类对象，传入编码编号，种子，即可按配置要求获得最新的唯一编码。

实例说明：

| 编码编号 | 编码名称 | 编码长度 | 编码规则 | 是否循环 | 最小值 | 最大值 |
|--------|------|------|------|------|-----|--------|
| 100001 | 学生编号 | 6 | 叠加种子 | 否 | 1 | 999999 |

1、传入种子“0910”两位年+两位月，生成的编码为：0910000001 —— 0910999999

2、传入种子“0911”两位年+两位月，生成的编码为：0911000001 —— 0911999999

每次传入不同的种子，则重新开始一个新的编码，当我们不需要叠加种子的

时候，只生成如 0000001 这样的编码时，则只需要在编码规则上修改为不叠加种子即可。

图例展示：



2.5、系统日志

记录了所有用户的登录信息，如登录时间，登录 IP、用户单位、用户名称、用户编码等信息，用于系统跟踪。

图例展示：

| 登录日志列表 | | | | | | |
|--------|----------------|------------|-------|--------|-----------|---------------------|
| | 日志编号 | 用户编号 | 用户姓名 | 用户单位名称 | 登录IP地址 | 登录时间 |
| 1 | 44190000002215 | 4201000001 | 系统管理员 | 测试单位 | 127.0.0.1 | 2009-11-18 9:20:16 |
| 2 | 44190000002214 | 0000000108 | 张三 | 测试单位 | 127.0.0.1 | 2009-11-17 22:59:52 |
| 3 | 44190000002213 | 4201000001 | 系统管理员 | 测试单位 | 127.0.0.1 | 2009-11-17 22:30:56 |
| 4 | 44190000002212 | 4201000001 | 系统管理员 | 测试单位 | 127.0.0.1 | 2009-11-3 20:29:22 |
| 5 | 44190000002211 | 4201000001 | 系统管理员 | 测试单位 | 127.0.0.1 | 2009-11-3 20:10:27 |
| 6 | 44190000002210 | 4201000001 | 系统管理员 | 测试单位 | 127.0.0.1 | 2009-11-3 20:07:03 |

2.6、错误日志

只要调用了标准的业务存储层类对象，所有的错误执行的 SQL 语句都将记录下来，便于开发人员排错处理，其中注意：对象标识相同的 SQL 语句错误日志为同一个事务处理对象封装的 SQL，一起执行的，其中至少有一条语句错误，所以导致事务提交失败而回滚。

图例展示：

| ID | 产生时间 | 对象标识 | 错误描述 |
|-----|---------------------|---------------------|--|
| 1 | 2009-10-28 17:19 | 09-1030000000000000 | UPDATE EVENTTYPE SET EVENTTYPENAME='管理学生信息' /AFFACTYPEID='000001' /EVENTTYPEID='000001'... |
| 2 | 2009-10-30 11:53:58 | 09-0280000000000000 | UPDATE SPELL SET WORD=' ' /SPELL=' ' /SPELL=' '... |
| 3 | 2009-10-30 11:53:53 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 4 | 2009-10-30 11:53:49 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 5 | 2009-10-30 11:53:43 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 6 | 2009-10-30 11:53:38 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 7 | 2009-10-30 11:53:33 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 8 | 2009-10-30 11:53:28 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 9 | 2009-10-30 11:53:23 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 10 | 2009-10-30 11:53:18 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 11 | 2009-10-30 11:53:13 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 12 | 2009-10-30 11:53:08 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 13 | 2009-10-30 11:53:03 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 14 | 2009-10-30 11:52:58 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 15 | 2009-10-30 11:52:53 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 16 | 2009-10-30 11:52:48 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 17 | 2009-10-30 11:52:43 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 18 | 2009-10-30 11:52:38 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 19 | 2009-10-30 11:52:33 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 20 | 2009-10-30 11:52:28 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 21 | 2009-10-30 11:52:23 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 22 | 2009-10-30 11:52:18 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 23 | 2009-10-30 11:52:13 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 24 | 2009-10-30 11:52:08 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 25 | 2009-10-30 11:52:03 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 26 | 2009-10-30 11:51:58 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 27 | 2009-10-30 11:51:53 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 28 | 2009-10-30 11:51:48 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 29 | 2009-10-30 11:51:43 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 30 | 2009-10-30 11:51:38 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 31 | 2009-10-30 11:51:33 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 32 | 2009-10-30 11:51:28 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 33 | 2009-10-30 11:51:23 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 34 | 2009-10-30 11:51:18 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 35 | 2009-10-30 11:51:13 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 36 | 2009-10-30 11:51:08 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 37 | 2009-10-30 11:51:03 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 38 | 2009-10-30 11:50:58 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 39 | 2009-10-30 11:50:53 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 40 | 2009-10-30 11:50:48 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 41 | 2009-10-30 11:50:43 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 42 | 2009-10-30 11:50:38 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 43 | 2009-10-30 11:50:33 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 44 | 2009-10-30 11:50:28 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 45 | 2009-10-30 11:50:23 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 46 | 2009-10-30 11:50:18 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 47 | 2009-10-30 11:50:13 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 48 | 2009-10-30 11:50:08 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 49 | 2009-10-30 11:50:03 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 50 | 2009-10-30 11:49:58 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 51 | 2009-10-30 11:49:53 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 52 | 2009-10-30 11:49:48 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 53 | 2009-10-30 11:49:43 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 54 | 2009-10-30 11:49:38 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 55 | 2009-10-30 11:49:33 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 56 | 2009-10-30 11:49:28 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 57 | 2009-10-30 11:49:23 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 58 | 2009-10-30 11:49:18 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 59 | 2009-10-30 11:49:13 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 60 | 2009-10-30 11:49:08 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 61 | 2009-10-30 11:49:03 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 62 | 2009-10-30 11:48:58 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 63 | 2009-10-30 11:48:53 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 64 | 2009-10-30 11:48:48 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 65 | 2009-10-30 11:48:43 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 66 | 2009-10-30 11:48:38 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 67 | 2009-10-30 11:48:33 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 68 | 2009-10-30 11:48:28 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 69 | 2009-10-30 11:48:23 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 70 | 2009-10-30 11:48:18 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 71 | 2009-10-30 11:48:13 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 72 | 2009-10-30 11:48:08 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 73 | 2009-10-30 11:48:03 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 74 | 2009-10-30 11:47:58 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 75 | 2009-10-30 11:47:53 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 76 | 2009-10-30 11:47:48 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 77 | 2009-10-30 11:47:43 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 78 | 2009-10-30 11:47:38 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 79 | 2009-10-30 11:47:33 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 80 | 2009-10-30 11:47:28 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 81 | 2009-10-30 11:47:23 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 82 | 2009-10-30 11:47:18 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 83 | 2009-10-30 11:47:13 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 84 | 2009-10-30 11:47:08 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 85 | 2009-10-30 11:47:03 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 86 | 2009-10-30 11:46:58 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 87 | 2009-10-30 11:46:53 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 88 | 2009-10-30 11:46:48 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 89 | 2009-10-30 11:46:43 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 90 | 2009-10-30 11:46:38 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 91 | 2009-10-30 11:46:33 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 92 | 2009-10-30 11:46:28 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 93 | 2009-10-30 11:46:23 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 94 | 2009-10-30 11:46:18 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 95 | 2009-10-30 11:46:13 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 96 | 2009-10-30 11:46:08 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 97 | 2009-10-30 11:46:03 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 98 | 2009-10-30 11:45:58 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 99 | 2009-10-30 11:45:53 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |
| 100 | 2009-10-30 11:45:48 | 09-0280000000000000 | UPDATE SPELL SET TEXT=' ' /SPELL=' ' /SPELL=' '... |

2.7、汉字管理

EfsFrame 框架收录了 2 万多个汉字的全拼和简拼信息，存储在数据库中，在系统启动时，会自动将其缓存到服务器内存中，便于进行一些拼音翻译处理，如：字典的翻译，简拼或全拼的同音查询操作等。

图例展示：

| 字 | 拼音头 | 全拼 |
|---|-----|----|
| 埃 | a | ai |
| 挨 | a | ai |
| 洩 | a | ai |
| 鏖 | a | ai |
| 鏖 | a | ai |

3、EfsFrame 框架用户体验篇

3.1、 总体标准页面布局



3.2、 标准样式按钮（支持快捷键、小图标）



3.3、 多样式的 Menu 菜单（支持快捷键、小图标）



3.4、 Window 窗体



3.5、 分页查询列表

| 字典列表 | | | | |
|--------------------------------|--------|------|------|--|
| 字典操作 字典条目维护(E) 生成字典文件(C) 返回(B) | | | | |
| 字典名称 | 字典物理名称 | 编码长度 | 字典类型 | |
| 1 DIC_ABLE | 能否 | 1 | 开发字典 | |
| 2 DIC_ACCUSAL | 罪名 | 8 | 标准字典 | |
| 3 DIC_AFFAIRTYPE... | 事务类型模式 | 1 | 开发字典 | |
| 4 DIC_AFFAIRTYPE... | 事务类型分类 | 1 | 开发字典 | |
| 5 DIC_CESHI | 测试用的 | 2 | 普通字典 | |
| 6 DIC_CLAN | 政治面貌 | 2 | 开发字典 | |
| 7 DIC_CODE | 行政区划 | 6 | 标准字典 | |
| 8 DIC_DICEDITABLE | 字典修改权限 | 1 | 开发字典 | |
| 9 DIC_DUTY | 职务 | 2 | 标准字典 | |
| 10 DIC_EDUCATION | 文化程度 | 2 | 标准字典 | |
| 11 DIC_GENDER | 性别 | 1 | 标准字典 | |
| 12 DIC_HAVEORNO | 有无 | 1 | 开发字典 | |
| 13 DIC_HEALTH | 健康状况 | 1 | 标准字典 | |

第 1 页 共 2 页 显示第 1 - 25 条 共 26 条 Design By:efsframe.cn

3.6、 多页签布局

| | |
|--------|------|
| 字典简单查询 | 字典列表 |
| 字典名称 | |
| 字典描述 | |
| 字典编码长度 | |
| 字典修改权限 | |

| | |
|---------------------|--------|
| 字典简单查询 | 字典列表 |
| 字典名称 | 字典物理名称 |
| 1 DIC_ABLE | 能否 |
| 2 DIC_ACCUSAL | 罪名 |
| 3 DIC_AFFAIRTYPE... | 事务类型模式 |

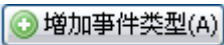
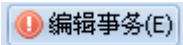
3.7、 统一的表单域样式

| | | | |
|------|--|------|--|
| 编码编号 | | 编码名称 | |
| 编码长度 | | 编码规则 | |
| 是否循环 | | 最小值 | |
| 最大值 | | | |

3.8、 全键盘事件

EfsFrame 框架为每个表单域、按钮都提供了键盘事件的支持，如上图的表单

域中我们只要将焦点移动到其中任何一个表单域上，即可通过键盘的上（↑）下键（↓）来快速的移动焦点，如果是 `textarea` 表单域则需采用 `Ctrl + ↑` 或 `Ctrl + ↓` 进行焦点的上下移动。

按钮或菜单的快捷键会显示在按钮的后方的括弧中，即可通过 `Alt + 字母` 即可快速完成点击  表示快捷键（`Alt + A`）；  表示快捷键为（`Alt + E`）；

3.9、字典快速索引选择

EfsFrame 框架抛弃了原来的下拉框的方式，采用了字典选择的模式，如下图在输入籍贯的时候出现的籍贯字典：



为了方便快捷的进行字典项的选择，EfsFrame 框架为其提供了多种快速输入模式，如：编码检索、简拼检索、文字检索、全拼检索，如下图：



3.10、 表单域输入的及时验证及输入提醒

EfsFrame 框架中的表单域，根据 kind 属性进行数据校验，如果为无效的数据，将会取消录入或者红色提醒，如图：

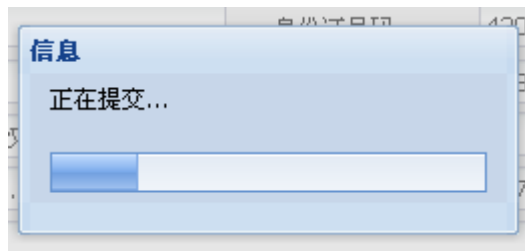
| | |
|--------|---|
| 用户姓名 | <input type="text"/> |
| 职务 | <input type="text"/> ❗ 该输入项为必填项 |
| 公民身份号码 | <input type="text" value="42011419811004"/> 性别 |
| 出生日期 | <input type="text"/> ❗ 请输入15或18位的公民身份号码 |
| 出生日期 | <input type="text"/> |
| 籍贯 | <input type="text"/> ❗ 日期格式 2009-10-10 |

| | | | |
|------|---|--------------|-------------------------------------|
| 出生日期 | <input type="text" value="2009-01"/> | 民族 | <input type="text"/> |
| 籍贯 | <div>❗ 2009-01 是无效的日期 - 必须符合格式：Y-m-d</div> | | |
| 用户类型 | <div>普通管理员 请从字典中选择数据</div> | | |
| 邮箱 | <input type="text" value="lk "/> | 电话号码 | <input type="text" value="027878"/> |
| 备注 | <div>❗ 该输入项必须是电子邮件地址，格式如： "user@domain.com"</div> | | |
| 年龄 | <input type="text"/> | | |
| 电话号码 | <input type="text" value="02 请输入整数 5"/> | 非数字字符，不能输入进去 | |

3.11、 Ajax 技术实现的各种异步提交

EfsFrame 框架中的对表单域对象，Grid 列表都可以轻松实现异步提交获取数据，最大限度的减少页面刷新，提升用户体验

表单域增加、修改、删除等操作的异步提交



Grid 列表的异步获取列表数据

| 身份证号 | 性别 | 籍 |
|--------------|--------------------|-----|
| 198605040526 | 女 | 湖北 |
| 983010122 | <div>⌂ 加载...</div> | 北武汉 |

4、EfsFrame 框架业务操作流程分析

4.1、添加、修改、删除操作流程分析

构造添加的实例代码:

```
<div xtype="panel" iconCls="icon-panel" title="添加学生基本信息" border="false"
buttonAlign="center" autoScroll="true">
  <form id="frmData" class="efs-box" url="../../sysadmin/baseRefWeb.aspx?method=PsnAdd"
method="post" onEfsSuccess="frmPostSubBack(true)" onEfsFailure="frmPostSubBack(false)">
    <TABLE class="formArea">
      <TR>
        <TD width="100" labelFor="name">姓 名</TD>
        <TD><INPUT id="name" type="text" kind="zhunicode" must="true" maxlength="50"
fieldname="PERSON/NAME" datatype="0" state="0"></TD>
        <TD width="20"></TD>
        <TD width="100">身份证号码</TD>
        <TD><INPUT type="text" kind="idcard" fieldname="PERSON/IDCARD" sex="sex"
birthday="birthday" datatype="0" state="0"></TD>
      </TR>
      <TR>
        <TD colspan="4"><INPUT type="hidden" kind="text" fieldname="PERSON/PERSONID" datatype="0" state="0"
operation="0" writeevent="0" ><!--operation="0" 定义为添加接口-->
        </TD>
      </TR>
    </TABLE>
  </form>
  <div xtype="buttons">
    <div text="确 定" onEfsClick="doSubmit()"></div>
    <div text="返 回" onEfsClick="doRet()"></div>
  </div>
</div>
```

特别说明:

- 1、form 中的 url 指异步提交的后台指向地址;
- 2、form 中的 onEfsSuccess 对应函数是表示异步提交成功后,返回执行的函数;
- 3、form 中的 onEfsFailure 对应函数是表示异步提交失败后,返回执行的函数;
- 4、input 中 kind 属性表示了数据输入类型,对应说明请参考《扩展 INPUT 标签属性说明》;
- 5、input 中的 fieldname 属性对应了需要拼写的 xml 文档对应路径;
- 6、input 中的 state, datatype 属性说明,请参考《扩展 INPUT 标签属性说明》;
- 7、input 中 operation 属性很重要,在标准 xml 结构中,“0”表示了添加,“1”表示修改,“2”表示删除操作,删除和修改操作时,请注意将关键字段的

state 属性设置为 “5”，则表示 **where** 条件。

确定提交后执行 **doSubmit()** 方法如下：

```
function doSubmit()  
{  
    Efs.getExt("frmData").submit();  
}  
  
<form      id="frmData"      class="efs-box"      url="../sysadmin/baseRefWeb.aspx?method=PsnAdd"  
method="post" onEfsSuccess="frmPostSubBack(true)" onEfsFailure="frmPostSubBack(false)">
```

备注说明：

- 1、**frmData** 为前面构造的 **form** 对象的 **id** 名；
- 2、**Efs.getExt("frmData").submit()** 方法分解说明：
- 3、第一步实现了数据表单的验证，根据所有 **Input** 的 **kind** 属性，自动完成数据校验，如果校验失败，则执行 **onEfsFailure** 函数返回；
- 4、第二步，根据 **Input** 表单的 **fieldname** 属性，自动完成标准 **xml** 结构的拼写，标准的数据交换结构，请参考《EfsFrame 框架标准数据交换 XML 结构说明》；
- 5、第三不，将拼写完成的 **xml** 存放在表单域下面的 **<input type="hidden" name="txtXML">** 里面，如果 **txtXML** 的隐藏表单域不存在，则自动添加一个该隐藏域；
- 6、如果 **form** 的 **action** 不为空，则以同步方式提交 **form** 表单；
- 7、如果 **form** 的 **url** 不为空，则以异步的方式提交 **form** 表单；
- 8、异步方式提交后，如果提交成功，执行 **onEfsSuccess** 对应函数，如果提交失败，执行 **onEfsFailure** 对应函数；

Form 表单域中的提交后台执行分解说明：

```
<form      id="frmData"      class="efs-box"      url="../sysadmin/baseRefWeb.aspx?method=PsnAdd"  
method="post" onEfsSuccess="frmPostSubBack(true)" onEfsFailure="frmPostSubBack(false)">
```

- 1、**url** 指向了统一的公共处理的页面，后面的 **method** 表示调用的方法，在 **baseRefWeb.cs** 中，采用发射的方式，执行到 **Efsframe.cn.baseCls.baseRef** 类中的 **PsnAdd** 方法；
- 2、**PsnAdd** 方法则调用实际的业务操作组件即可；

组件业务代码实现说明:

- 1、如果是已经构造好的 xml 结构文档，则只需要调用 Operation 类中的 dealWithXml 方法即可；
- 2、如需要特殊处理，如给关键字段分配唯一编码，则需要单独写类方法来实现以下，具体方式，见实例文档中的“添加人员”组件方法，同时也可以参考角色管理里面的相关组件方法；

总结一下业务开发步骤:

- 1、完成表结构设计；
- 2、根据表结构设计，完成表现层 form 表单域的各种属性配置，特别注意 kind, datatype, fieldname 属性；
- 3、根据业务操作类型，定义 operation 属性，该属性只组要在同一个表中的字段上定义一次即可，不同的表则需定义多次；
- 4、当操作类型为修改、删除时，修改 operation 属性的同时，需将作为 Where 条件的主关键字指定对应的 state 为 “5”；
- 5、根据标准的组件方法，写一个类和方法来完成对应的业务操作；
- 6、在类 baseRefWeb.cs 中写一个对应的方法去调用业务组件方法，便于页面统一调用，这时候页面只需要关心方法名即可，不用关心具体的业务组件的实现；
- 7、将 baseRefWeb.cs 中对应的业务组件执行的方法写在 form 表单的 url 的 method 后面即可。
- 8、由于我们采用的是标准的 xml 作为数据交换的接口，所有的业务类方法的实现都只需要一个参数即可，所以我们能很轻易的通过反射机制实现业务组件接口的调用，至于组件接口中如何分析和处理这些 xml 结构到数据库中，页面开发人员就不用很关心，节省了页面开发人员的大量时间。

4.2、查询列表功能操作流程分析

构造查询列表的实例代码：

```
<!--Grid列表-->
<div id="psnGrid" region="center" xtype="grid" pagingBar="true" pageSize="25"
onEfsRowClick="doGridClick()" onEfsRowDbClick="onEditEx()">
  <!--数据源-->
  <div id="psnList" xtype="store" url="../sysadmin/baseRefWeb.aspx?method=QryPersonList"
txtXML="" autoLoad="true">
    <div xtype="xmlreader" fieldid="PERSONID" record="ROW" tabName="PERSON"
totalRecords="QUERYINFO@records">
      <div name="PERSONID" mapping="PERSONID"></div>
      <div name="NAME" mapping="NAME"></div>
      <div name="IDCARD"></div>
      <div name="SEX"></div>
      <div name="PLACECODE"></div>
      <div name="BIRTHDAY"></div>
      <div name="TEL"></div>
    </div>
  </div>
  <div xtype="colmodel"><!--列模式-->
    <div type="checkbox"></div>
    <div header="学生编码" width="80" sortable="true" dataIndex="PERSONID"></div>
    <div header="学生姓名" width="80" sortable="true" dataIndex="NAME"></div>
    <div header="身份证号" width="120" sortable="true" dataIndex="IDCARD"
align="center"></div>
    <div header="性别" width="40" sortable="true" dataIndex="SEX" kind="dic"
src="DIC_SEX"></div>
    <div header="籍贯" width="120" sortable="true" dataIndex="PLACECODE" kind="dic"
src="DIC_CODE" align="center"></div>
    <div header="出生日期" width="100" sortable="true" dataIndex="BIRTHDAY"
align="center"></div>
    <div header="联系电话" width="100" sortable="true" dataIndex="TEL"></div>
  </div>
</div>
```

特别说明：

- 1、grid 中的 pagingBar 表示是否显示分页栏，pageSize 表示每页显示多少条数据；
- 2、store 中的 url 指向异步获取列表数据的后台指向地址；
- 3、store 中的 autoLoad 表示是否自动加载数据；
- 4、xmlreader 中的 record 表示读取返回 XML 数据中列的根节点；

- 5、xmlreader 下每一个 div 代表一列，根据其 mapping 来获取 xml 节点的值，如果 mapping 为空，则自动根据 name 来获取节点值；
- 6、colmodel 下的每一个 div 表示对应的列的表现模式，header 表示列头名称，dataIndex 与 xmlreader 中的 name 对应；
- 7、colmodel 下的第一个 div，如果其 type 为 checkbox 表示为可多选的行，表头会出现全线框，如果为 radio 则表示单选行，默认为单选行模式；
- 8、colmodel 下的 div 如果将其 kind 属性这是为 dic，并同时给出 src 的字典名称，说明改列需要做相应的字典翻译，Efs 会自动在客户端列表中完成字典翻译处理，省去了服务端字典翻译的功能；
- 9、grid 中的 onEfsRowClick， onEfsRowDbClick 分别表示单击事件和双击事件，具体的事件返回参数类型，请参考：

行单击事件：

onEfsRowClick: (Object data, Grid this, Number rowIndex, Ext.EventObject e)

参数说明：Object data: 通过data[xmlreader的名称]获取数据,如：

data["PERSONID"] 获取人员编号，以下相同；

行双击事件：

onEfsRowDbClick: (Object data, Grid this, Number rowIndex, Ext.EventObject e)

单元格单击事件：

onEfsCellClick : (Object data, Grid this, Number rowIndex, Number columnIndex, Ext.EventObject e)

单元格双击事件：

onEfsCellDbClick : (Object data, Grid this, Number rowIndex, Number columnIndex, Ext.EventObject e)

构造查询条件进行有条件查询

```
<div iconCls="icon-panel" region="north" height="60" title="查询学生列表" border="false">
  <form id="frmQry" method="post">
    <TABLE class="formAreaTop" width="100%" height="100%" cellpadding="0" cellspacing="0">
      <tr>
```

```
<td>&nbsp;</td>
<td width="60">姓名</td>
<td width="160"><input type="text" class="Edit" kind="text" fieldname="NAME"
operation="like" maxlength="30" hint="模糊查询"></td>
<td width="40">性别</td>
<td width="160"><input type="text" class="Edit" kind="dic" src="DIC_SEX"
fieldname="SEX"></td>
<td width="40">籍贯</td>
<td width="160"><input type="text" class="Edit" kind="dic" src="DIC_CODE"
fieldname="PLACECODE"></td>
<td><input iconCls="icon-qry" type="button" value="查 询" onEfsClick="doQry()"></td>
<td>&nbsp;</td>
</tr>
</TABLE>
</form>
</div>

// 进入查询
function doQry()
{
    // 构造标准的查询条件xml
    var strXml = Efs.Common.getQryXml(Efs.getExt("frmQry"));
    // 将查询条件xml放在grid的txtXML参数中
    Efs.getDom("psnList").txtXML = strXml;
    // 重新加载grid数据
    Efs.getExt("psnGrid").store.load();
}
```

备注说明:

- 1、Efs.Common.getQryXml(Efs.getExt("frmQry"))方法是将指定的 frmQry 表单中所有不为空的表单域根据 fieldname 构造出标准的查询型 xml (查询型 xml 请参考《EfsFrame 框架标准数据交换 XML 结构说明》);
- 2、根据 id, 获得 grid 的 div 对象的 txtXML 属性设置为标准的查询型 XML, 如:
Efs.getDom("psnList").txtXML = strXml;
- 3、重新加载 grid 的数据, 即可完成, 如 Efs.getExt("psnGrid").store.load();
- 4、当空条件查询的时候, 可以设置 Efs.getDom("psnList").txtXML = Efs.Common.getQryXml() 即可, Efs.Common.getQryXml() 获得的是一个空条件的查询型 XML;