密级：普通

文档编号：DevPlan-201801 第 1 版

分册名称： 第 1 册/共1册

南华大学教材管理信息系统

Textbook Management Information System

征订计划子系统

Subscription Plan SubSystem

项目编号

201801

开发教程

目录

[一. 申报计划导入](#_Toc330550266) 4

[1.1 创建教师申报计划和学生申报计划表](#_Toc330550267) 4

[1.2 为教师申报计划和学生申报计划表生成代码](#_Toc330550268) 4

[1.3 自定义申报计划导入界面](#_Toc330550269) 5

[1.4 重写列的标题和宽度](#_Toc330550270) 7

[1.5 添加一个FileName字段](#_Toc330550270) 8

[1.6 添加教师用书和学生用书导入计划对话框](#_Toc330550270) 8

[1.7 添加导入计划按钮](#_Toc330550270) 9

[1.8 实现导入的功能](#_Toc330550270) 10

[1.9 为课程表和教材表生成代码](#_Toc330550270) 15

[1.10 显示课程信息](#_Toc330550270) 21

[1.11 显示教材信息](#_Toc330550270) 23

[1.12 添加删除按钮](#_Toc330550270) 25

[1.13 在新增教师/学生用书中添加课程、教材](#_Toc330550270) 27

[1.14 为课程和教材声明一个检索脚本](#_Toc330550270) 28

[1.15 对课程和教材使用LookupEidtor](#_Toc330550270) 29

[1.16 处理导航](#_Toc330550270) 31

[二. 申报计划下达](#_Toc330550271) 32

[2.1 为相关表生成代码](#_Toc330550272) 32

[2.2 处理导航界面](#_Toc330550273) 33

[2.3 设置界面标题](#_Toc330550274) 33

[2.4 设置属性显示格式](#_Toc330550275) 34

[2.5 设置表格显示内容 3](#_Toc330550276)4

[2.6 设置编辑框显示内容](#_Toc330550276) 35

[2.7 生成检索脚本](#_Toc330550276) 35

[2.8 最终内容显示](#_Toc330550276) 40

[三. 申报计划维护](#_Toc330550277) 41

[3.1 创建教师申报计划表](#_Toc330550278) 41

[3.2 自定义教师申报计划维护表界面](#_Toc330550288) 44

[3.3 筛选学院的申报计划](#_Toc330550295) 46

[3.4 二级教研室筛选](#_Toc330550302) 47

[3.5 点击列中属性显示详细信息](#_Toc330550309) 50

[3.6 修改信息时的二级关联](#_Toc330550316) 51

[3.7 删除新增按钮](#_Toc330550323) 53

[四. 征订计划回告](#_Toc330550271) 54

[4.1 为相关表生成代码](#_Toc330550272) 54

[4.2 处理导航界面](#_Toc330550273) 55

[4.3 设置界面标题](#_Toc330550274) 55

[4.4 设置属性显示格式](#_Toc330550275) 56

[4.5 设置表格显示内容](#_Toc330550276) 56

[4.6 设置编辑框显示内容](#_Toc330550276) 57

[4.7 设置相关属性只读](#_Toc330550276) 57

[4.8 删除新增按钮](#_Toc330550276) 58

[4.9 最终内容显示](#_Toc330550276) 58

[五. 征订计划导出](#_Toc330550271) 59

[5.1 创建一个名为TbMis的新项目](#_Toc330550272) 59

[5.2 创建教材征订计划导出表](#_Toc330550273) 59

[5.3 为教材征订计划导出表生成代码](#_Toc330550274) 60

[5.4 自定义教师用书征订计划导出表界面](#_Toc330550275) 61

[5.5 修改表页面显示内容](#_Toc330550276) 62

[5.6 修改列显示内容](#_Toc330550276) 62

[5.7 在列中显示属性](#_Toc330550276) 63

[5.8 按征订计划状态筛选并选出](#_Toc330550276) 64

[5.9 设置点击不同的列可查看不同的表属性](#_Toc330550276) 65

[5.10 增加将表导出为Excel功能](#_Toc330550276) 66

[5.11 删除新增按钮并设置表属性不可改](#_Toc330550276) 68

**一、申报计划导入**

**1.1 创建教师申报计划和学生申报计划（TeacherWholeData、StudentWholeData）表**

要存储教师申报计划和学生申报计划，我们需要两张计划表。

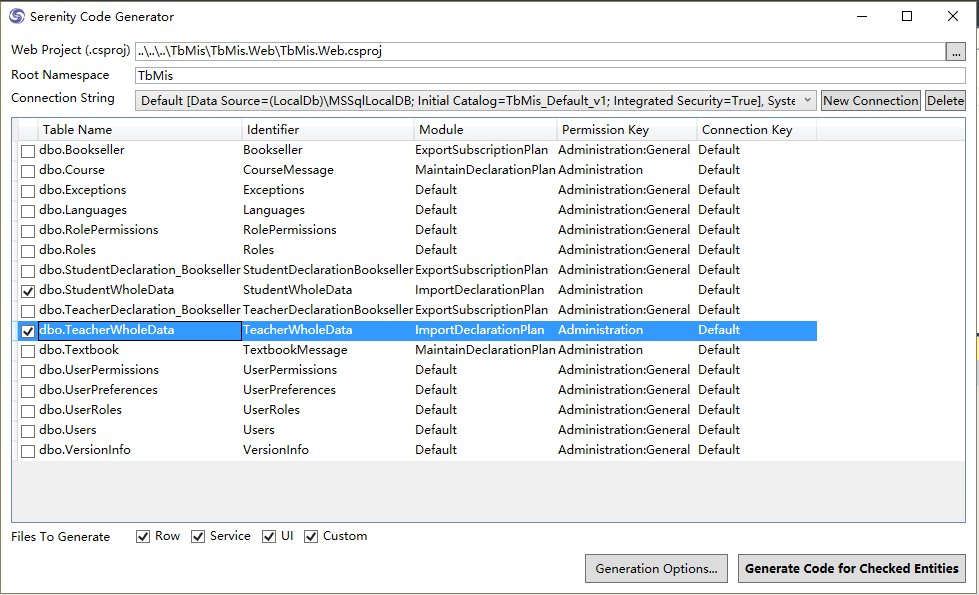
**1.2 为教师申报计划和学生申报计划（TeacherWholeData、StudentWholeData）表生成代码**

Serenity代码生成器

确保表存到数据库，我们将使用Serenity 代码生成器（sergen.exe）生成初始的可编辑界面。

在Visual Studio中，通过点击视图=>其它窗体=>包管理器控制台打开包管理器控制台。

输入sergen并按Enter 键确认。

****

设置项目位置

当第一次运行 Sergen，Web Project 字段默认为：

* ..\..\..\TbMis\TbMis.Web\TbMis.Web.csproj

如果你把该值修改为其他路径，并且生成你的第一个页面，再次使用时，将不需要再次设置。所有的设置将保存在解决方案目录下的 Serenity.CodeGenerator.config。

该值是必填项，因为 Sergen 会借此把生成的文件自动包含到你的 WEB 项目。

Root Namespace 选项

该选项设置为该解决方案使用的名称，如TbMis。

这是很关键的，所以确保你不能把它设置为其他值，因为默认情况下，Serene 模板期望所有生成的代码都在这个根命名空间下面。

该选项也被保存，因此下次使用时你将不用再次填写。

### 选择 Connection String

一旦设置 Web project 名称，Sergen 使用 web.congfig 的连接字符串中填充连接字符串下拉列表。我们这里只有 Default，所以选择 Default。

选择需要生成代码的表（Table Name）

选择TeacherWholeData和StudentWholeData表

设置Module

Module name用于决定命名空间 和生成页面的url。

在此，我们将Module设置为ImportDeclarationPlan。

设置Identifier

这通常对应表名，但是有时候表名含有下划线或其他无效字符，所以需要你决定如何生成代码中的实体名称（一个有效的标识名称）

在此，我们使用默认的表名。

设置Permission Key

我们的申报计划导入页面只有管理员用户可使用，因此，现在让我们把它设为Administration。

生成申报计划导入页面代码

在设置如图所示的参数后，单击Generate Code for Entity按钮。

Sergen将生成几个文件并将它们包含进 TbMis.Web和 TbMis.Script项目。

关闭Sergen并重新加载更改，重新运行会发现已生成申报计划导入页面。

**1.3 自定义申报计划导入界面（以教师用书为例）**

自定义字段标题

当Sergen为教师申报计划（TeacherWholeData）表生成代码时，创建了一个叫TeacherWholeDataRow的实体类，我们可以在Modules/ImportDeclarationPlan/TeacherWholeData/TeacherWholeDataRow.cs找到它。

我们只需要将TeacherWholeDataRow中的属性的DisplayName特性值进行修改即可：

namespace TbMis.ImportDeclarationPlan.Entities

{

// ...

[DisplayName("教师用书"), InstanceName("教师用书")]

public sealed class TeacherWholeDataRow : Row, IIdRow, INameRow

{

[DisplayName("申报编号"), Column("DeclarationID"), Identity]

public Int32? DeclarationId

{

get { return Fields.DeclarationId[this]; }

set { Fields.DeclarationId[this] = value; }

}

[DisplayName("学期编号"), Size(20), NotNull]

public String TermCode

{

get { return Fields.TermCode[this]; }

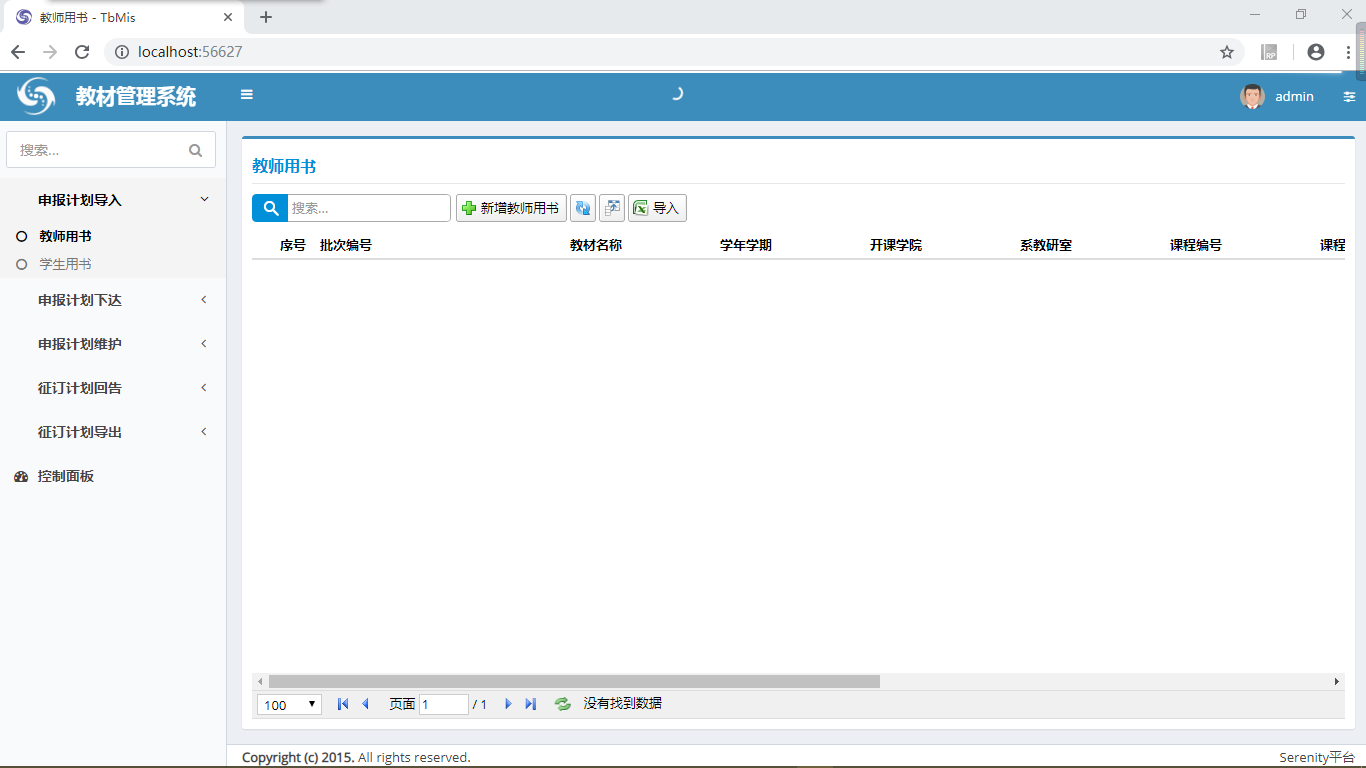
set { Fields.TermCode[this] = value; }

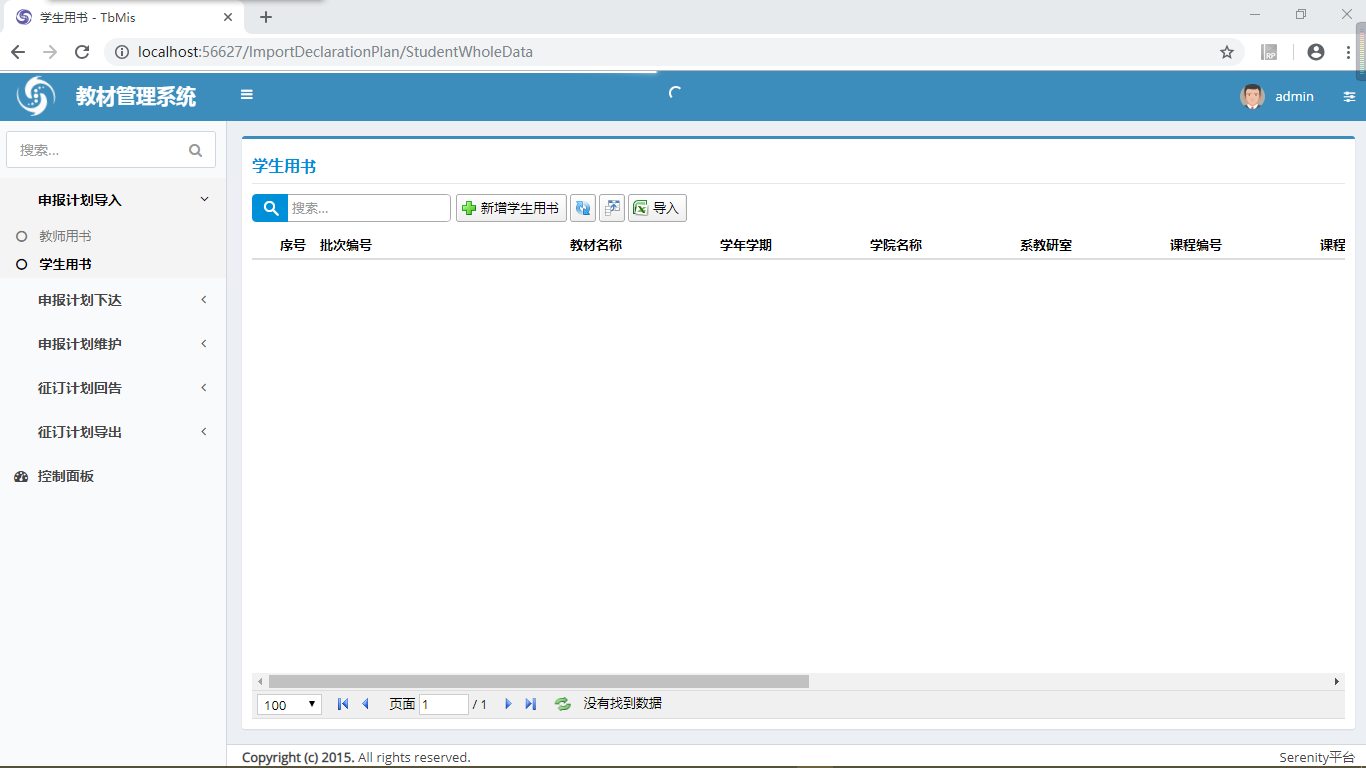
}

// ...

}

现在生成并运行应用程序，你会在网格列表和对话框中看到字段标题的变化：





**1.4 重写列的标题和宽度**

找到Modules下的ImportDeclarationData，在TeacherWholeData下的TeacherWholeDataColumns.cs文件进行修改：  
namespace TbMis.ImportDeclarationPlan.Columns

{

// ...

public class TeacherWholeDataColumns

{

[Width(40),DisplayName("Db.Shared.RecordId"), AlignRight]

public Int32 DeclarationId { get; set; }

[Width(250)]

public String BatchId { get; set; }

// ...

}

}

同时，根据需要显示的属性进行修改并调整位置，不需要显示在页面上的属性将其删除，而对于课程编号和名称、教材名称则需要加上：

public String TextbookNum2TextbookName { get; set; }

public String CourseNumCourseCode { get; set; }

public String CourseNumCourseName { get; set; }

**1.5 添加一个FileName（文件名）字段**

由于要实现导入功能，需要一个导入按钮，一个对话框，一个文件名的字段，以及导入功能操作的实现，因此：

找到Modules下的ImportDeclarationData，在TeacherWholeData下创建一个TeacherInfoExcelImportForm.cs，这是由于表单中字段的显示/编辑是在Form中控制的，在其中写入如下代码：

namespace TbMis.ImportDeclarationPlan.Forms

{

using Serenity.ComponentModel;

using Serenity.Web;

using System;

[FormScript("ImportDeclarationPlan.TeacherInfoExcelImport")]

public class TeacherInfoExcelImportForm

{

[FileUploadEditor, Required]

public String FileName { get; set; }

}

}

**1.6 添加教师用书和学生用书导入计划对话框**

找到Modules下的ImportDeclarationData，在TeacherWholeData下创建一个TypeScript文件----TeacherInfoExcelImportDialog.ts，在其中写入如下代码：

namespace TbMis.ImportDeclarationPlan {

@Serenity.Decorators.registerClass()

export class TeacherInfoExcelImportDialog extends Serenity.PropertyDialog<any, any> {

private form: TeacherInfoExcelImportForm;

constructor() {

super();

this.form = new TeacherInfoExcelImportForm(this.idPrefix);

}

protected getDialogButtons(): Serenity.DialogButton[] {

return [

{

text: '导入',

click: () => {

if (!this.validateBeforeSave())

return;

if (this.form.FileName.value == null ||

Q.isEmptyOrNull(this.form.FileName.value.Filename)) {

Q.notifyError("请选择文件!");

return;

}

},

},

{

text: '取消',

click: () => this.dialogClose()

}

];

}

}

}

同样，对于学生用书也是一样。

此时，在TeacherInfoExcelImportForm和StudentTeacherInfoExcelImportForm处会报错，我们只需打开生成->重新生成解决方案，然后转换所有T4模板即可。

**1.7 添加导入计划按钮**

我们已经创建好了一个对话框，但是并没有进入对话框的按钮，因此：

找到Modules下的ImportDeclarationData，在TeacherWholeData下的TeacherWholeDataGrid.ts的文件中添加如下代码：

namespace TbMis.ImportDeclarationPlan {

// ...

/\*\*

\* get a button of ExcelImport.

\*/

protected getButtons(): Serenity.ToolButton[] {

// call base method to get list of buttons

var buttons = super.getButtons();

// add our import button

buttons.push({

title: '导入',

cssClass: 'export-xlsx-button',

onClick: () => {

// open import dialog, let it handle rest

var dialog = new TeacherInfoExcelImportDialog();

dialog.element.on('dialogclose', () => {

this.refresh();

dialog = null;

});

dialog.dialogOpen();

}

});

return buttons;

}

}

同样，对于学生用书也只要在StudentWholeData下的StudentWholeDataGrid.ts的文件

中增加如上代码，但要将代码中的TeacherInfoExcelImportDialog()改为StudentInfoExcelImportDialog()即可。

**1.8 实现导入的功能**

找到Modules下的ImportDeclarationData，在TeacherWholeData下创建一个TeacherInfoExcelImportEndpoint.cs，在其中写入如下代码：(以教师用书为例)

namespace TbMis.ImportDeclarationPlan.Endpoints

{

using CourseInfo.Entities;

using CourseInfo.Repositories;

using OfficeOpenXml;

using Repositories;

using Serenity;

using Serenity.Data;

using Serenity.Services;

using Serenity.Web;

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Web.Mvc;

using TextBookInfo.Entities;

using TextBookInfo.Repositories;

using MyRow = Entities.TeacherWholeDataRow;

[RoutePrefix("Services/ImportDeclarationPlan/TeacherInfoExcelImport"), Route("{action}")]

[ConnectionKey(typeof(MyRow)), ServiceAuthorize]

public class TeacherInfoExcelImportController : ServiceEndpoint

{

[HttpPost]

public ExcelImportResponse TeacherExcelImport(IUnitOfWork uow, ExcelImportRequest request)

{

request.CheckNotNull();

Check.NotNullOrWhiteSpace(request.FileName, "filename");

UploadHelper.CheckFileNameSecurity(request.FileName);

if (!request.FileName.StartsWith("temporary/"))

throw new ArgumentOutOfRangeException("filename");

ExcelPackage ep = new ExcelPackage();

using (var fs = new FileStream(UploadHelper.DbFilePath(request.FileName), FileMode.Open, FileAccess.Read))

ep.Load(fs);

var t = MyRow.Fields;

var response = new ExcelImportResponse();

response.ErrorList = new List<string>();

var worksheet = ep.Workbook.Worksheets[1];

for (var row = 2; row <= worksheet.Dimension.End.Row; row++)

{

try

{

//TeacherWholeData

var batchId = Convert.ToString(worksheet.Cells[row, 3].Value ?? "");

if (batchId.IsTrimmedEmpty())

continue;

var declaration = uow.Connection.TryFirst<MyRow>(q1 => q1

.Select(t.DeclarationId)

.Where(t.DeclarationId == -1));

if (declaration == null)

{

declaration = new MyRow

{

BatchId = batchId

};

}

else

{

declaration.TrackWithChecks = false;

}

declaration.TermCode = Convert.ToString(worksheet.Cells[row, 1].Value ?? "");

declaration.TermName = Convert.ToString(worksheet.Cells[row, 2].Value ?? "");

declaration.SchoolId = Convert.ToString(worksheet.Cells[row, 4].Value ?? "");

declaration.SchoolName = Convert.ToString(worksheet.Cells[row, 5].Value ?? "");

declaration.DepartmentId = Convert.ToString(worksheet.Cells[row, 6].Value ?? "");

declaration.DepartmentName = Convert.ToString(worksheet.Cells[row, 7].Value ?? "");

declaration.EducationalType = Convert.ToString(worksheet.Cells[row, 9].Value ?? "");

declaration.ApprovedAmount = Convert.ToInt32(worksheet.Cells[row, 21].Value ?? 0);

declaration.Priority = Convert.ToString(worksheet.Cells[row, 22].Value ?? "");

declaration.Phone = Convert.ToString(worksheet.Cells[row, 23].Value ?? "");

declaration.Remarks = Convert.ToString(worksheet.Cells[row, 24].Value ?? "");

declaration.CheckState = Convert.ToString(worksheet.Cells[row, 25].Value ?? "");

declaration.DataSign = Convert.ToString(worksheet.Cells[row, 26].Value ?? "");

if (declaration.DeclarationId == null)

{

new TeacherWholeDataRepository().Create(uow, new SaveRequest<MyRow>

{

Entity = declaration

});

response.Inserted = response.Inserted + 1;

}

else

{

new TeacherWholeDataRepository().Update(uow, new SaveRequest<MyRow>

{

Entity = declaration,

EntityId = declaration.DeclarationId.Value

});

response.Updated = response.Updated + 1;

}

}

catch (Exception ex)

{

response.ErrorList.Add("Exception on Row " + row + ": " + ex.Message);

}

}

return response;

}

}

}

学生用书也是一样，只要修改对应的地方即可。

此时，我们需要能够点击导入能实现如上将Excel表中的数据导入对应的数据库中，因此，我们需要在刚刚创建的TeacherInfoExcelImportDialog.ts中加入如下代码：

namespace TbMis.ImportDeclarationPlan {

// ...

TeacherInfoExcelImportService.TeacherExcelImport({

FileName: this.form.FileName.value.Filename

}, response => {

Q.notifyInfo(

'导入: ' + (response.Inserted || 0) +

', 更新: ' + (response.Updated || 0));

if (response.ErrorList != null && response.ErrorList.length > 0) {

Q.notifyError(response.ErrorList.join(',\r\n '));

}

this.dialogClose();

});

}

打开Modules/ImportDeclarationPlan/TeacherWholeData/TeacherWholeDataRepository.cs,进行如下修改：

namespace TbMis.ImportDeclarationPlan.Repositories

{

// ...

public class TeacherWholeDataRepository

{

// ...

private class MySaveHandler : SaveRequestHandler<MyRow, SaveRequest<MyRow>, SaveResponse>

{

protected override void AfterSave()

{

base.AfterSave();

if (Request.Localizations != null)

foreach (var pair in Request.Localizations)

{

pair.Value.DeclarationId = Row.DeclarationId.Value;

new LocalizationRowHandler<MyRow>().Update<MyRow>(this.UnitOfWork, pair.Value, Convert.ToInt32(pair.Key));

}

}

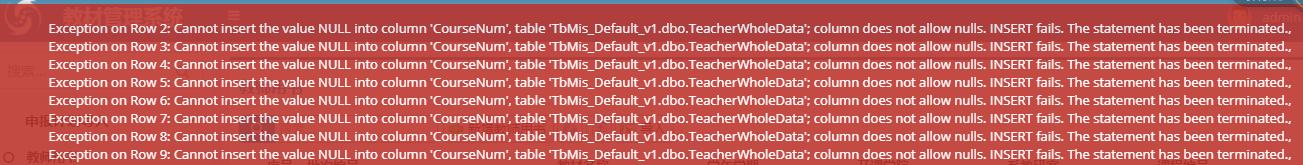
}

}

}

由于我们在应用程序中添加了一个新实体，我们应该在生成解决方案之后运行T4 模板。

到此，基本实现了导入功能，但是此时运行会发现导入时会报错：



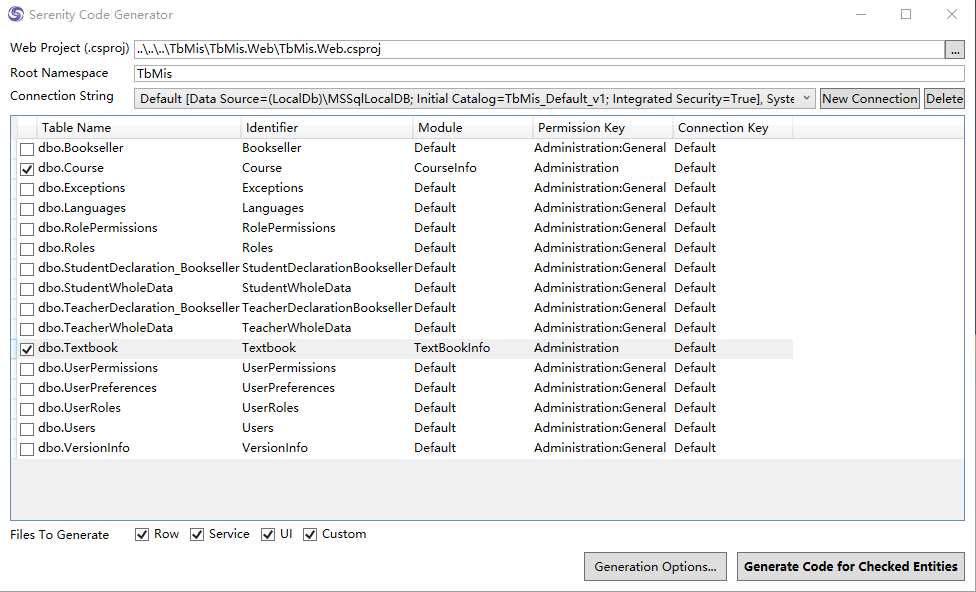
这是由于CourseNum和TextbookNum2这两个外键不能为空，此时，我们只是将数据导入两张申报计划表之中，并没有将对应数据导入至课程表和教材表中，因此：

**1.9 为课程(Course)表和教材(Textbook)表生成代码**

我们将使用Serenity代码生成器（sergen.exe）生成初始的可编辑界面。

在Visual Studio中，通过点击视图=>其它窗体=>包管理器控制台打开包管理器控制台。

输入sergen并按Enter 键确认。



然后，在TeacherWholeData/TeacherInfoExcelImportEndpoint.cs中加入如下代码：

namespace TbMis.ImportDeclarationPlan.Endpoints

{

// ...

public class TeacherInfoExcelImportController : ServiceEndpoint

{

[HttpPost]

public ExcelImportResponse TeacherExcelImport(IUnitOfWork uow, ExcelImportRequest request)

{

// ...

var b = TextbookRow.Fields;

var c = CourseRow.Fields;

for (var row = 2; row <= worksheet.Dimension.End.Row; row++)

{

try

{

//Textbook

var textbookID = Convert.ToString(worksheet.Cells[row, 12].Value ?? "");

if (textbookID.IsTrimmedEmpty())

continue;

var textbook = uow.Connection.TryFirst<TextbookRow>(q => q

.Select(b.TextbookNum2)

.Where(b.TextbookId == textbookID));

if (textbook == null)

{

textbook = new TextbookRow

{

TextbookId = textbookID

};

}

else

{

textbook.TrackWithChecks = false;

}

textbook.TextbookName = Convert.ToString(worksheet.Cells[row, 13].Value ?? "");

textbook.Author = Convert.ToString(worksheet.Cells[row, 14].Value ?? "");

textbook.Isbn = Convert.ToString(worksheet.Cells[row, 15].Value ?? "");

textbook.Press = Convert.ToString(worksheet.Cells[row, 16].Value ?? "");

textbook.Edition = Convert.ToString(worksheet.Cells[row, 17].Value ?? "");

textbook.PrintingCount = Convert.ToString(worksheet.Cells[row, 18].Value ?? "");

textbook.TextbookType = Convert.ToString(worksheet.Cells[row, 19].Value ?? "");

textbook.Price = Convert.ToString(worksheet.Cells[row, 20].Value ?? "");

textbook.IsSelfCompile = Convert.ToString("0");

if (textbook.TextbookNum2 == null)

{

new TextbookRepository().Create(uow, new SaveRequest<TextbookRow>

{

Entity = textbook

});

}

else

{

new TextbookRepository().Update(uow, new SaveRequest<TextbookRow>

{

Entity = textbook,

EntityId = textbook.TextbookNum2.Value

});

}

//Course

var courseId = Convert.ToString(worksheet.Cells[row, 8].Value ?? "");

if (courseId.IsTrimmedEmpty())

continue;

var course = uow.Connection.TryFirst<CourseRow>(q => q

.Select(c.CourseNum)

.Where(c.CourseId == courseId));

if (course == null)

{

course = new CourseRow

{

CourseId = courseId

};

}

else

{

course.TrackWithChecks = false;

}

course.CourseCode = Convert.ToString(worksheet.Cells[row, 10].Value ?? "空");

course.CourseName = Convert.ToString(worksheet.Cells[row, 11].Value ?? "");

if (course.CourseNum == null)

{

new CourseRepository().Create(uow, new SaveRequest<CourseRow>

{

Entity = course

});

}

else

{

new CourseRepository().Update(uow, new SaveRequest<CourseRow>

{

Entity = course,

EntityId = course.CourseNum.Value

});

}

declaration.CourseNum = course.CourseNum.Value;

declaration.TextbookNum2 = textbook.TextbookNum2.Value;

}

}

}

}

}

打开Modules/CourseInfo/Course/CourseRepository.cs,进行如下修改：

namespace TbMis.CourseInfo.Repositories

{

// ...

public class CourseRepository

{

// ...

private class MySaveHandler : SaveRequestHandler<MyRow, SaveRequest<MyRow>, SaveResponse>

{

protected override void AfterSave()

{

base.AfterSave();

if (Request.Localizations != null)

foreach (var pair in Request.Localizations)

{

pair.Value.CourseNum = Row.CourseNum.Value;

new LocalizationRowHandler<CourseRow>().Update<Entities.CourseRow>(this.UnitOfWork, pair.Value, Convert.ToInt32(pair.Key));

}

}

}

}

}

打开Modules/TextBookInfo/Textbook/TextbookRepository.cs,进行如下修改：

namespace TbMis.TextBookInfo.Repositories

{

// ...

public class TextbookRepository

{

// ...

private class MySaveHandler : SaveRequestHandler<MyRow, SaveRequest<MyRow>, SaveResponse>

{

protected override void AfterSave()

{

base.AfterSave();

if (Request.Localizations != null)

foreach (var pair in Request.Localizations)

{

pair.Value.TextbookNum2 = Row.TextbookNum2.Value;

new LocalizationRowHandler<TextbookRow>().Update<Entities.TextbookRow>(this.UnitOfWork, pair.Value, Convert.ToInt32(pair.Key));

}

}

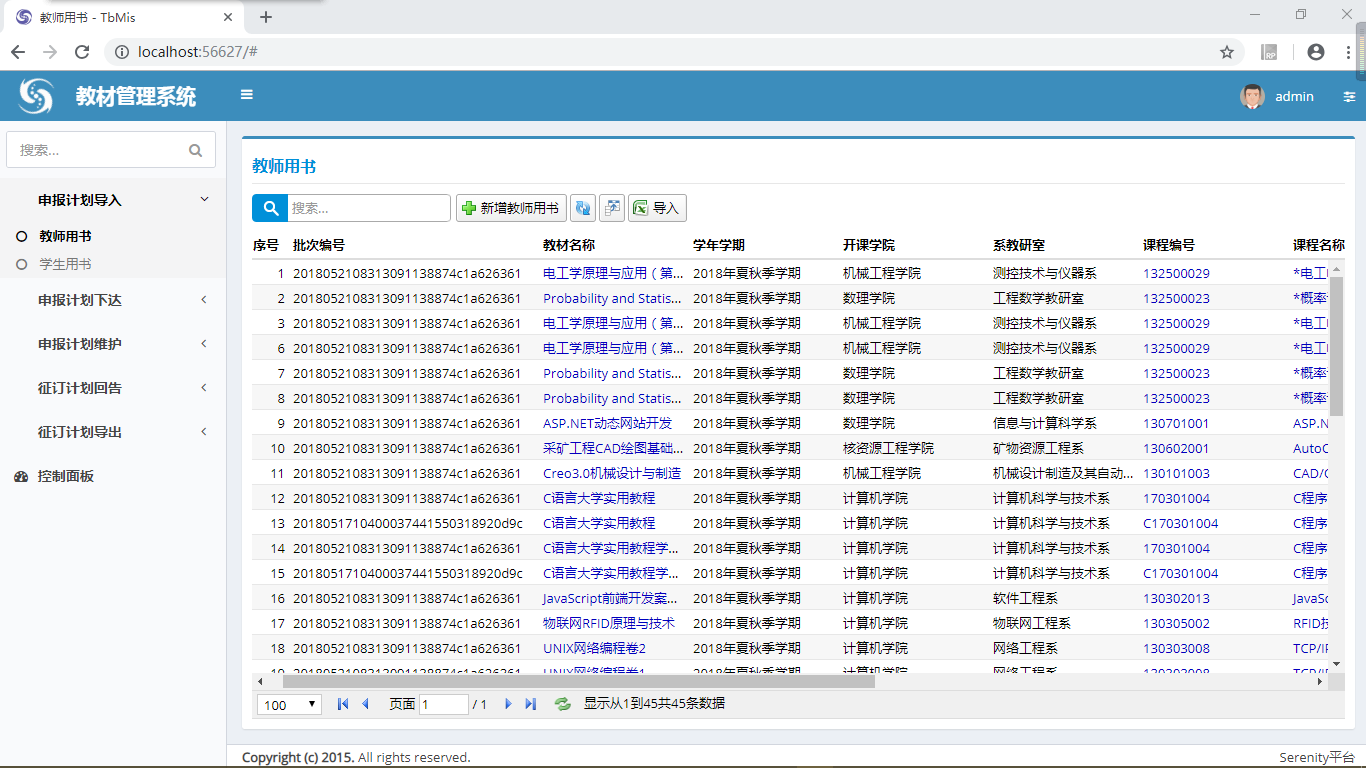
}

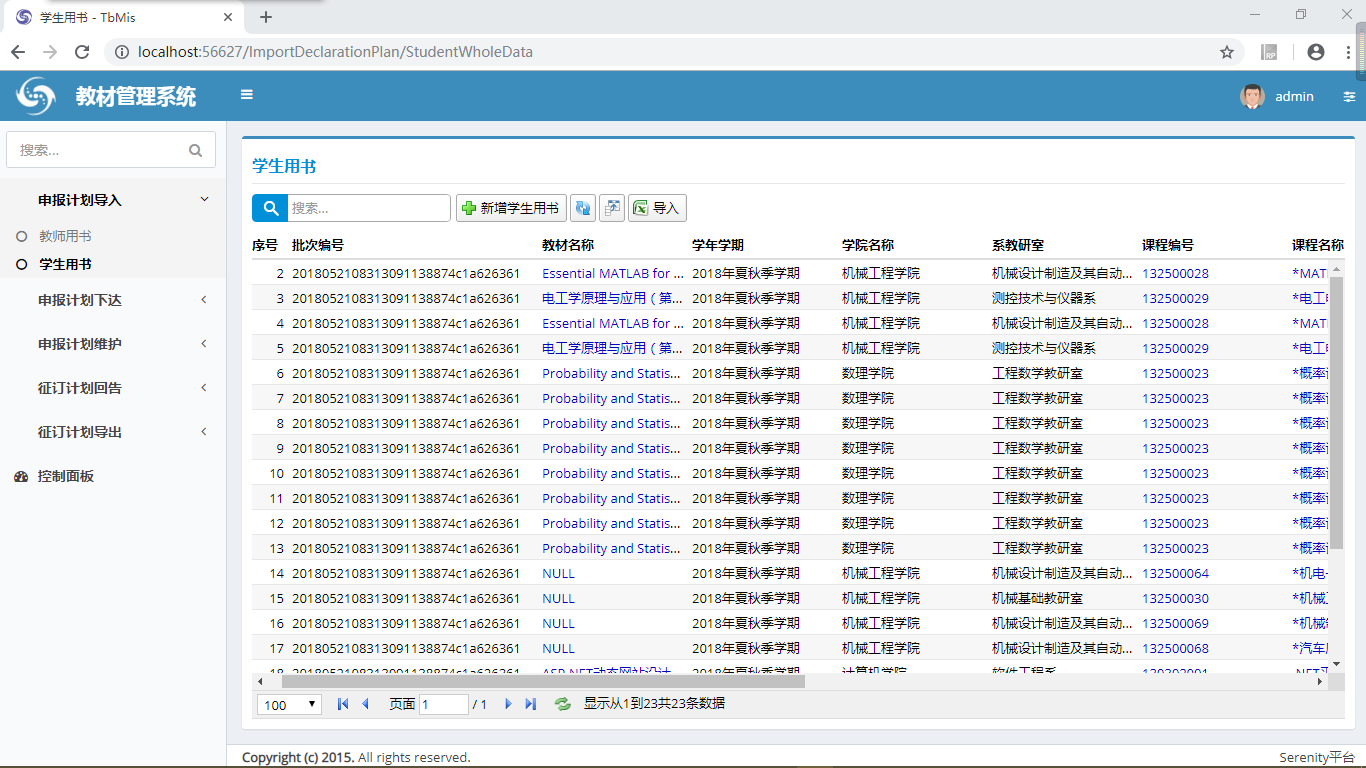
}

}

此时重新运行，点击导入->选择文件->导入,最终所有数据导入至了TeacherWholeData、StudentWholeData、Course、Textbook四张表中。

运行结果如下：





**1.10 显示课程信息**

将对话框的所有属性设为只可读以及删除所有按钮

打开Modules/CourseInfo/Course/CourseDialog.ts文件：加入如下代码：

namespace TbMis.CourseInfo {

@Serenity.Decorators.registerClass()

export class CourseDialog extends Serenity.EntityDialog<CourseRow, any> {

// ...

protected getToolbarButtons(): Serenity.ToolButton[] {

let buttons = super.getToolbarButtons();

buttons.splice(Q.indexOf(buttons, x => x.cssClass == "save-and-close-button"), 1);

buttons.splice(Q.indexOf(buttons, x => x.cssClass == "apply-changes-button"), 1);

return buttons;

}

protected updateInterface(): void {

super.updateInterface();

Serenity.EditorUtils.setReadonly(this.element.find('.editor'), true);

// remove required asterisk (\*)

this.element.find('sup').hide();

this.deleteButton.hide();

}

protected getEntityTitle(): string {

if (!this.isEditMode()) {

return "课程信息";

}

else {

var entityType = super.getEntitySingular();

// get name field value of record this dialog edits

let name = this.getEntityNameFieldValue() || "";

return 'View ' + entityType + " (" + name + ")";

}

}

protected updateTitle(): void {

super.updateTitle();

}

}

}

如何使得单击课程编号或名称能够显示相应的信息，(以教师用书为例)只需在Modules/ImportDeclarationPlan/TeacherWholeData/TeacherWholeDataColumns.cs中加入如下代码即可：

namespace TbMis.ImportDeclarationPlan.Columns

{

// ...

public class TeacherWholeDataColumns

{

// ...

[Width(150),EditLink(ItemType = "CourseInfo.Course")]

public String CourseNumCourseCode { get; set; }

[Width(150),EditLink(ItemType = "CourseInfo.Course")]

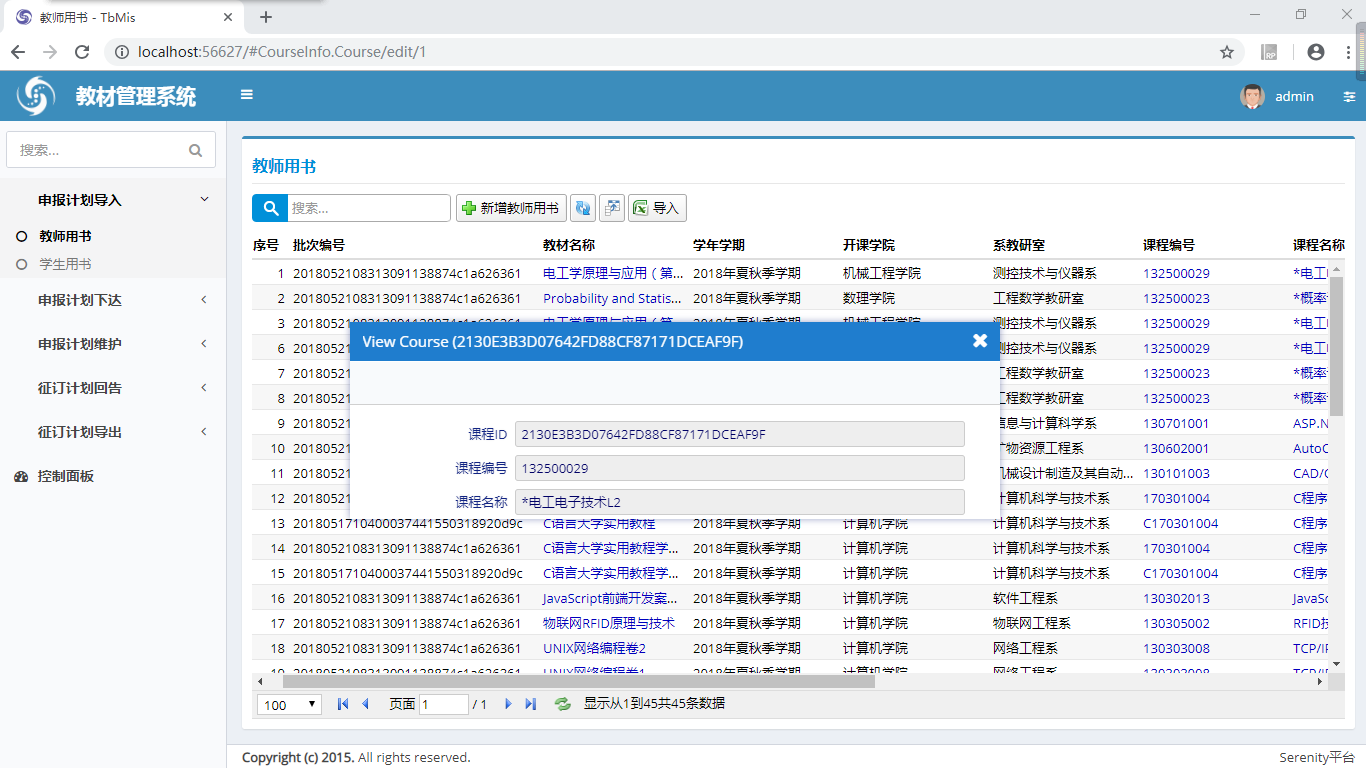
public String CourseNumCourseName { get; set; }

// ...

}

}

最后，重新运行：



**1.11 显示教材信息**

将对话框的所有属性设为只可读以及删除所有按钮

namespace TbMis.TextBookInfo {

@Serenity.Decorators.registerClass()

export class TextbookDialog extends Serenity.EntityDialog<TextbookRow, any> {

// ...

protected getToolbarButtons(): Serenity.ToolButton[] {

let buttons = super.getToolbarButtons();

buttons.splice(Q.indexOf(buttons, x => x.cssClass == "save-and-close-button"), 1);

buttons.splice(Q.indexOf(buttons, x => x.cssClass == "apply-changes-button"), 1);

return buttons;

}

protected updateInterface(): void {

super.updateInterface();

Serenity.EditorUtils.setReadonly(this.element.find('.editor'), true);

// remove required asterisk (\*)

this.element.find('sup').hide();

this.deleteButton.hide();

}

protected getEntityTitle(): string {

if (!this.isEditMode()) {

return "教材信息";

}

else {

var entityType = super.getEntitySingular();

// get name field value of record this dialog edits

let name = this.getEntityNameFieldValue() || "";

return 'View ' + entityType + " (" + name + ")";

}

}

protected updateTitle(): void {

super.updateTitle();

}

}

}

如何使得单击教材名称能够显示相应的信息，(以教师用书为例)只需在Modules/ImportDeclarationPlan/TeacherWholeData/TeacherWholeDataColumns.cs中加入如下代码即可：

namespace TbMis.ImportDeclarationPlan.Columns

{

// ...

public class TeacherWholeDataColumns

{

// ...

[Width(150), EditLink(ItemType = "TextBookInfo.Textbook")]

public String TextbookNum2TextbookName { get; set; }

// ...

}

}

最后，重新运行：



**1.12 添加删除按钮**

在申报计划列的首端添加一列删除按钮，使得能够删除对应列的申报计划信息。

打开Modules/ImportDeclarationPlan/TeacherWholeData/TeacherWholeDataGrid.ts中加入如下代码即可(以教师用书为例)：

namespace TbMis.ImportDeclarationPlan {

@Serenity.Decorators.registerClass()

export class TeacherWholeDataGrid extends Serenity.EntityGrid<TeacherWholeDataRow, any> {

// ...

/\*\*

\* get a column of delete buttons

\*/

protected getColumns() {

var column = super.getColumns();

column.unshift({

field: 'Delete Row',

name: '',

format: ctx => '<a class="inline-action delete-row" title="delete">' +

'<i class="fa fa-trash-o text-red"></i></a>',

width: 24,

minWidth: 24,

maxWidth: 24

});

return column;

}

protected onClick(e: JQueryEventObject, row: number, cell: number) {

super.onClick(e, row, cell);

if (e.isDefaultPrevented())

return;

var item = this.itemAt(row);

var target = $(e.target);

// if user clicks "i" element, e.g. icon

if (target.parent().hasClass('inline-action'))

target = target.parent();

if (target.hasClass('inline-action')) {

e.preventDefault();

if (target.hasClass('delete-row')) {

Q.confirm('Delete record?', () => {

ImportDeclarationPlan.TeacherWholeDataService.Delete({

EntityId: item.DeclarationId,

}, response => {

this.refresh();

});

});

}

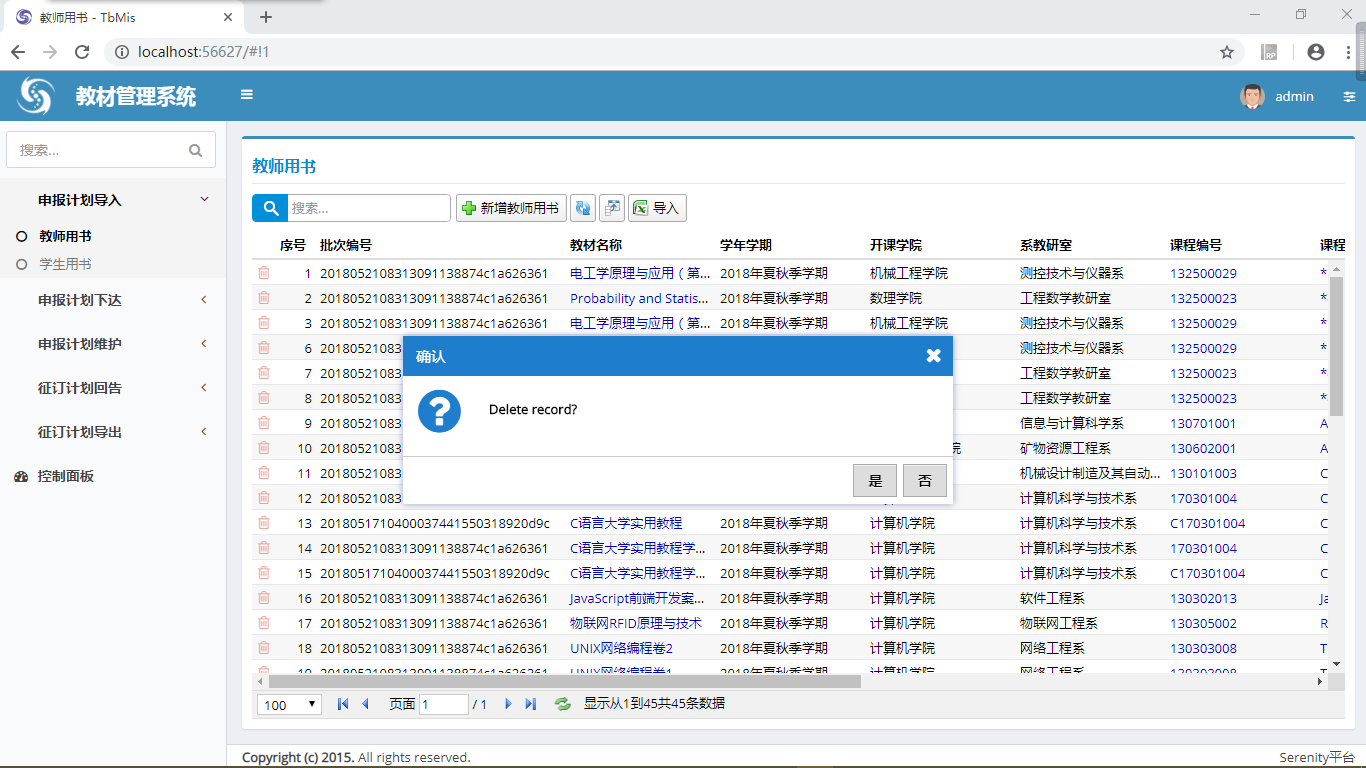
}

}

}

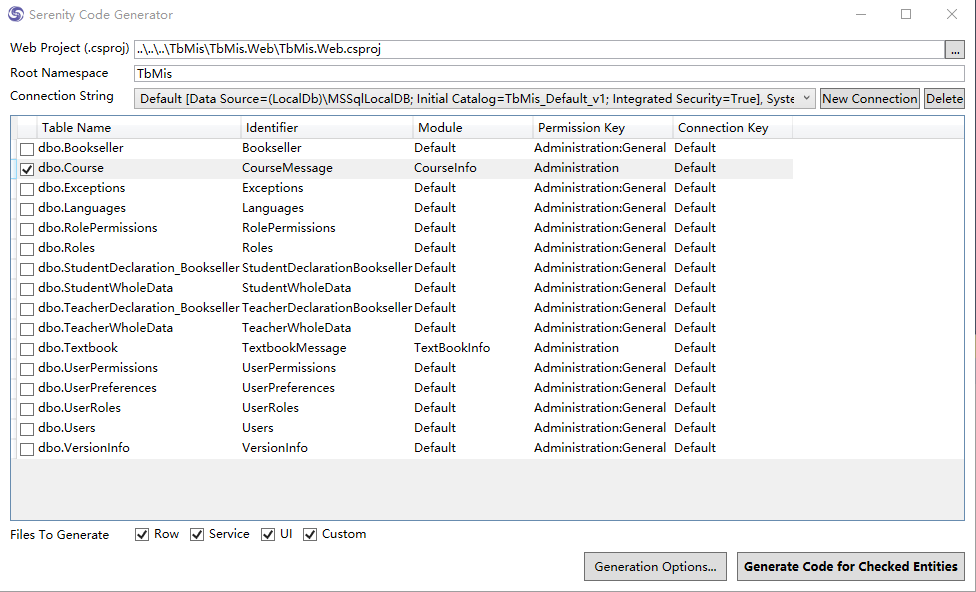
}

重新运行发现申报计划信息列的首端多了一个删除图标：

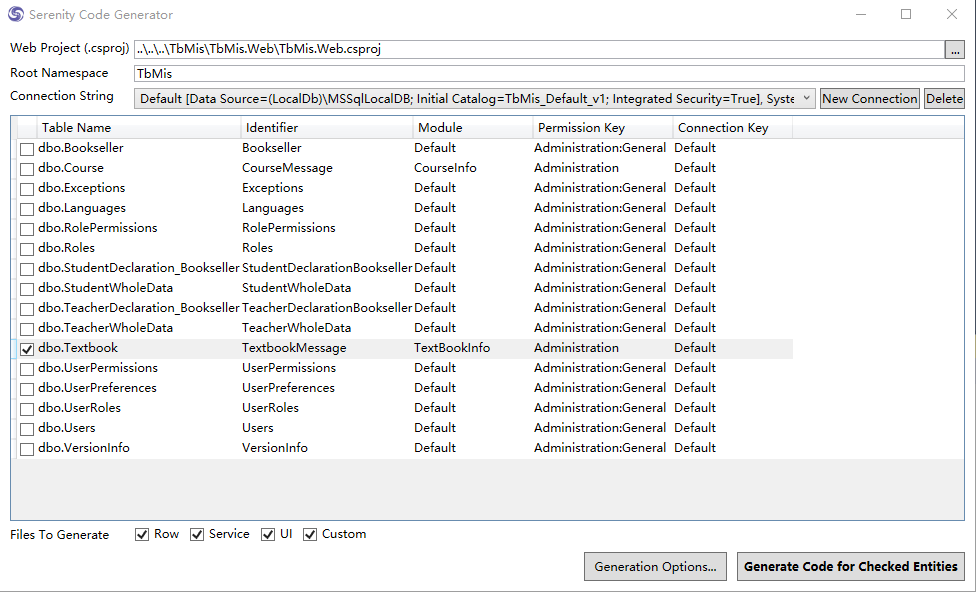


**1.13 在新增教师/学生用书中添加课程、教材**

再次为Course表生成代码



再次为Textbook表生成代码



**1.14 为课程（CourseMessage）和教材（TextbookMessage）声明一个检索脚本（LookupScript）**

打开Modules/CourseInfo/CourseMessage/CourseMessageRow.cs，在其中添加如下代码：

namespace TbMis.CourseInfo.Entities

{

// ...

[ConnectionKey("Default"), Module("CourseInfo"), TableName("[dbo].[Course]")]

[DisplayName("Course Message"), InstanceName("Course Message")]

[ReadPermission("Administration")]

[ModifyPermission("Administration")]

[LookupScript]

// ...

}

打开Modules/TextBookInfo/TextbookMessage/TextbookMessageRow.cs，在其中添加如下代码：

namespace TbMis.TextBookInfo.Entities

{

// ...

[ConnectionKey("Default"), Module("TextBookInfo"), TableName("[dbo].[Textbook]")]

[DisplayName("Textbook Message"), InstanceName("Textbook Message")]

[ReadPermission("Administration")]

[ModifyPermission("Administration")]

[LookupScript]

// ...

}

**1.15 对课程和教材使用LookupEditor**

打开Modules/ImportDeclarationPlan/TeacherWholeData/TeacherWholeDataRow.cs，在其中添加如下代码：

namespace TbMis.ImportDeclarationPlan.Entities

{

// ...

public sealed class TeacherWholeDataRow : Row, IIdRow, INameRow

{

// ...

[DisplayName("教材名称"), ForeignKey("[dbo].[Textbook]", "TextbookNum2"), LeftJoin("jTextbookNum2"), TextualField("TextbookNum2TextbookId")]

[LookupEditor(typeof(TextbookMessageRow), InplaceAdd = true)]

public Int32? TextbookNum2

{

get { return Fields.TextbookNum2[this]; }

set { Fields.TextbookNum2[this] = value; }

}

[DisplayName("课程信息"), NotNull, ForeignKey("[dbo].[Course]", "CourseNum"), LeftJoin("jCourseNum"), TextualField("CourseNumCourseId")]

[LookupEditor(typeof(CourseMessageRow), InplaceAdd = true)]

public Int32? CourseNum

{

get { return Fields.CourseNum[this]; }

set { Fields.CourseNum[this] = value; }

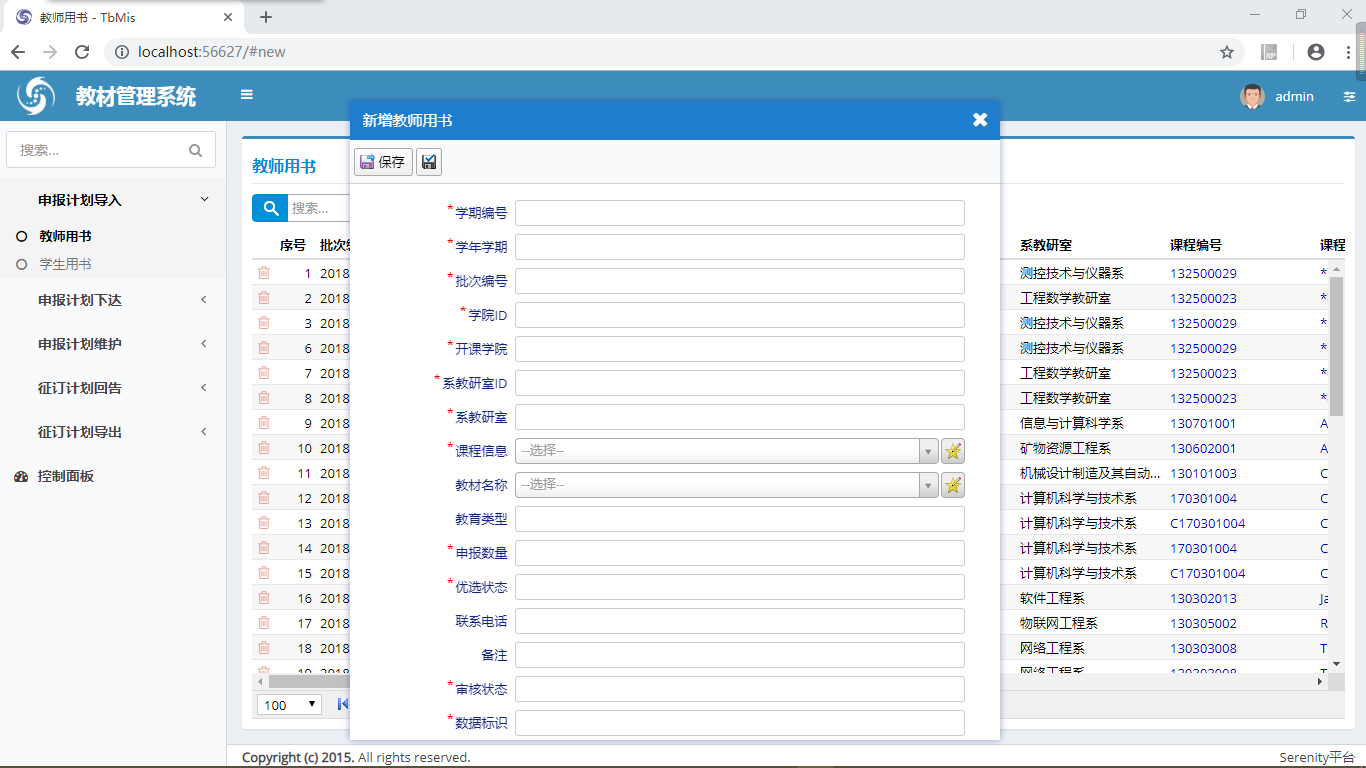
}

// ...

}

}

此时，运行点开新增教师用书发现在课程信息和教材信息处多了一个选择下拉框和按钮：



但是此时的选择下拉框里显示的是课程ID和教材ID，如何让它显示课程名称和教材名称，我们可以在Modules/CourseInfo/CourseMessage/CourseMessageRow.cs中进行修改：

StringField INameRow.NameField

{

get { return Fields.CourseName; }

}

在Modules/TextBookInfo/TextbookMessage/TextbookMessageRow.cs中改为：

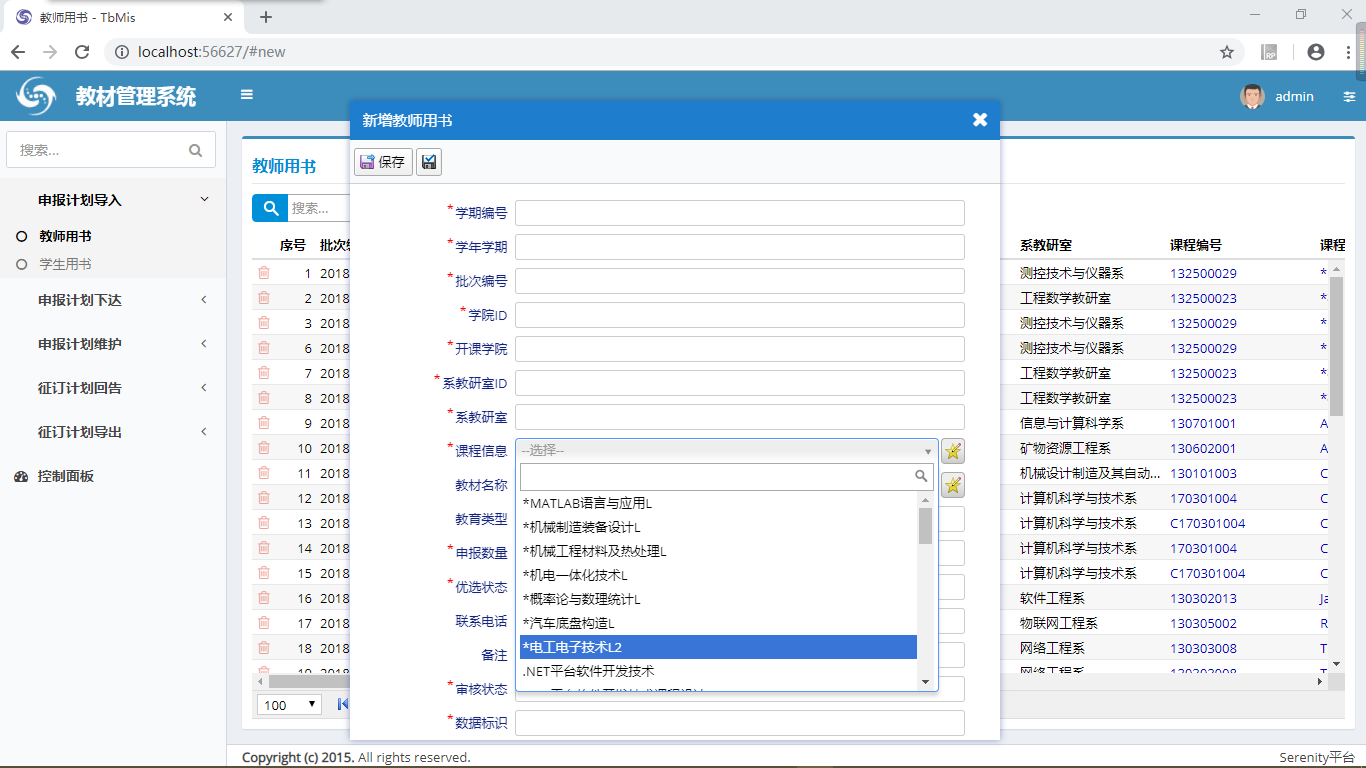
StringField INameRow.NameField

{

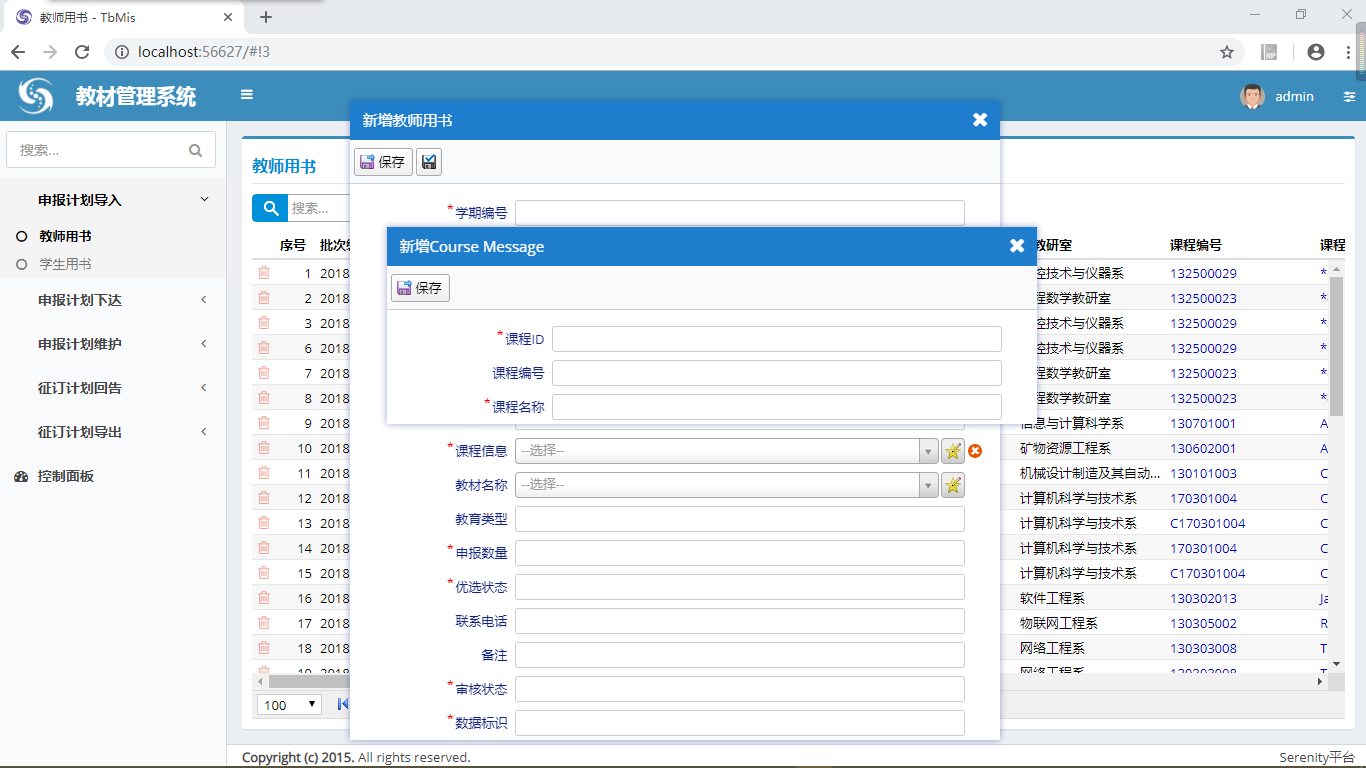
get { return Fields.TextbookName; }

}

此时，则可显示名称了：



现在，我们可以点击课程（Course）字段或教材（Textbook）字段旁边的图标添加新的课程或教材：



**1.16 处理导航**

将CourseInfo和TextBookInfo的导航项删除

打开Modules/CourseInfo/CourseInfoNavigation.cs,将其中这两行代码注释：

[assembly: NavigationLink(int.MaxValue, "CourseInfo/Course", typeof(MyPages.CourseController), icon: null)]

[assembly: NavigationLink(int.MaxValue, "CourseInfo/Course Message", typeof(MyPages.CourseMessageController), icon: null)]

打开Modules/TextBookInfo/TextBookInfoNavigation.cs,将其中这两行代码注释：

[assembly: NavigationLink(int.MaxValue, "TextBookInfo/Textbook", typeof(MyPages.TextbookController), icon: null)]

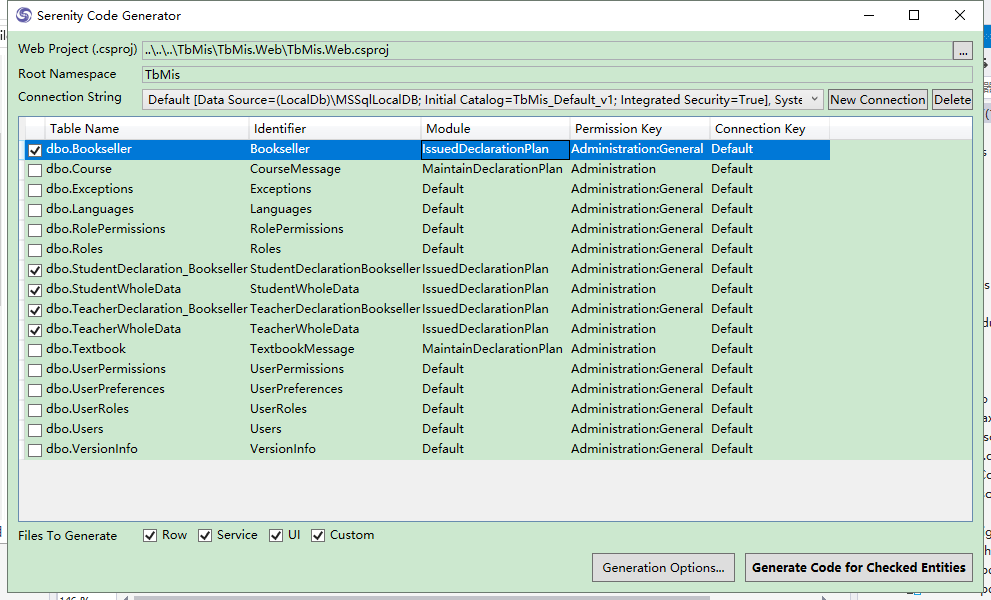
[assembly: NavigationLink(int.MaxValue, "TextBookInfo/Textbook Message", typeof(MyPages.TextbookMessageController), icon: null)]

最终，在导航栏处只会显示申报计划导入了。

**二、申报计划下达**

**2.1 为相关表生成代码**

再次使用程序包管理器控制台打开 sergen.exe，并为 Genre 表设置生成代码的参数，如下所示：



**2.2 处理导航界面**

重新生成解决方案后，我们将该文件的内容

Modules/IsuuedDeclarationPlan/IsuuedDeclarationPlanNavigation.cs修改为：

*若文件夹中无此文件，请选择IssuedDeclarationPlan文件夹右击添加->类，添加IssuedDeclarationPlanNavigation.cs文件*

//Add these

using Serenity.Navigation;

using MyPages = TbMis.IssuedDeclarationPlan.Pages;

[assembly: NavigationLink(2, "申报计划下达/教师用书", typeof(MyPages.TeacherDeclarationBooksellerController), icon: null)]

[assembly: NavigationLink(2, "申报计划下达/学生用书", typeof(MyPages.StudentDeclarationBooksellerController), icon: null)]

**2.3 设置界面标题**

为了设置界面的标题，我们在

Modules/IsuuedDeclarationPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerRow.cs

以及

Modules/IsuuedDeclarationPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerRow.cs

文件中修改以下内容：

// This...

[DisplayName("学生用书"), InstanceName("学生用书")]

//This...

[DisplayName("教师用书"), InstanceName("教师用书")]

**2.4 设置属性显示格式**

为了设置属性以中文的格式显示，我们在

Modules/IsuuedDeclarationPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerRow.cs

以及

Modules/IsuuedDeclarationPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerRow.cs

文件中将相应的属性修改为以下形式：

//This...

[DisplayName("征订状态")]

[Size(1), NotNull]

public String SubscriptionStatus

{

get { return Fields.SubscriptionStatus[this]; }

set { Fields.SubscriptionStatus[this] = value; }

}

**2.5 设置表格显示内容**

我们根据在表格所需要显示的内容，在

Modules/IsuuedDeclarationPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerForm.cs

以及

Modules/IsuuedDeclarationPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerForm.cs

文件中修改以下内容：

//Add these

public String SchoolName { get; set; }

public Int32 DeclarationId { get; set; }

public String BooksellerName { get; set; }

public String BooksellerId { get; set; }

public String SubscriptionStatus { get; set; }

public DateTime FeedbackDate { get; set; }

public String FeedbackStaff { get; set; }

public String Remark { get; set; }

public String ReDeclareStatus { get; set; }

**2.6 设置编辑框显示内容**

我们根据在表格所需要显示的内容，在

Modules/IsuuedDeclarationPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerColumns.cs

以及

Modules/IsuuedDeclarationPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerColumns.cs

文件中修改以下内容：

//Add These

[EditLink]

public Int32 DeclarationId { get; set; }

public String SchoolName { get; set; }

public String BooksellerId { get; set; }

public String DeclarationTermName { get; set; }

public String BooksellerName { get; set; }

**2.7 生成检索脚本**

在新增界面我们需要将学院名称，申报编号，书商名称，书商编号设置可选择形式，所以我们将这部分属性编辑框设置为检索脚本形式，具体实现步骤如下：

此处我们以学生用书为例，教师用书可参照该部分修改：

①：在Modules/IsuuedDeclarationPlan/Bookseller中添加BooksellerNameLookup.cs文件，其中将文件内容修改为：

//Add these

using Serenity.ComponentModel;

using Serenity.Data;

using Serenity.Web;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using TbMis.IssuedDeclarationPlan.Entities;

namespace TbMis.Lookups

{

[LookupScript]

public class BooksellerNameLookup : RowLookupScript<BooksellerRow>

{

public BooksellerNameLookup()

{

IdField = TextField = "BooksellerName";

}

protected override void PrepareQuery(SqlQuery query)

{

var fld = BooksellerRow.Fields;

query.Distinct(true)

.Select(fld.BooksellerName)

.Where(

new Criteria(fld.BooksellerName) != "" &

new Criteria(fld.BooksellerName).IsNotNull());

}

protected override void ApplyOrder(SqlQuery query)

{

}

}

}

②：在Modules/IsuuedDeclarationPlan/Bookseller中添加BooksellerIdLookup.cs文件，其中将文件内容修改为：

//Add these

using Serenity.ComponentModel;

using Serenity.Data;

using Serenity.Web;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using TbMis.IssuedDeclarationPlan.Entities;

namespace TbMis.Lookups

{

[LookupScript]

public class BooksellerIdLookup : RowLookupScript<BooksellerRow>

{

public BooksellerIdLookup()

{

IdField = TextField = BooksellerRow.Fields.BooksellerId.PropertyName;

}

protected override void PrepareQuery(SqlQuery query)

{

var fld = BooksellerRow.Fields;

query.Distinct(true)

.Select(fld.BooksellerName)

.Select(fld.BooksellerId)

.Where(

new Criteria(fld.BooksellerName) != "" &

new Criteria(fld.BooksellerName).IsNotNull() &

new Criteria(fld.BooksellerId) != "" &

new Criteria(fld.BooksellerId).IsNotNull());

}

protected override void ApplyOrder(SqlQuery query)

{

}

}

}

③：在Modules/IsuuedDeclarationPlan/StudentWholeData中添加DeclarationIdLookup.cs文件，其中将文件内容修改为：

//Add these

using Serenity.ComponentModel;

using Serenity.Data;

using Serenity.Web;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using TbMis.ImportDeclarationPlan.Entities;

namespace TbMis.Student.Lookups

{

[LookupScript]

public class DeclarationIdLookup : RowLookupScript<StudentWholeDataRow>

{

public DeclarationIdLookup()

{

IdField = TextField = StudentWholeDataRow.Fields.DeclarationId.PropertyName;

}

protected override void PrepareQuery(SqlQuery query)

{

var fld = StudentWholeDataRow.Fields;

query.Distinct(true)

.Select(fld.SchoolName)

.Select(fld.DeclarationId)

.Select(fld.TextbookNum2TextbookName)

.Where(

new Criteria(fld.SchoolName) != "" &

new Criteria(fld.SchoolName).IsNotNull() &

new Criteria(fld.DeclarationId) != "" &

new Criteria(fld.DeclarationId).IsNotNull() &

new Criteria(fld.TextbookNum2TextbookName) != "NULL" &

new Criteria(fld.TextbookNum2TextbookName).IsNotNull());

}

protected override void ApplyOrder(SqlQuery query)

{

}

}

}

④：在Modules/IsuuedDeclarationPlan/StudentWholeData中添加SchoolNameLookup.cs文件，其中将文件内容修改为：

//Add these

using Serenity.ComponentModel;

using Serenity.Data;

using Serenity.Web;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using TbMis.ImportDeclarationPlan.Entities;

namespace TbMis.Student.Lookups

{

[LookupScript]

public class SchoolNameLookup : RowLookupScript<StudentWholeDataRow>

{

public SchoolNameLookup()

{

IdField = TextField = "SchoolName";

}

protected override void PrepareQuery(SqlQuery query)

{

var fld = StudentWholeDataRow.Fields;

query.Distinct(true)

.Select(fld.SchoolName)

.Where(

new Criteria(fld.SchoolName) != "" &

new Criteria(fld.SchoolName).IsNotNull());

}

protected override void ApplyOrder(SqlQuery query)

{

}

}

}

⑤：在

Modules/IsuuedDeclarationPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerRow.cs中，我们将相应属性的内容修改名称：

*若不修改名称，将会出现相关属性查找不到的情况*

//Add these

public StringField SchoolName;

public Int32Field DeclarationId;

public StringField BooksellerName;

public StringField BooksellerId;

此后我们将相应的属性内容与编辑脚本绑定：

//Add these

[DisplayName("申报编号"), Column("DeclarationID"), PrimaryKey, NotNull ,ForeignKey("[dbo].[StudentWholeData]", "DeclarationID"), LeftJoin("jDeclaration"), TextualField("DeclarationTermCode")]

[LookupEditor(typeof(Student.Lookups.DeclarationIdLookup), CascadeFrom = "SchoolName", AutoComplete = true)]

public Int32? DeclarationId

{

get { return Fields.DeclarationId[this]; }

set { Fields.DeclarationId[this] = value; }

}

[DisplayName("书商编号"), Column("BooksellerID"), Size(50), NotNull, ForeignKey("[dbo].[Bookseller]", "BooksellerID"), LeftJoin("jBookseller"), QuickSearch, TextualField("BooksellerBooksellerName")]

[LookupEditor(typeof(Lookups.BooksellerIdLookup), CascadeFrom = "BooksellerName", AutoComplete = true)]

public String BooksellerId

{

get { return Fields.BooksellerId[this]; }

set { Fields.BooksellerId[this] = value; }

}

[DisplayName("学生学院"), Expression("jDeclaration.[SchoolName]"), NotNull]

[LookupEditor(typeof(Student.Lookups.SchoolNameLookup), AutoComplete = true)]

public String SchoolName

{

get { return Fields.SchoolName[this]; }

set { Fields.SchoolName[this] = value; }

}

[DisplayName("书商名称"), Expression("jBookseller.[BooksellerName]"), NotNull]

[LookupEditor(typeof(Lookups.BooksellerNameLookup), AutoComplete = true)]

public String BooksellerName

{

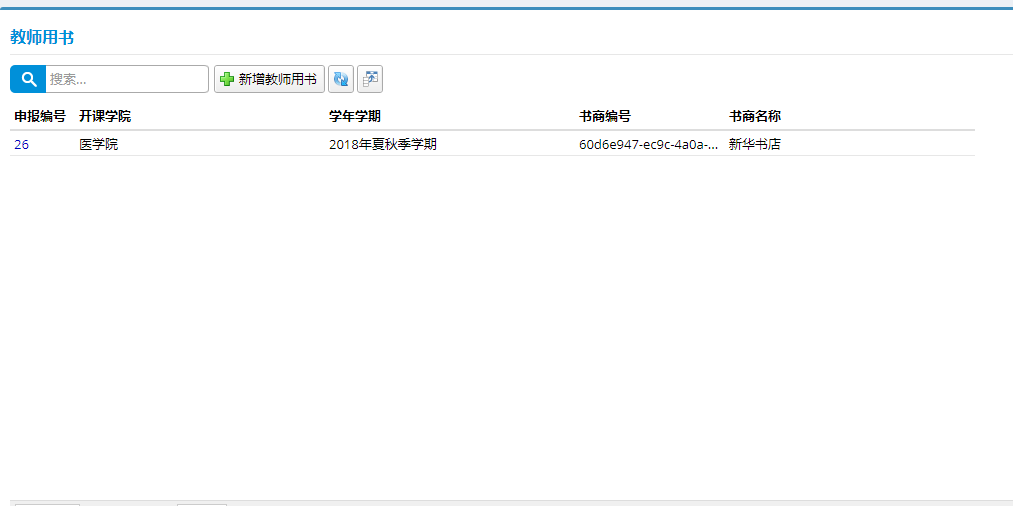
get { return Fields.BooksellerName[this]; }

set { Fields.BooksellerName[this] = value; }

}

**2.8 最终内容显示**

在完成以上步骤之后，我们就得到了该模块的所有内容，界面实现如下：







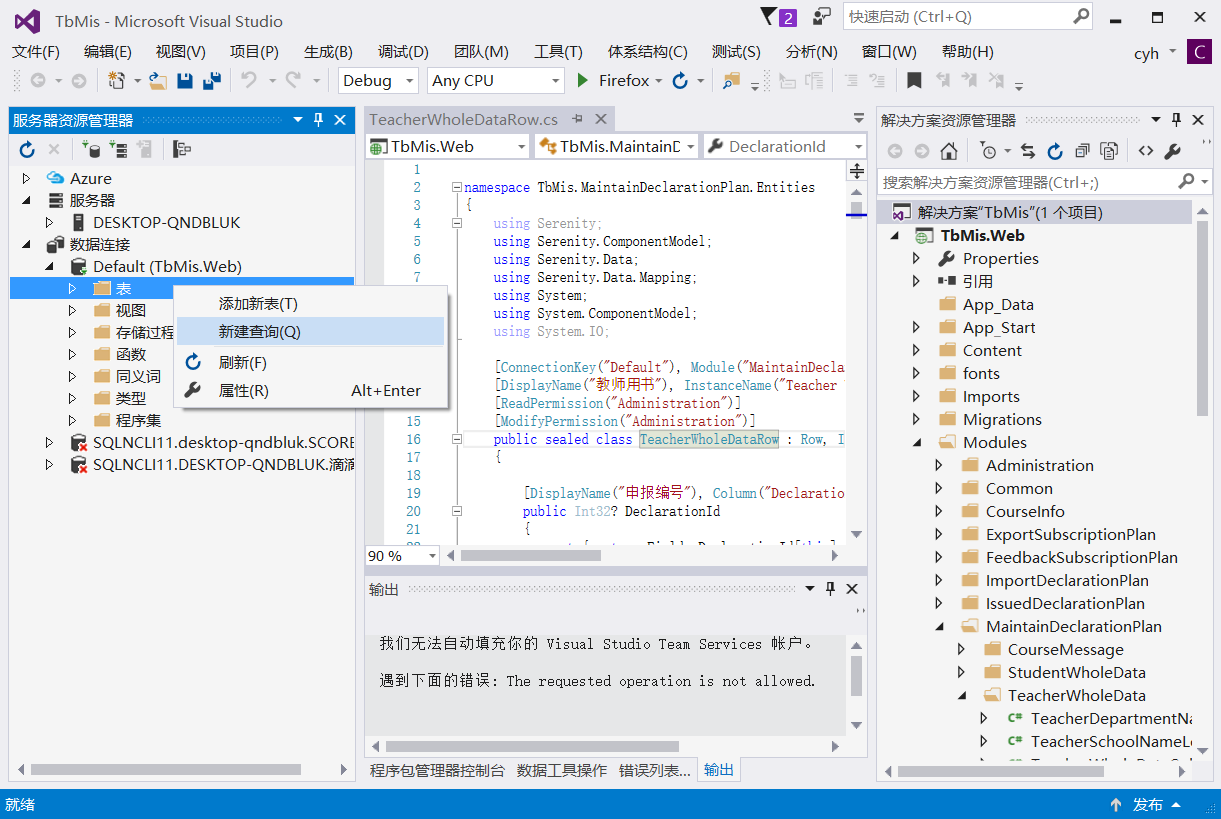
**三、申报计划维护**

**3.1 创建教师申报计划（TeacherWholeData）表**

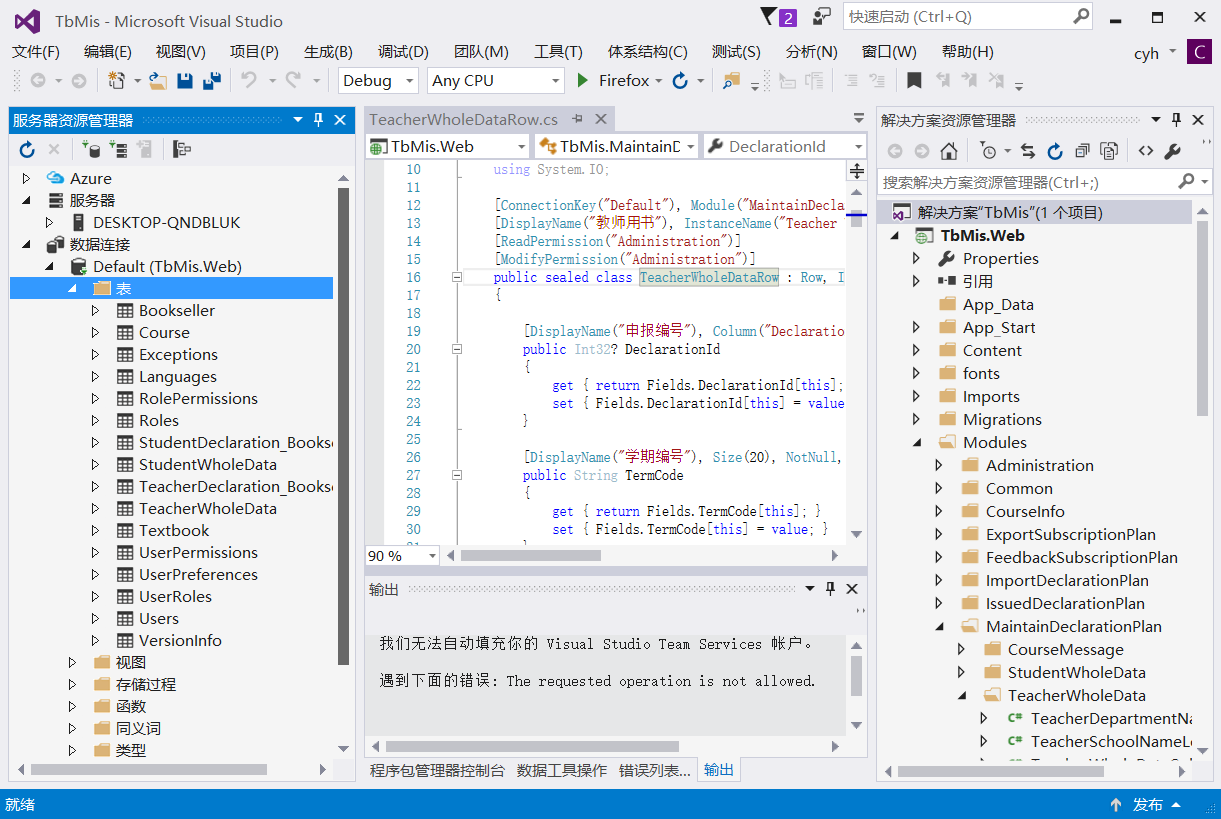
要存储申报计划，我们需要一张申报计划表（TeacherWholeData）我们需要将在sql中写好的脚本导入到文件中。

//这里我们没有采用写迁移类来实现表和数据构建，我们采用我们表熟悉的数据库sqlsever2012构建好表和导入数据，在导出生成脚本，通过新建查询来的方式创建表。

在服务器资源管理器中导航到Default(TbMis.Web)/表，右键单击表新建查询，把我们写好的脚本复制过去。



将数据库脚本运行后刷新表，就可以实现把表导进去了

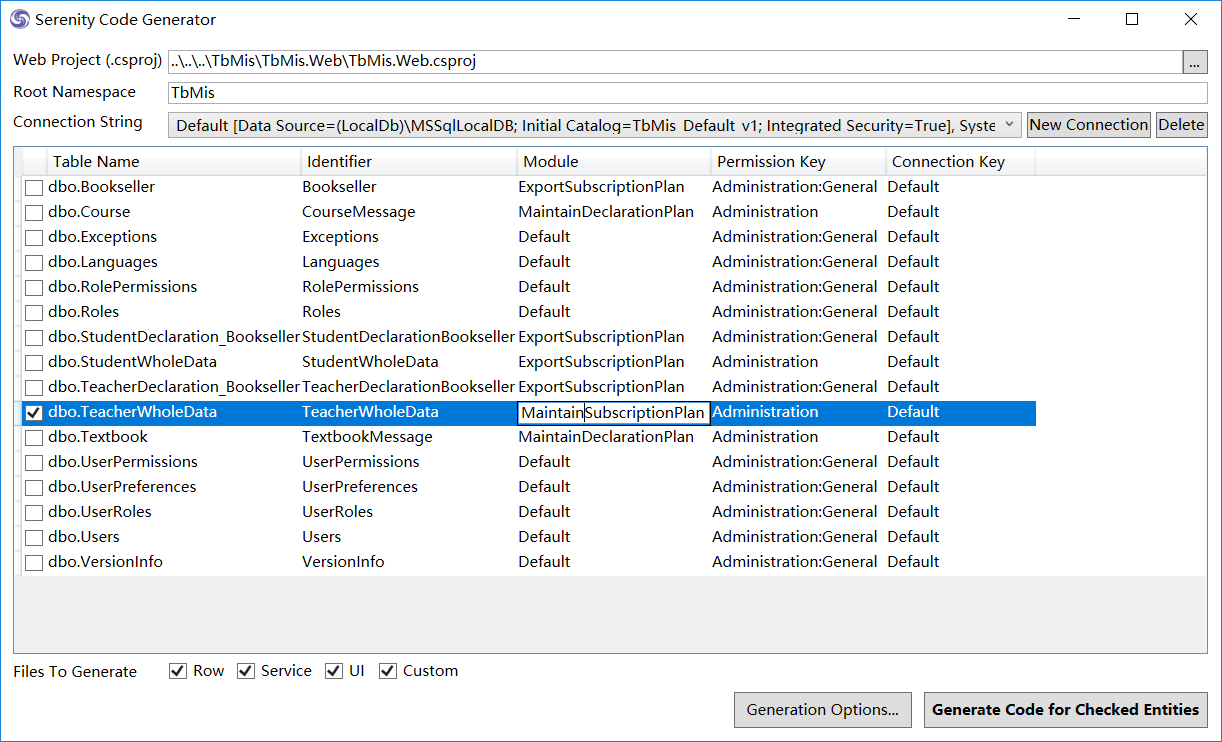


我们的脚本和数据文件是下面这样的



为教师计划申报计划表（TeacherWholeData）生成代码

确保表存到数据库，我们将使用 Serenity 代码生成器（sergen.exe）生成初始的可编辑界面。在 Visual Studio 中，通过点击视图 => 其它窗体 => 程序包管理器控制台打开包管理器控制台。输入sergen并按 Enter 键确认。



Connection String :

点击下拉框选择Default，选择后就会出现该数据库中表的信息

Identifier:

可以修改生成后的表文件的名字

Module:

在 Serenity 术语中，一个模块（module）是一组逻辑页面（pages），共享同一个目标。为教师计划申报计划表（TeacherWholeData）生成代码例如，在 Serene 模板中，所有与MaintainDeclarationPlan相关的页面都属于MaintainDeclarationPlan模块。

学生的申报计划维护表也是采用同样的方法进行实现。

**3.2 自定义教师申报计划维护表界面**

修改显示在导航目录下显示的名字

我们在MaintainDeclarationPlanNavigation文件中将导航信息修改为“申报计划维护/教师用书”

using Serenity.Navigation;

using MyPages = TbMis.MaintainDeclarationPlan.Pages;

[assembly: NavigationLink(3, "申报计划维护/教师用书", typeof(MyPages.TeacherWholeDataController), icon: null)]

[assembly: NavigationLink(3, "申报计划维护/学生用书", typeof(MyPages.StudentWholeDataController), icon: null)]

修改页面左上角的名字

我们找到TeacherWholeDataRow将DisplayName中的名字修改为教师用书

namespace TbMis.MaintainDeclarationPlan.Entities

{

//...

[ConnectionKey("Default"), Module("MaintainDeclarationPlan"), TableName("[dbo].[TeacherWholeData]")]

[DisplayName("教师用书"), InstanceName("Teacher Whole Data")]

[ReadPermission("Administration")]

[ModifyPermission("Administration")]

//...

}

修改列显示的名字：

我们找到TeacherWholeDataRow将属性上的DisplayName改成自己希望显示的名字，这里一DeclarationId申报ID为例

[DisplayName("申报编号"), Column("DeclarationID"), Identity]

public Int32? DeclarationId

{

get { return Fields.DeclarationId[this]; }

set { Fields.DeclarationId[this] = value; }

}

将属性在列中中显示出来

这个功能我们在TeacherWholeDataColumns中实现，我们将要展示的属性按顺序写好，按格式写上代码就可以在列中显示出来

public class TeacherWholeDataColumns

{

[EditLink, DisplayName("Db.Shared.RecordId"), AlignRight]

public Int32 DeclarationId { get; set; }

[Width(150), EditLink(ItemType = "MaintainDeclarationPlan.TextbookMessage")]

public String TextbookNum2TextbookName { get; set; }

public String TermName { get; set; } [LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherSchoolNameLookup)), QuickFilter(CssClass = "hidden-xs")]

public String SchoolName { get; set; } [LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherDepartmentNameLookup))]

[QuickFilter(CssClass = "hidden-xs"), QuickFilterOption("cascadeFrom", "SchoolName")]

public String DepartmentName { get; set; }

[Width(150), EditLink(ItemType = "MaintainDeclarationPlan.CourseMessage")]

public String CourseNumCourseCode { get; set; }

[Width(150), EditLink(ItemType = "MaintainDeclarationPlan.CourseMessage")]

public String CourseNumCourseName { get; set; }

public String Phone { get; set; }

public String CheckState { get; set; }

public String Priority { get; set; }

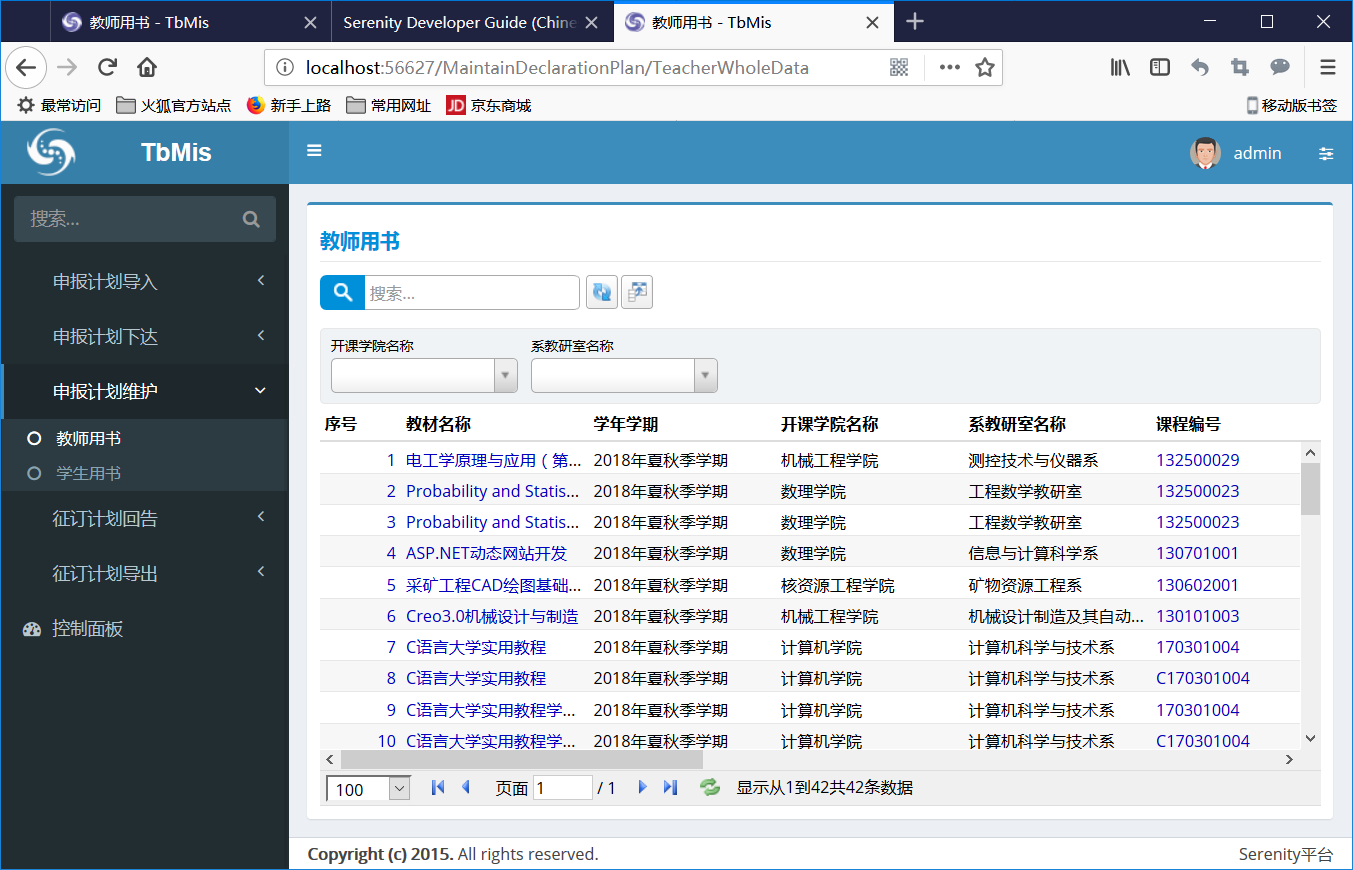
public String DataSign { get; set; }

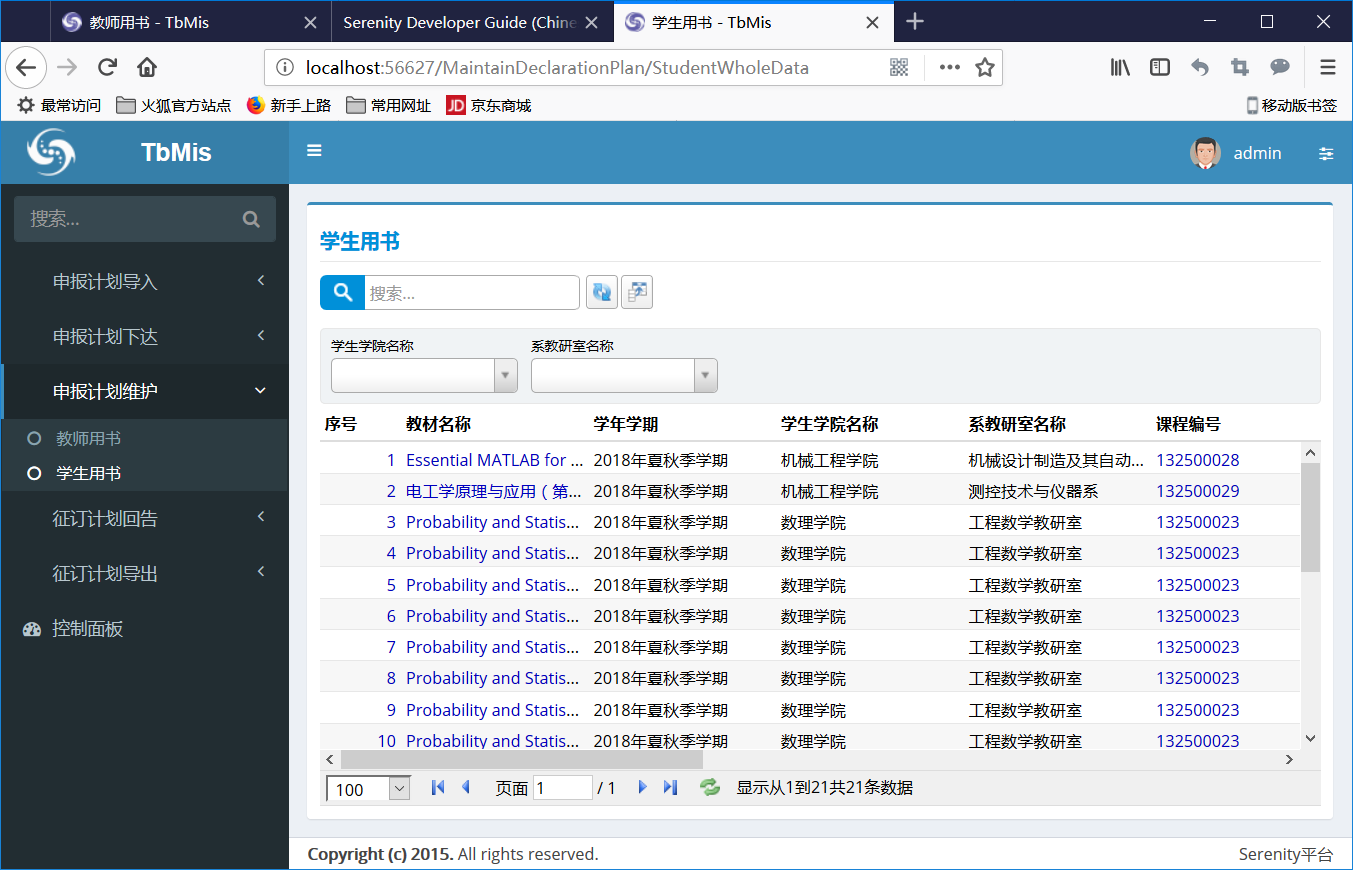
public Int32 ApprovedAmount { get; set; }

public String Remarks { get; set; }

}

学生的申报计划维护表也是采用同样的方法进行实现。修改好后我们可以运行一下我们的程序可以看到我们界面的样子





**3.3 筛选学院的申报计划**

我们MaintainDeclarationPlan/TeacherWholeData目录下创建一个新的类项，名为TeacherSchoolNameLookup，来实现我们筛选。

代码如下：

namespace TbMis.Modules.MaintainDeclarationPlan.Lookups

{

using Serenity.ComponentModel;

using Serenity.Data;

using Serenity.Web;

using TbMis.MaintainDeclarationPlan.Entities;

[LookupScript]

public class TeacherSchoolNameLookup : RowLookupScript<TeacherWholeDataRow>

{

public TeacherSchoolNameLookup()

{

IdField = TextField = TeacherWholeDataRow.Fields.SchoolName.PropertyName;

}

protected override void PrepareQuery(SqlQuery query)

{

var fld = TeacherWholeDataRow.Fields;

query.Distinct(true)

.Select(fld.SchoolName)

.Where(

new Criteria(fld.SchoolName) != "" &

new Criteria(fld.SchoolName).IsNotNull());

}

protected override void ApplyOrder(SqlQuery query)

{

}

}

}

接着让我们在TeacherWholeDataColumns中做修改：

[LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherSchoolNameLookup)), QuickFilter(CssClass = "hidden-xs")]

public String SchoolName { get; set; }

//在SchoolName这条语句上写上[LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherSchoolNameLookup)), QuickFilter(CssClass = "hidden-xs")]语句就可以实现一级的筛选了

**3.4 二级教研室筛选**

二级的教研室筛选是在学院筛选的基础上进行的，我们同样先在MaintainDeclarationPlan/TeacherWholeData目录下创建一个新的类项，名为TeacherDepartmentNameLookup，来实现我们筛选。代码如下：

//二级筛选与一级的不同的地方在于Select是由条件的，所以写了两条。

namespace TbMis.Modules.MaintainDeclarationPlan.Lookups

{

using Serenity.ComponentModel;

using Serenity.Data;

using Serenity.Web;

using TbMis.MaintainDeclarationPlan.Entities;

[LookupScript]

public class TeacherDepartmentNameLookup : RowLookupScript<TeacherWholeDataRow>

{

public TeacherDepartmentNameLookup()

{

IdField = TextField = TeacherWholeDataRow.Fields.DepartmentName.PropertyName;

}

protected override void PrepareQuery(SqlQuery query)

{

var fld = TeacherWholeDataRow.Fields;

query.Distinct(true)

.Select(fld.SchoolName)

.Select(fld.DepartmentName)

.Where(

new Criteria(fld.SchoolName) != "" &

new Criteria(fld.SchoolName).IsNotNull() &

new Criteria(fld.DepartmentName) != "" &

new Criteria(fld.DepartmentName).IsNotNull());

}

protected override void ApplyOrder(SqlQuery query)

{

}

}

}

接着让我们在TeacherWholeDataColumns中做修改：

[LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherDepartmentNameLookup))]

[QuickFilter(CssClass = "hidden-xs"), QuickFilterOption("cascadeFrom", "SchoolName")]

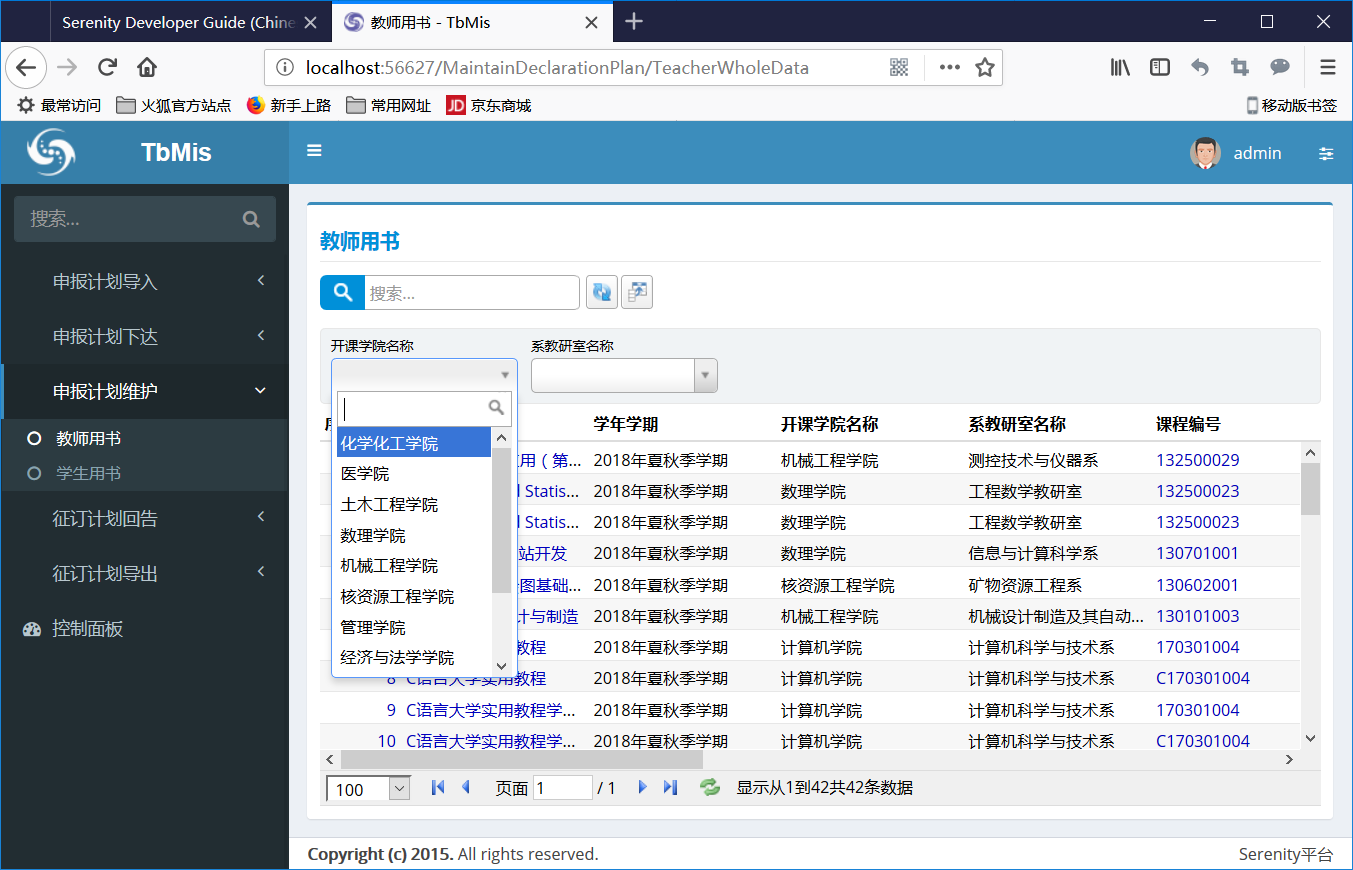
public String DepartmentName { get; set; }

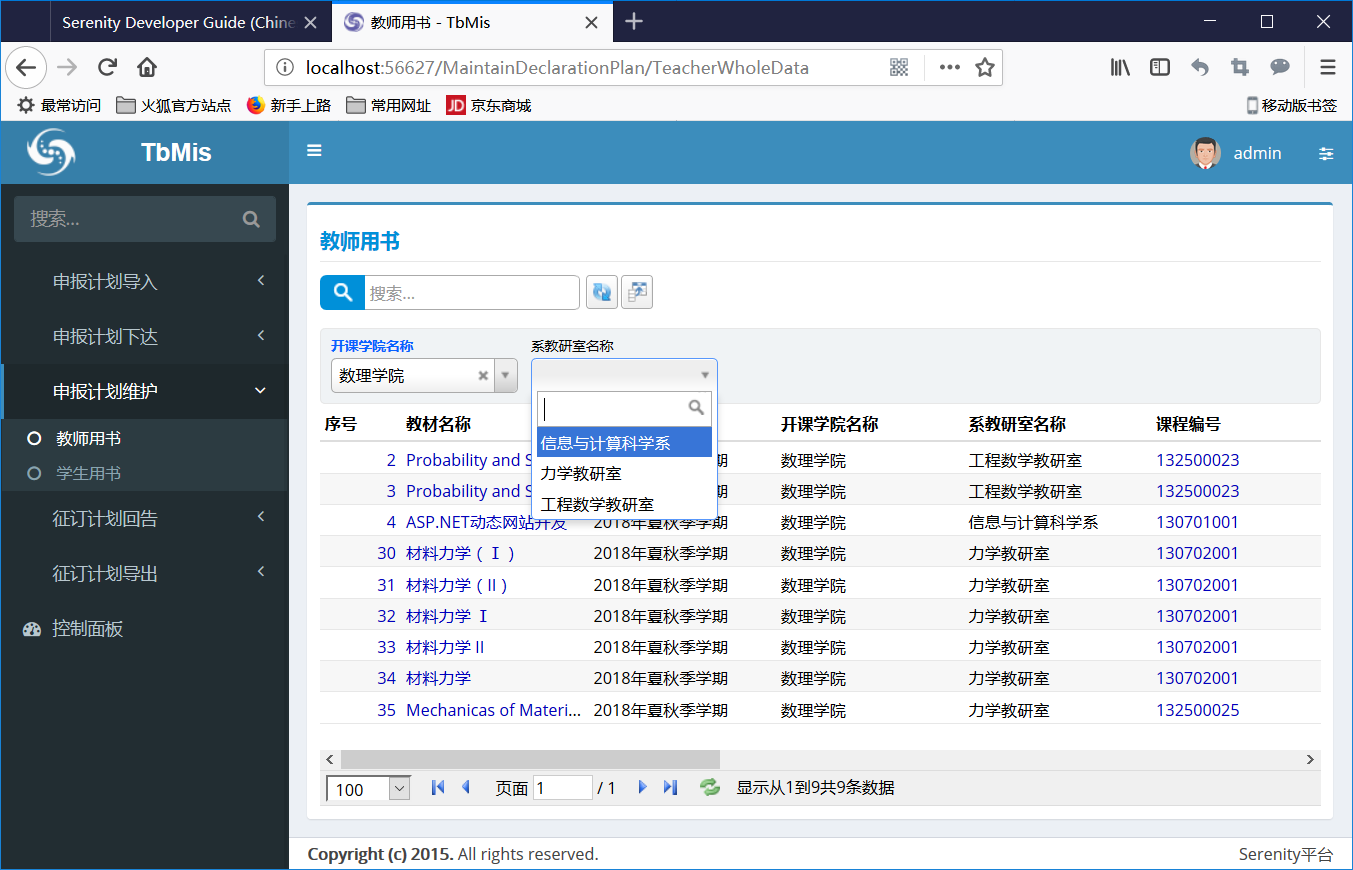
//在SchoolName这条语句上写上[LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherDepartmentNameLookup))]

[QuickFilter(CssClass = "hidden-xs"), QuickFilterOption("cascadeFrom", "SchoolName")]

语句就可以实现二级的筛选了。

学生的申报计划维护表也是采用同样的方法进行实现。修改好后我们可以运行一下我们的程序可以看到我们筛选框。





**3.5 点击列中属性显示详细信息**

我们在MaintainDeclarationPlan目录下通过程序包控制台中的sergen生成我们需要表的代码，我们这里需要Course和Textbook两个表。我们命名为CourseMessage和TextbookMessage

生成这两个表的代码后，我们在MaintainDeclarationPlanNavigation文件中把对应的代码删掉，来控制他们不会在导航中显示出来。只保留这两行，

[assembly: NavigationLink(3, "申报计划维护/教师用书", typeof(MyPages.TeacherWholeDataController), icon: null)]

[assembly: NavigationLink(3, "申报计划维护/学生用书", typeof(MyPages.StudentWholeDataController), icon: null)]

在CourseMessageRow和TextbookMessageRow中将属性的DisplayName修改好

[DisplayName("课程ID"), Column("CourseID"), Size(50), NotNull, QuickSearch]

public String CourseId

{

get { return Fields.CourseId[this]; }

set { Fields.CourseId[this] = value; }

}

[DisplayName("教材ID"), Column("TextbookID"), Size(50), NotNull, QuickSearch]

public String TextbookId

{

get { return Fields.TextbookId[this]; }

set { Fields.TextbookId[this] = value; }

}

做好上面的操作后，我们在TeacherWholeDataColumns中做修改，在需要点击的属性上加上一行代码。

[Width(150), EditLink(ItemType = "MaintainDeclarationPlan.TextbookMessage")]

public String TextbookNum2TextbookName { get; set; }

[Width(150), EditLink(ItemType = "MaintainDeclarationPlan.CourseMessage")]

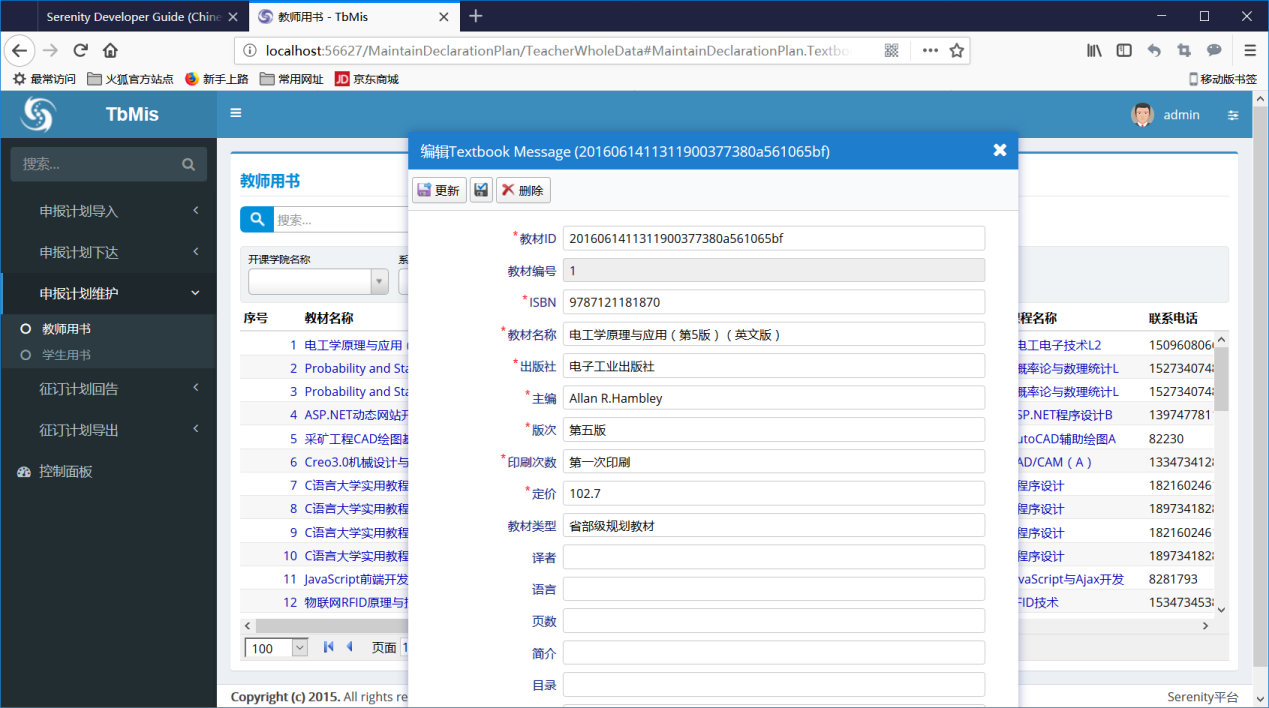
public String CourseNumCourseCode { get; set; }

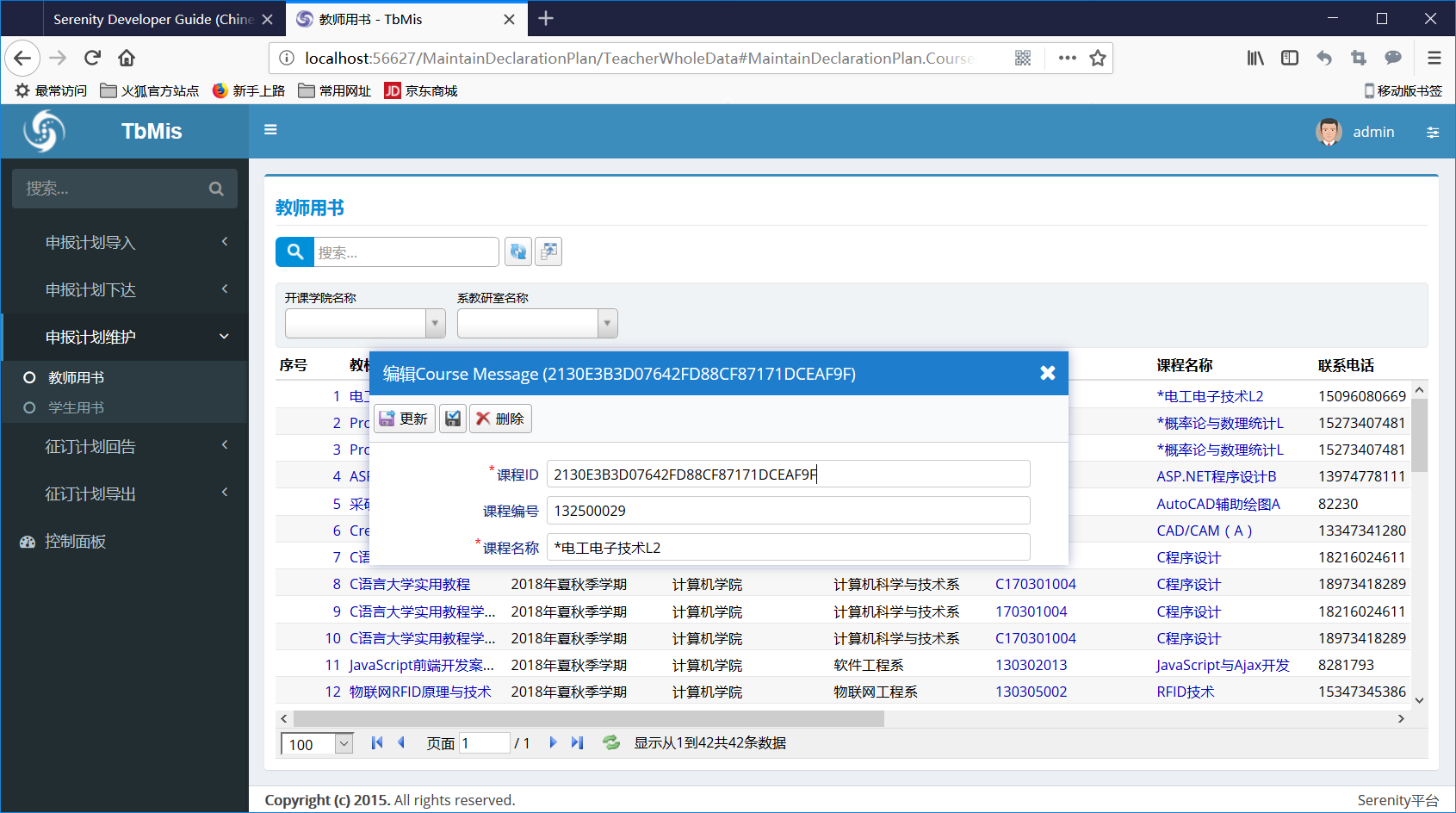
[Width(150), EditLink(ItemType = "MaintainDeclarationPlan.CourseMessage")]

public String CourseNumCourseName { get; set; }

我们这里有三个属性可以点击，所以加了三行代码。

学生的申报计划维护也是这样进行操作，我们运行一下可以看到我们代码的效果。





**3.6 修改信息时的二级关联**

我们同样需要用到之前写的两个Lookup类, TeacherSchoolNameLookup和TeacherDepartmentNameLookup,我们在TeacherWholeDataRow中进行添加代码。

[DisplayName("开课学院名称"), Size(50), NotNull, LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherSchoolNameLookup), AutoComplete = true)]

public String SchoolName

{

get { return Fields.SchoolName[this]; }

set { Fields.SchoolName[this] = value; }

}

[DisplayName("系教研室名称"), Size(50), NotNull, LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherDepartmentNameLookup), CascadeFrom = "SchoolName", AutoComplete = true)]

public String DepartmentName

{

get { return Fields.DepartmentName[this]; }

set { Fields.DepartmentName[this] = value; }

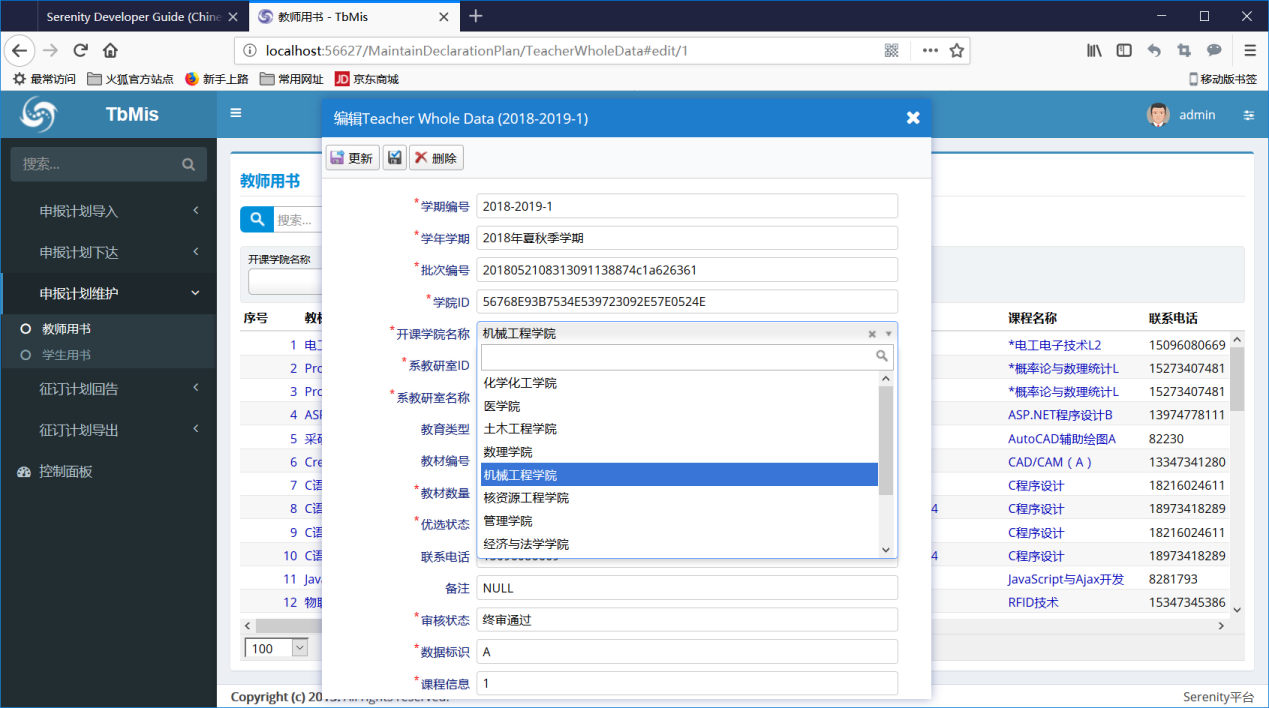
}

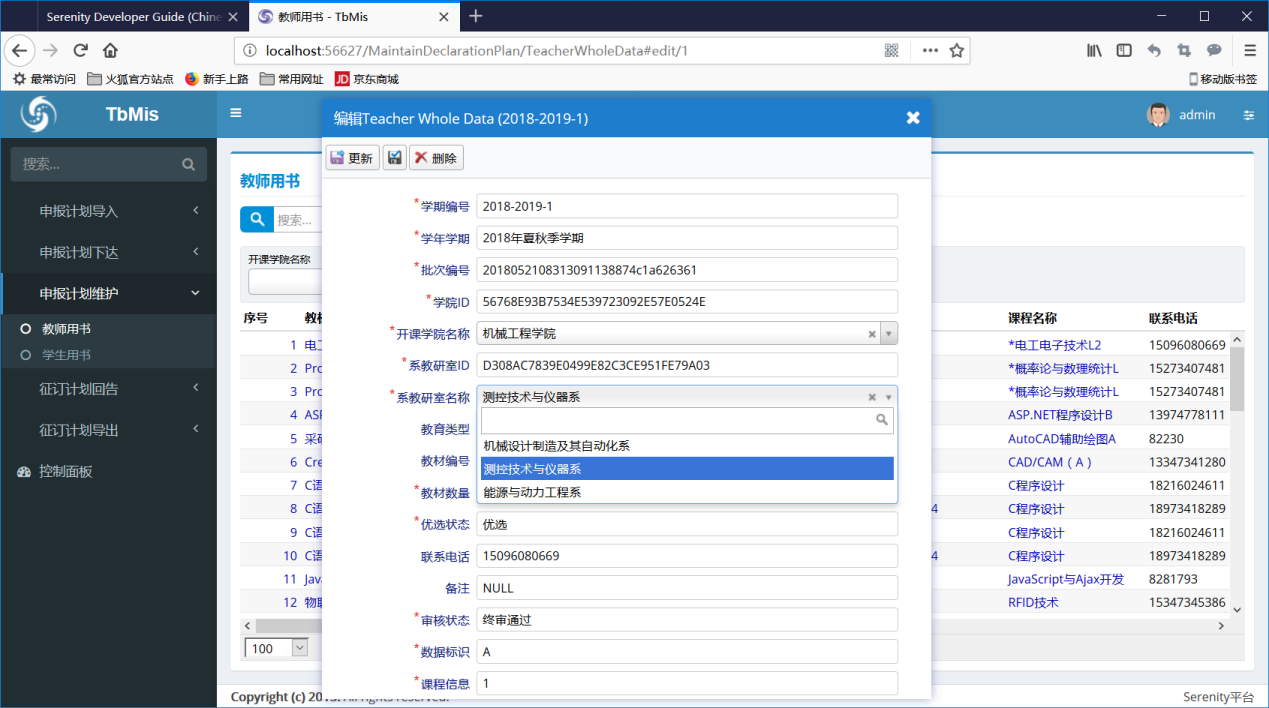
在需要关联的属性上加上一行代码

[DisplayName("开课学院名称"), Size(50), NotNull, LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherSchoolNameLookup), AutoComplete = true)]

[DisplayName("系教研室名称"), Size(50), NotNull, LookupEditor(typeof(Modules.MaintainDeclarationPlan.Lookups.TeacherDepartmentNameLookup), CascadeFrom = "SchoolName", AutoComplete = true)]

学生的申报计划维护表也是同样的操作，效果如下





**3.7 删除新增按钮**

我们这个维护的部分不需要新增计划，所以我们要把新增的按钮删掉。

我们在MaintainDeclarationPlan/TeacherWholeData目录下找到TeacherWholeDataGrid文件添加上一段代码。

namespace TbMis.MaintainDeclarationPlan {

@Serenity.Decorators.registerClass()

export class TeacherWholeDataGrid extends Serenity.EntityGrid<TeacherWholeDataRow, any> {

protected getColumnsKey() { return 'MaintainDeclarationPlan.TeacherWholeData'; }

protected getDialogType() { return TeacherWholeDataDialog; }

protected getIdProperty() { return TeacherWholeDataRow.idProperty; }

protected getLocalTextPrefix() { return TeacherWholeDataRow.localTextPrefix; }

protected getService() { return TeacherWholeDataService.baseUrl; }

constructor(container: JQuery) {

super(container);

}

//add

protected getButtons() {

var buttons = super.getButtons(); buttons.splice(Q.indexOf(buttons, x => x.cssClass == "add-button"), 1);

return buttons;

}

}

}

我们添加上的代码是

protected getButtons() {

var buttons = super.getButtons(); buttons.splice(Q.indexOf(buttons, x => x.cssClass == "add-button"), 1);

return buttons;

}

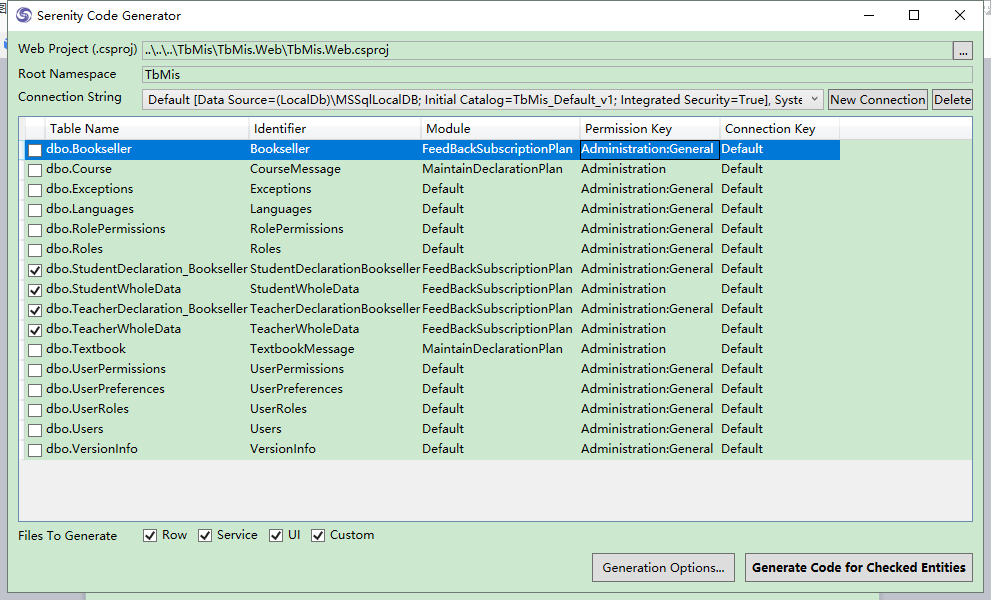
学生的申报计划维护也一样采用同样的方法。最终我们的页面效果如下:



**四、征订计划回告**

**4.1 为相关表生成代码**

再次使用程序包管理器控制台打开 sergen.exe，并为 Genre 表设置生成代码的参数，如下所示：



**4.2 处理导航界面**

重新生成解决方案后，我们将该文件的内容

Modules/FeedbackSubscriptionPlan/FeedbackSubscriptionPlanNavigation.cs修改为：

*若文件夹中无此文件，请选择*FeedbackSubscriptionPlan*文件夹右击添加->类，添加*FeedbackSubscriptionPlan*Navigation.cs文件*

//Add these

using Serenity.Navigation;

using MyPages = TbMis.FeedbackSubscriptionPlan.Pages;

[assembly: NavigationLink(4, "征订计划回告/教师用书", typeof(MyPages.TeacherDeclarationBooksellerController), icon: null)]

[assembly: NavigationLink(4, "征订计划回告/学生用书", typeof(MyPages.StudentDeclarationBooksellerController), icon: null)]

**4.3 设置界面标题**

为了设置界面的标题，我们在

Modules/FeedbackSubscriptionPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerRow.cs

以及

Modules/FeedbackSubscriptionPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerRow.cs

文件中修改以下内容：

// This...

[DisplayName("学生用书"), InstanceName("学生用书")]

//This...

[DisplayName("教师用书"), InstanceName("教师用书")]

**4.4 设置属性显示格式**

为了设置属性以中文的格式显示，我们在

Modules/FeedbackSubscriptionPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerRow.cs

以及

Modules/FeedbackSubscriptionPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerRow.cs

文件中将相应的属性修改为以下形式：

//This...

[DisplayName("征订状态")]

[Size(1), NotNull]

public String SubscriptionStatus

{

get { return Fields.SubscriptionStatus[this]; }

set { Fields.SubscriptionStatus[this] = value; }

}

**4.5 设置表格显示内容**

我们根据在表格所需要显示的内容，在

Modules/FeedbackSubscriptionPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerForm.cs

以及

Modules/FeedbackSubscriptionPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerForm.cs

文件中修改以下内容：

//Add these

public String SchoolName { get; set; }

public Int32 DeclarationId { get; set; }

public String BooksellerName { get; set; }

public String BooksellerId { get; set; }

public String SubscriptionStatus { get; set; }

public DateTime FeedbackDate { get; set; }

public String FeedbackStaff { get; set; }

public String Remark { get; set; }

public String ReDeclareStatus { get; set; }

**4.6 设置编辑框显示内容**

我们根据在表格所需要显示的内容，在

Modules/FeedbackSubscriptionPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerColumns.cs

以及

Modules/FeedbackSubscriptionPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerColumns.cs

文件中修改以下内容：

//Add These

[EditLink]

public String SchoolName { get; set; }

public Int32 DeclarationId { get; set; }

public String BooksellerName { get; set; }

public String BooksellerId { get; set; }

public String SubscriptionStatus { get; set; }

public DateTime FeedbackDate { get; set; }

public String FeedbackStaff { get; set; }

public String Remark { get; set; }

public String ReDeclareStatus { get; set; }

**4.7 设置相关属性只读**

由于在征订计划回告部分，不可修改申报编号，学院名称，书商编号，书商名称，所以我们将该部分的属性设置为只读状态，在

Modules/FeedbackSubscriptionPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerRow.cs

以及

Modules/FeedbackSubscriptionPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerRow.cs中，修改以下内容：

//Add this

[ReadOnly(true)]

public Int32? DeclarationId

{

get { return Fields.DeclarationId[this]; }

set { Fields.DeclarationId[this] = value; }

}

其中其他属性均按照该内容编辑，可类推。

**4.8 删除新增按钮**

由于征订计划回告部分是不需要新增功能的，所以我们将模板中的新增按钮删去，具体实现可修改

Modules/FeedbackSubscriptionPlan/StudentDeclarationBookseller/StudentDeclarationBookseller/StudentDeclarationBooksellerGrid.cs

以及

Modules/FeedbackSubscriptionPlan/TeacherDeclarationBookseller/TeacherDeclarationBookseller/StudentDeclarationBooksellerGrid.cs中的内容**：**

//Add these

protected getButtons() {

var buttons = super.getButtons();

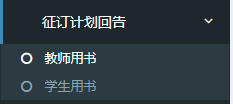
buttons.splice(Q.indexOf(buttons, x => x.cssClass == "add-button"), 1);

return buttons;

}

**4.9 最终内容显示**

在完成以上步骤之后，我们就得到了该模块的所有内容，界面实现如下：







1. **教材征订计划导出**

**5.1 创建一个名为TbMis的新项目**

在Visual Studio中选择文件->新建项目。选择Serene模板，在名称中输入TbMis并单击确定按钮。

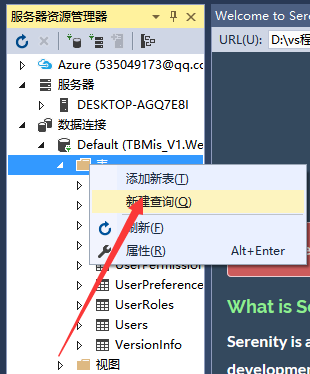
在解决方案资源管理器中，你应该看到一个名称为TbMis.Web的项目。

TbMis.Web是一个ASP.NETMVC应用程序，包含服务器端代码以及静态资源，如CSS文件，图片等。

**5.2 创建教材征订计划导出表**

要存储教材征订计划，我们需要一张教材征订计划表。我们使用SQL脚本创建教材征订计划导出表。当然，也可以使用Fluent Migrator创建迁移类（migration）的方式创建表。

教材征订计划导出表SQL脚本下载：https://github.com/idhua/TBMisSampleDate打开服务器资源管理器，如图，点击新建查询，将下载的SQL脚本粘贴到新建查询，然后执行，可以看见项目数据库中已建好了表。



**5.3 为教材征订计划导出表生成代码**

确保表存到数据库，我们将使用Serenity代码生成器（sergen.exe）生成初始的可 编辑界面。在Visual Studio中，通过点击 视图 => 其它窗体 => 包管理器控制台 打开包管理器控制台。输入sergen并按Enter键确认。



Connection String :

点击下拉框选择Default，选择后就会出现该数据库中表的信息

Identifier:

可以修改生成后的表文件的名字

Module:

在 Serenity 术语中，一个模块（module）是一组逻辑页面（pages），共享同一个目标。为教师计划申报计划表（TeacherDeclaration\_Bookseller）生成代码例如，在 Serene 模板中，所有与ExportSubscriptionPlan相关的页面都属于ExportSubscriptionPlan模块。

在设置如图所示的参数（你只需要设置Module Name其他使用默认值）之后，单击Generate Codefor Entity按钮。Sergen将生成几个文件并将它们包含进 TbMis.Web和TbMis.Scrip

t项目。现在你可以关闭Sergen，并返回到Visual Studio。修改项目，Visual Studio将询问你是否要重新加载更改，请单击加载所有。重新生成解决方案，然后按F5启动应用程序。使用admin作为用户名，并以serenity作为密码登录系统。当你看到控制面板页面时，你会发现左侧导航的底部有一个新的菜单ExportSubscriptionPlan。单击以展开它，然后单击TeacherDeclaration\_Bookseller，打开使用Sergen生成的第一个页面。

生成代码后，我们可以在TbMis.Web/Modules/ExportSubscriptionPlan/TeacherDeclarat

ion\_Bookseller查看生成的代码

**5.4 自定义教师用书征订计划导出表界面**

修改显示在导航目录下显示的名字

我们在ExportSubscriptionPlan文件中将导航信息修改为“征订计划导出/教师用书”

选择目录TbMis.Web/Modules/ExportSubscriptionPlan/TeacherDeclaration\_Bookseller

/ExportSubscriptionPlanNavigation.cs。你将看到：

using Serenity.Navigation;

using MyPages = TbMis.ExportSubscriptionPlan.Pages;

[assembly: NavigationLink(int.MaxValue, "ExportSubscriptionPlan/Student Declaration Bookseller", typeof(MyPages.StudentDeclarationBooksellerController), icon: null)]

[assembly: NavigationLink(int.MaxValue, "ExportSubscriptionPlan/Teacher Declaration Bookseller", typeof(MyPages.TeacherDeclarationBooksellerController), icon: null)]

将其改为：

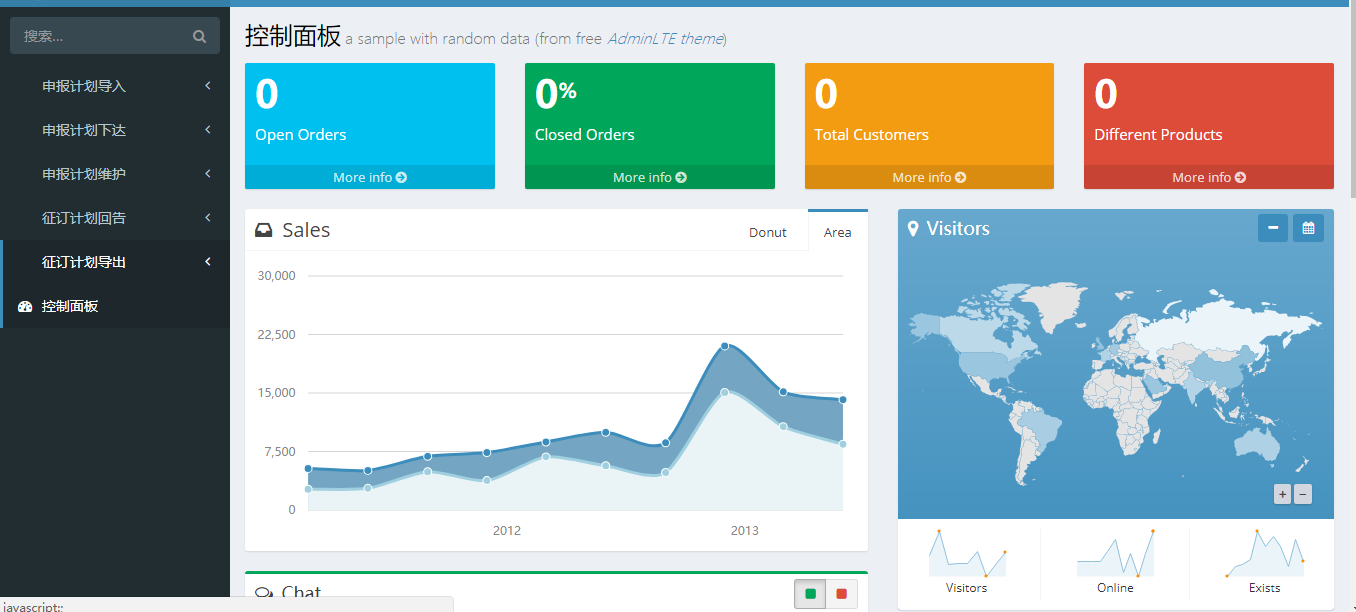
using Serenity.Navigation;

using MyPages = TbMis.ExportSubscriptionPlan.Pages;

[assembly: NavigationLink(5, "征订计划导出/教师用书", typeof(MyPages.TeacherDeclarationBooksellerController), icon: null)]

[assembly: NavigationLink(5, "征订计划导出/学生用书", typeof(MyPages.StudentDeclarationBooksellerController), icon: null)]

重新运行，将看到导航界面信息已修改。



**5.5 修改表页面显示内容**

我们找到TeacherDeclarationBooksellerRow.cs将DisplayName中的名字修改为教师用书

namespace TbMis.ExportSubscriptionPlan.Entities

{

//...

[ConnectionKey("Default"), Module("ExportSubscriptionPlan"), TableName("[dbo].[TeacherDeclaration\_Bookseller]")]

[DisplayName("教师用书"), InstanceName("教师用书")]

[ReadPermission("Administration:General")]

[ModifyPermission("Administration:General")]

//...

}

**5.6 修改列显示内容**

我们找到TeacherDeclarationBooksellerRow.cs将属性上的DisplayName改成自己希望显示的名字，这里以DeclarationId申报ID为例

//...

[DisplayName("申报编号"), Column("DeclarationID"), PrimaryKey, ForeignKey("[dbo].[TeacherWholeData]", "DeclarationID"), LeftJoin("jDeclaration"), TextualField("DeclarationTermCode")]

public Int32? DeclarationId

{

get { return Fields.DeclarationId[this]; }

set { Fields.DeclarationId[this] = value; }

}

//...

**5.7 在列中显示属性**

这个功能我们在TeacherDeclarationBooksellerColumns.cs中实现，我们将要展示的属性按顺序写好，按格式写上代码就可以在列中显示出来

namespace TbMis.ExportSubscriptionPlan.Columns

{

using Serenity;

using Serenity.ComponentModel;

using Serenity.Data;

using System;

using System.ComponentModel;

using System.Collections.Generic;

using System.IO;

[ColumnsScript("ExportSubscriptionPlan.TeacherDeclarationBookseller")]

[BasedOnRow(typeof(Entities.TeacherDeclarationBooksellerRow), CheckNames = true)]

public class TeacherDeclarationBooksellerColumns

{

[EditLink, DisplayName("Db.Shared.RecordId"), AlignRight]

public Int32 DeclarationId { set; get; }

public String DeclarationBatchId { set; get; }

public String DeclarationTermName { set; get; }

public String DeclarationSchoolName { set; get; }

public String DeclarationDepartmentName { set; get; }

public String CourseCode { get; set; }

public String CourseName{ get; set; }

public String DeclarationPhone { set; get; }

public String DeclarationCheckState { set; get; }

public String DeclarationPriority { set; get; }

public String DeclarationDataSign { set; get; }

public Int32 DeclarationApprovedAmount { set; get; }

public String DeclarationRemarks { set; get; }

public String BooksellerId { set; get; }

public String BooksellerBooksellerName { set; get; }

public String SubscriptionStatus { set; get; }

public DateTime FeedbackDate { set; get; }

public String FeedbackStaff { set; get; }

public String Remark { set; get; }

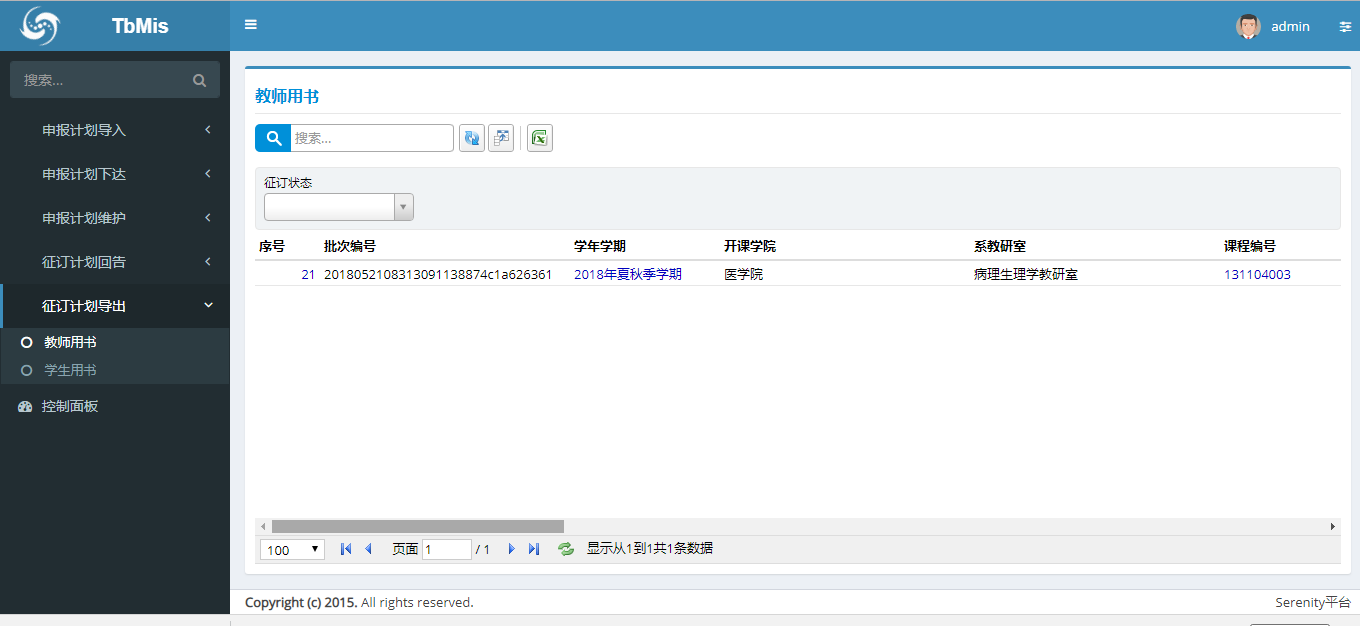
public String ReDeclareStatus { set; get; }

public String DeclarationEducationalType { set; get; }

}

}

再次生成并运行项目，我们将看到显示的表名和列名发生了变化：



**5.8 按征订状态筛选表并导出**

我们ExportSubscriptionPlan/TeacherDeclarationBookseller目录下创建一个新的类项，名为TeacherCheckState.cs，来实现我们筛选。

代码如下：

namespace TbMis.Modules.ExportSubscriptionPlan.Lookups

{

using Serenity.ComponentModel;

using Serenity.Data;

using Serenity.Web;

using TbMis.ExportSubscriptionPlan.Entities;

[LookupScript]

public class TeacherSubscriptionStatusLookup : RowLookupScript<TeacherDeclarationBooksellerRow>

{

public TeacherSubscriptionStatusLookup()

{

IdField = TextField = TeacherDeclarationBooksellerRow.Fields.SubscriptionStatus.PropertyName;

}

protected override void PrepareQuery(SqlQuery query)

{

var fld = TeacherDeclarationBooksellerRow.Fields;

query.Distinct(true)

.Select(fld.SubscriptionStatus)

.Where(

new Criteria(fld.SubscriptionStatus) != "" &

new Criteria(fld.SubscriptionStatus).IsNotNull()

);

}

protected override void ApplyOrder(SqlQuery query)

{

}

}

}

并为TeacherDeclarationBooksellerColumns.cs的SubscriptionStatus属性添加LookupEditor特性：

//...

[LookupEditor(typeof(Modules.ExportSubscriptionPlan.Lookups.TeacherSubscriptionStatusLookup)), QuickFilter(CssClass = "hidden-xs")]

public String SubscriptionStatus { set; get; }

//...

在上面我们写了一段实现类似Sql语句的类，并把这个类作为LookupEditor的一个参数，加入SubscriptionStatus的特性。

**5.9 设置点击不同的列可查看不同的表属性：**

在TeacherDeclarationBooksellerColumns.cs下为DeclarationTermName和CourseName添加EditLink特性：

//...

[Width(150), EditLink(ItemType = "TextBookInfo.Textbook")]

public String DeclarationTermName { set; get; }

//...

[Width(150), EditLink(ItemType = "CourseInfo.Course")]

public String CourseCode { get; set; }

[Width(150), EditLink(ItemType = "CourseInfo.Course")]

public String CourseName{ get; set; }

//...

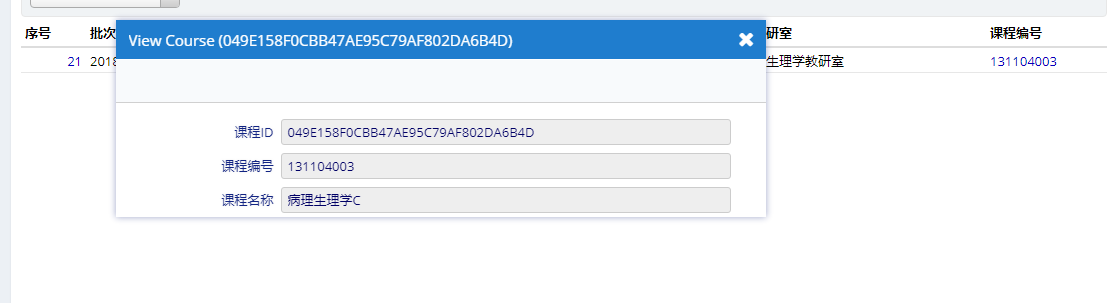
在这里，Width（150）代表设置列宽为150px，而EditLink(ItemType = "TextBookInfo.Textbook")表示点击该列会显示CourseInfo.Course目录下的表属性，而这里的CourseInfo.Course在申报计划导入时就已经写好了的。

现在我们可以运行一下，看看效果：

点击学期学年：



点击课程编号：



**5.10 增加将表导出为Excel功能：**

在TeacherDeclarationBooksellerGrid.ts下增加导出按钮：

//...

constructor(container: JQuery) {

super(container);

}

protected getButtons() {

var buttons = super.getButtons();

buttons.push(Common.ExcelExportHelper.createToolButton({

grid: this,

service: TeacherDeclarationBooksellerService.baseUrl + '/ListExcel',

onViewSubmit: () => this.onViewSubmit(),

separator: true

}));

return buttons;

}

//...

在TeacherDeclarationBooksellerEndpoint.cs增加导出功能：

//...

public FileContentResult ListExcel(IDbConnection connection, ListRequest request)

{

var data = List(connection, request).Entities;

var report = new DynamicDataReport(data, request.IncludeColumns, typeof(Columns.TeacherDeclarationBooksellerColumns));

var bytes = new ReportRepository().Render(report);

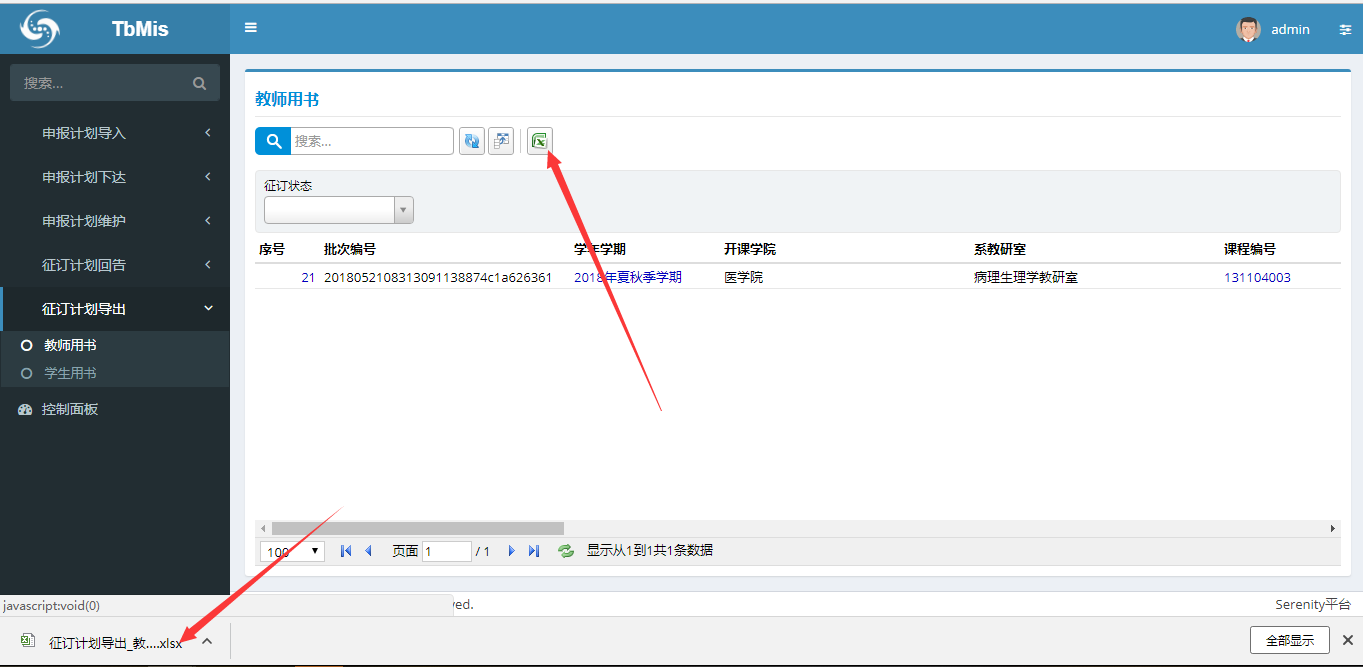
return ExcelContentResult.Create(bytes, "征订计划导出\_教师用书\_" +

DateTime.Now.ToString("yyyyMMdd\_HHmmss") + ".xlsx");

}

在这里，我们新增了按钮并实现了导出功能，在新增按钮时我们调用了Serenity的Common.ExcelExportHelper.createToolButton函数，设传入service为TeacherDeclarationBooksellerService.baseUrl + '/ListExcel'，在Endpoint.cs我们为了避免重名，将下载的表名字改为征订计划导出\_教师用书\_+当前日期（精确到秒）.xlsx。

现在，我们运行一下，试试导出功能：



**5.11 删除新增按钮并设置表属性不可改：**

在TeacherDeclarationBooksellerGrid.ts删除新增按钮：

//...

protected getButtons() {

//...

buttons.splice(Q.indexOf(buttons, x => x.cssClass == "add-button"), 1);

return buttons;

}

在TeacherDeclarationBooksellerDialog.ts设置表属性不可改：

//...

protected form = new TeacherDeclarationBooksellerForm(this.idPrefix);

protected getToolbarButtons(): Serenity.ToolButton[] {

let buttons = super.getToolbarButtons();

buttons.splice(Q.indexOf(buttons, x => x.cssClass == "save-and-close-button"), 1);

buttons.splice(Q.indexOf(buttons, x => x.cssClass == "apply-changes-button"), 1);

return buttons;

}

protected updateInterface(): void {

super.updateInterface();

Serenity.EditorUtils.setReadonly(this.element.find('.editor'), true);

this.element.find('sup').hide();

this.deleteButton.hide();

}

protected getEntityTitle(): string {

if (!this.isEditMode()) {

return "How did you manage to open this dialog in new record mode?";

}

else {

var entityType = super.getEntitySingular();

let name = this.getEntityNameFieldValue() || "";

return 'View ' + entityType + " (" + name + ")";

}

}

protected updateTitle(): void {

super.updateTitle();

}

//...

按F5运行，效果如下：

