САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Направление подготовки: математическое обеспечение и администрирование информационных систем

Сервис для сдачи и проверки домашних работ с помощью WebSharper: Client

Курсовая работа студента 242 группы
Кижнерова Павла Александровича

Научный руководитель:
доц. Литвинов Ю.В.

Санкт-Петербург
2018

Оглавление

Введение

Выполнение и сдача домашних работ – неотъемлемая часть современной системы обучения. Эффективность усваивания информации студентом напрямую зависит от удобства используемых средств для сдачи и проверки заданий: студент тем быстрее получит конструктивную критику по поводу корректности своей работы, чем комфортней преподавателю будет ее проверять.

Цель проекта – создание сервиса для сдачи и проверки домашних работ с использованием WebSharper.

WebSharper – веб-инструмент с открытым исходным кодом, предоставляемый IntelliFactory, для создания клиент-серверных приложений полностью на F# (с версии 4.0 и на C#).

Основная концепция WebSharper:
Конверсия F# (C#) в JS/CSS/HTML на этапе компиляции[1].

Основная концепция использования сервиса:
Преподаватели создают курс или выбирают шаблон курса из банка, указывая какие задания необходимо и достаточно выполнить для получения зачета по заданной дисциплине.
Задания могут быть либо взяты из банка, либо созданы новые.
Студенты выбирают курсы и записывается на них.
Преподаватель, которому принадлежит курс, либо отклоняет, либо принимает заявку на вступление.
Студенты размещают решения задач.
Преподаватели получает решения, и либо помечает его как корректное, либо запрашивает изменения.
Также ведется таблица с успехами всех студентов конкретного курса.

Общая структура приложений на WebSharper выглядит следующим образом:

- Main.html
- Client.fs
- Site.fs
- Remoting.fs

В Main.html хранится шаблон страницы, в html-элементах которой есть заглушки, в которые могут быть вставлены другие html-элементы, созданные в Client.fs

Html-элементы, созданные в Client.fs называются pagelet'ами

Site.fs замещает заглушки в Main.html pagelet'ами из Client.fs, в зависимости от ситуации. Таким образом выполняется маршрутизация по сайту.

Разработка проекта на WebSharper - весьма трудоемкая деятельность ввиду недостаточного количества документации: на русскоязычных ресурсах отсутствует полностью, на англоязычных - представлена в виде набора методов, часто без указания случаев применения. Тем не менее, данный проект интересен тем, что в перспективе может послужить как пример использования данного инструмента.

Главная цель данной работы – получить опыт в веб-программировании и изучить инструмент WebSharper.

Для ее достижения было принято решение начать разработку сервиса для сдачи и проверки домашних работ.

Постановка задачи

В данной курсовой работе были поставлены следующие задачи, связанные с разработкой клиентской части:

- Реализовать pagelet’ы для конструирования веб-страниц
- Реализовать модели часто используемых сущностей
- Реализовать отображение моделей в зависимости от контекста

Обзор существующих решений

HwProj

Сервис для сдачи и проверки домашних работ, который был запущен в 2014 году и функционирует до сих пор.
Концепция использования данного сервиса положила фундамент нашему проекту:

- Существует две роли: студент и преподаватель
  Преподаватель может:
  - Создавать курс
  - Принимать заявки студентов на вступление
  - Назначать задания на конкретный курс, необходимые для получения зачета по дисциплине курса
  - Принимать решения, присланные студентами

  Студенты могут:
  - Отправлять заявку на вступление в курс
  - Присылать свои решения, получают критику от преподавателя

- Для каждого курса ведется сводная таблица успехов всех его участников: пересечение строки с именем студента и столбца с заданием может быть закрашено четырьмя цветами
  - Зеленый – решение принято
  - Оранжевый – преподаватель запросил коррективы
  - Светло-зеленый – решение загружено впервые и ожидает проверки преподавателем
  - Светло-оранжевый – решение претерпело изменения и ожидает проверки преподавателем

- Реализована система «медалек»: студенты, вошедшие в тройку первых сдавших решения, получают золотую, серебряную и бронзовую медали соответственно порядку сдачи.
- Решение задачи можно загрузить двумя способами:
  - Разместить решение непосредственно на сервисе
  - Разместить ссылку на pull request с решением на удаленном репозитории GitHub
- Есть возможность начать чат с преподавателем в окне размещения решения

В ходе активного четырехлетнего тестирования данного сервиса было выявлено, что в проекте присутствуют некоторые изъяны, известные автору данной курсовой работы:

- Невозможность покинуть курс
- При размещении решения с помощью ссылки на pull request в удаленном репозитории GitHub появляется ошибка, несмотря на то, что ссылка была успешно загружена в сервис

HwProj был разработан с использованием языка программирования Ruby. Ввиду того, что данным языком на должном уровне владеют малое количество программистов, стоит проблема сопровождения актуальной версии сервиса.

Подводя итог, самая идея такого сервиса получила много положительных отзывов, и, как следствие, является актуальной. Однако существующая реализация имеет вышеописанные недостатки, в связи с чем и возникла тема данной курсовой работы.

Описание решения

Актуальная структура проекта несущественно отличается от той, что была приведена во введении.

Для удобства было принято решение создать модели и представления часто встречающихся сущностей и вынести их в отдельный файл. При этом, исходя из соображений разделения кода на логические сегменты, файл Client.fs был разделен на RegClient.fs и CommonClient.fs. Теперь структура проекта имеет вид:

- Main.html
- RegClient.fs
- CommonClient.fs
- ModelsClient.fs
- RendersClient.fs
- Site.fs
- Remoting.fs

RegClient.fs хранит pagelet'ы, отвечающие за форму входа и регистрации, CommonClient.fs – pagelet'ы, отвечающие за вывод списков:

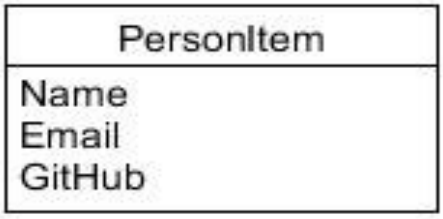
- Актуальные курсы
- Студенты конкретного курса
- Задания для выполнения конкретным студентом
- Решения для проверки конкретным преподавателем.

Реализация

Прежде всего, для предупреждения многочисленного повторения кода, было принято решение представить часто используемые сущности в виде объектов

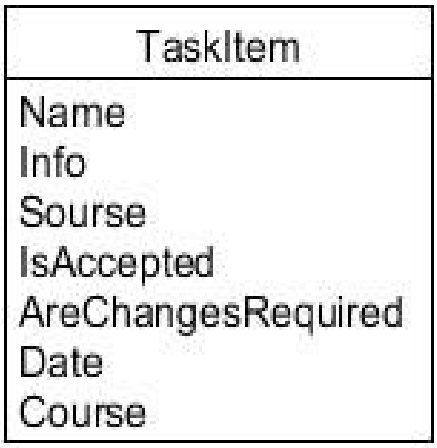
- PersonItem
- TaskItem
- RealTaskItem
- CourseItem

PersonItem



Представляет преподавателя или студента, а точнее, информацию о них: имя, эл. почта, ссылка на аккаунт GitHub

TaskItem

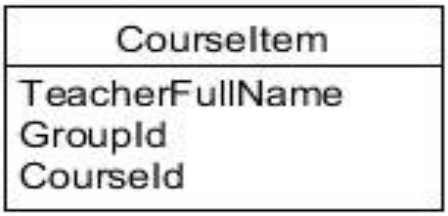


Представляет задание для выполнения студентом, или решение для проверки преподавателем.

Хранит информацию:

- Название задания
- Данные о сдающем задание студенте или принимающем решение преподавателе
- Ссылка на ресурс, где расположено условие или решение
- Зачтено ли решение
- Требует ли решение изменений

CourseItem



Представляет курс. Хранит информацию о преподавателе, академической группе, названии курса.

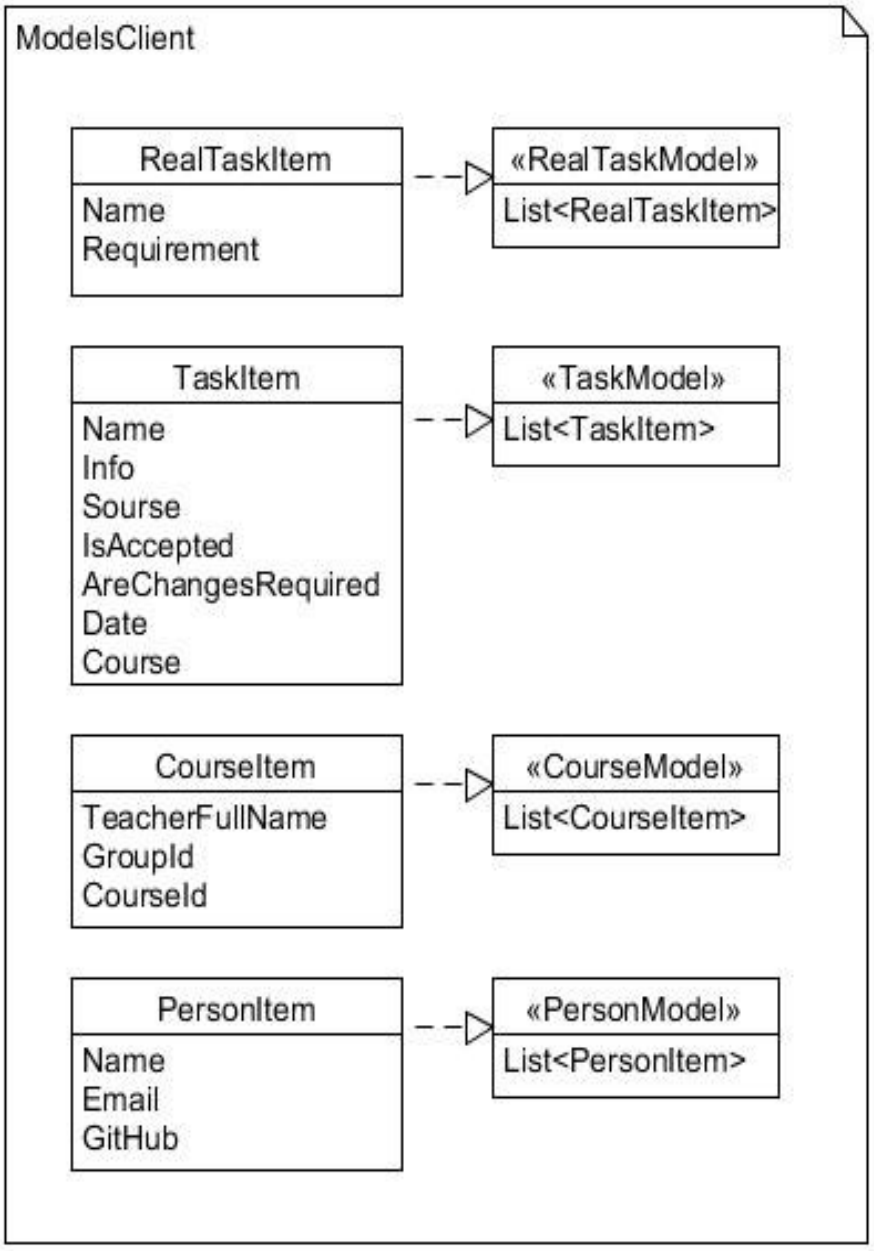
RealTaskItem

Представляет задание. Хранит информацию о названии задания и его условии.

Так как основной объем данных представляется в виде списков, были реализованы модели – объекты с полем-списком из объектов *Item

Каждому объекту *Item соответствует модель, содержащая список таких объетов

- PeopleModel
- TasksModel
- CoursesModel
- RealTasksModel



В зависимости от контекста может понадобиться разное представление одной и той же модели.

Список заданий для студента должен быть представлен в виде списка из TaskItem, где напротив каждого задания есть кнопка, позволяющая перейти на страницу загрузки решения.

Список решений для преподавателя – в виде списка из TaskItem, где напротив каждого решения есть две кнопки: одна для принятия решения, другая – для запроса изменений

В обеих ситуациях используется TasksModel, однако отображением списков для каждой конкретной ситуацией занимаются разные методы из RendersClient.fs

Аналогичным образом обстоят дела с остальными моделями: может потребоваться разное представление списков

Основным типом в WebSharper для построения HTML является Doc, который может представлять собой один узел DOM или последовательность таких узлов (в том числе нулевая последовательность). Один и тот же Doc может состоять из разных элементов в разное время, в зависимости от действий пользователя. Таким образом, в CommonClient.fs с помощью данной технологии были описаны методы (pagelet'ы):

- CheckTasks() – решения на проверку для преподавателя
- DoTasks() – задания для выполнения стунентом
- CoursesOverview() – обзор всех существующих курсов
- AppointTask() – назначение задания из банка для конкретного курса
- CreateTask() – создание нового задания

Результаты

- Реализованы pagelet’ы для конструирования веб-страниц
- Реализованы модели часто используемых сущностей
- Реализовано отображение моделей в зависимости от контекста

Список использованной литературы

[1] WebSharper Documentation – WebSharper 4.x for F# //–

https://developers.websharper.com/docs/v4.x/fs (дата обращения 5.06.2018)

[2] WebSharper Documentation – WebSharper 3.x for F# //–

https://developers.websharper.com/docs/v3.x/fs (дата обращения 5.06.2018)

[3] Don Syme – Expert F# 4.0 Chapter 14 //–

https://books.google.ru/books?id=XKhPCwAAQBAJ&pg=PA392&lpg=PA392&dq=websharper+sitelet&source=bl&ots=GIRgleBObX&sig=PV5eLhrmdtUcN9xU_g8y28WaPlA&hl=ru&sa=X&ved=0ahUKEwjLwI6_ytvaAhUmxqYKHQ66C9E4ChDoAQg2MAI#v=onepage&q&f=true (дата обращения 5.06.2018)

[4] HwProj – сайт для проверки домашних работ //–

http://hwproj.me (дата обращения 5.06.2018)