Система онлайн-сопровождения научно-образовательных процессов ДвинемНауку

В. В. Осокин, Р. Ф. Алимов

Рассматривается задача разработки современной системы онлайн-сопровождения научно-образовательных процессов ДвинемНауку (http://dvinemnauku.ru). Система позволяет сопровождать научно-образовательные процессы онлайн-отображениями этих процессов. Система разделена на две части: на серверную и на клиентскую. Серверная часть используется для хранения, обработки и выдачи данных. Клиентская часть используется для представления и сбора данных. В рамках работы реализована подсистема управления процессами, событиями и комментариями, спроектирована архитектура системы и структура базы данных, а также реализованы вспомогательные модули.

Ключевые слова: научно-образовательный процесс, проектирование баз данных, научное событие, ДвинемНауку, веб-приложение.

Введение

Рассматривается задача разработки современной системы онлайн-сопровождения научно-образовательных процессов ДвинемНауку (http://dvinemnauku.ru). Система должна позволять сопровождать научно-образовательные процессы онлайн-отображениями этих процессов. В качестве примеров процессов можно привести семинары, лекции, научную работу студентов и прочее. По аналогии с ситуацией в жизни, где научно-образовательные процессы суть последовательность событий в рамках этих процессов, в системе ДвинемНауку процесс — это последовательность событий. В рамках этой системы,

опять же, по аналогии с реальной жизнью, должна быть возможность создавать как открытые (с общим доступом), так и закрытые (с ограниченным доступом) процессы. Каждый процесс должен иметь руководителей, к закрытым процессам доступ должны иметь только руководители и участники этих процессов.

В рамках работы реализованы важные модули системы, такие как управление процессами и их событиями. Реализована подписка на события процессов: каждый пользователь видит список новых событий всех процессов, на которые он подписан, на своей стене. Каждое событие процесса можно комментировать, добавлять к нему теги и файлы.

Для удобства и скорости работы с системой реализовано создание событий и процессов на лету по нажатию пары клавиш. При этом значения соответствующих полей проставляются по умолчанию. Так, преподаватель может в два клика добавить событие «лекция от заданного числа» и разместить в нем описание лекции, материалы к лекции. По умолчанию процессы и события в них создаются неопубликованными. Для их публикации достаточно нажать на кнопку «Опубликовать».

Применение протокола RESTfull API позволяет создавать неограниченное число программ-клиентов системы для различных платформ, совершенно не затрагивая ее серверную часть. Это обеспечивается полным разделением кода клиентской и серверной частей.

Особое внимание уделено проектированию базы данных. Разработанная структура БД способствует дальнейшему развитию и масштабированию серверной части.

При создании системы применены наиболее современные технологии, фреймворки и библиотеки, такие как RESTfull API, Yii, MySQL 5, AngularJS, jQuery 2.0, Bootstrap, HTML5 Boilerplate, Modernizr.

Постановка задачи и полученные результаты

Требуется создать систему управления процессами и событиями в рамках системы онлайн-сопровождения научно-образовательных процессов ДвинемНауку, обеспечив пользователей возможностью на одной веб-странице управлять процессом, его событиями и коммента-

риями к ним. Необходимо полностью разделить серверную и клиентскую части кода для обеспечения кроссплатформенности системы. Серверная часть должна представлять собой RESTfull API сервис с форматом обмена данными в виде JSON. Также необходимо создать клиент для браузера в виде одностраничного приложения. Необходимо, чтобы клиент был максимально удобен и прост в использовании.

В рамках решения поставленной задачи получены следующие основные результаты.

Создана система управления процессами, событиями и комментариями. Процессы создаются одним нажатием кнопки, быстро редактируются, и снова одним нажатием кнопки публикуются, становясь видимыми для участников системы. Для редактирования заголовка достаточно кликнуть на него, отредактировать и кликнуть вне поля редактирования. Для редактирования даты начала и окончания процесса достаточно кликнуть на дату, и в открывшемся календаре снова кликнуть на необходимую дату. Для добавления руководителя или участника нужно кликнуть на копку «Добавить» рядом с соответствующим списком пользователей, в появившемся поле ввода текста начать набирать имя пользователя или его e-mail и выбрать его в списке подсказок.

По схожему принципу осуществляется создание и редактирование событий. Также события можно комментировать и прикреплять к ним файлы. Вначале подгружается только часть комментариев, но в дальнейшем одним нажатием кнопки можно подгрузить остальные комментарии и, при необходимости, нажатием на ту же кнопку снова их скрыть. Для прикрепления файлов достаточно перетащить их в соответствующую область на веб-станице (также их можно выбрать в диалоговом окне выбора файлов). Немаловажную роль в создании такого удобного UX (User Experience — англ. Пользовательский Опыт) сыграл используемый JavaScript фреймворк AngularJS.

Описна архитектура системы и проработана структура базы данных. Описана общая архитектура системы, ее основные компоненты и принятые технические решения для разделения серверной и клиентских частей.

Структура базы данных создана в соответствии с принятыми правилами проектирования реляционных баз данных. Несмотря на то, что таблиц в базе данных довольно много, структура ее не является сложной. В качестве СУБД использована MySQL 5.

Реализованы вспомогательные модули. Описаны вспомогательные модули, написанные для системы. Для загрузки файлов используется плагин для jQuery под названием jQuery File Upload. Он успешно используется в обеих частях системы — и в клиентской, и в серверной. Для внедрения плагина пришлось изменять серверную часть кода и писать соответствующие директивы для AngularJS. В системе применяется календарь Pickadate. Для него написана директива AngularJS. Его применение потребовало изменения его кода. Реализовано редактирование полей процессов и событий на лету, добавление руководителей, участников, тегов, поиск.

На серверной стороне фреймворк Yii существенно дополнен для создания RESTfullAPI. Среди дополнений можно отметить такие, как сильно дополненный стандартный контроллер, конвертирование моделей, массивов и других типов данных в формат JSON, выставление правильных HTTP статусов во время ответа с сервера, расширенный класс моделей и многое другое.

Система управления процессами, событиями и комментариями

Рассмотрим работу с процессами, событиями и комментариями на примере реального использования системы. Допустим, нужно написать доклад по предмету «Естествознание» на тему «Роль естествознания в творчестве Леонардо да Винчи» и выступить с ним на семинаре. Всего на подготовку доклада отведено две недели. В данном случае научно-образовательный процесс — это подготовка доклада длительностью в две недели.

Создание процесса

Для создания нового процесса достаточно одного клика на кнопку «Создать процесс».

Далее работаем с процессом. Необходимо поменять его название, дату начала и дату окончания, настройки доступа. Для редактирования названия кликаем на заголовок процесса, после чего он становится редактируемым. Вводим необходимое название «Доклад по Естествознанию на тему «Роль естествознания в творчестве Леонар-

до да Винчи», кликаем за пределами поля редактирования и данные сохраняются.

Далее указываем дату начала и окончания процесса. Кликаем на дату начала, после чего появляется календарь. Кликаем в календаре необходимую дату и данные сохраняются. Теперь у процесса указаны его даты начала и окончания.

Приступаем к настройкам видимости процесса. По умолчанию процессы создаются закрытыми. Пока оставим процесс закрытым.

Можно добавить преподавателя по Естествознанию к руководителям процесса. Для этого в списке руководителей нажимаем кнопку «Добавить», после чего появляется текстовое поле, в котором начинаем вводить имя преподавателя или его email. Выбираем в подсказках нужного пользователя, после чего сохраняются данные, и преподаватель оказывается среди руководителей процесса. Теперь он имеет те же права на процесс, что и текущий пользователь системы, и может добавлять события, загружать файлы, комментировать события и так далее.

Преподаватель может подписаться на созданный процесс.

Добавление участников закрытого процесса ничем не отличается от добавления руководителей. Однако участники процесса имеют меньшие права в процессе по сравнению с руководителями — они могут только подписаться на него и комментировать события.

Создание события

На данный момент в процессе нет ни одного события. Добавим первое событие. В системе первое событие представляет собой описание процесса. Для добавления события кликаем на кнопку «Новое событие», после чего сразу создается событие с некоторыми полями по умолчанию.

Как обычно, приступаем к редактированию полей события. Название события редактируется так же, как и название процесса.

Далее редактируем описание события. Так как событие первое, то его описание также становится описанием всего процесса. Редактирование описания события ничем не отличается от редактирования его заголовка за исключением, что система после завершения задает вопрос «Сохранить изменения?». Соглашаемся, и данные сохраняются.

Редактирование даты события ничем не отличается от редактирования даты начала или окончания процесса.

Добавляем теги для краткого описания события и процесса.

Созданное первое событие еще не опубликовано и его никто не увидит кроме нас. Для публикации кликаем на кнопку «Опубликовать» и событие становится опубликованным.

Теперь у нас есть процесс для сопровождения написания доклада. Однако он пока не опубликован. Для публикации, как и в случае с событием, кликаем на кнопку «Опубликовать».

Через некоторое время готова первая версия нашего доклада. Создаем по этому случаю событие. Редактируем его так же, как редактировали первое событие. Однако на этот раз нужно к событию добавить файлы: первая версия доклада и все дополнительные файлы. Перетаскиваем файлы в определенную область события и файлы загружаются. Теперь эти файлы может скачать любой пользователь, у которого есть доступ к данному процессу.

Комментирование событий

Так как преподаватель подписан на процесс, он может увидеть событие у себя на стене, скачать файлы и прокомментировать работу прямо со стены. И так до тех пор, пока доклад не будет готов, и студент не ответит на семинаре.

Архитектура системы и структура базы данных

Система разделена на две части: на серверную и на клиентскую. Серверная часть используется для хранения, обработки и выдачи данных. Клиентская часть используется для представления и сбора данных.

Связь клиентской и серверной частей

Серверная и клиентская части общаются посредством протокола RESTfull API, являющимся надстройкой над протоколом HTTP. Это является неоспоримым преимуществом, так как HTTP самый распространенный и проверенный протокол на сегодняшний день. По

своей сути, протокол HTTP предназначен для передачи гипертекста (Hypertext transfer protocol). Надстройка RESTfull API позволяет использовать его также для передачи данных различного формата. В случае системы ДвинемНауку данные передаются в формате JSON (JavaScript object notation), который является простым и выразительным, и вместе с этим очень гибким.

Серверная часть

Как говорилось выше, серверная часть используется для хранения, обработки и выдачи данных в формате JSON посредством протокола RESTfull API. Инфраструктура веб-сервера — очень распространенная LAMP (Linux, Apache, MySQL, PHP). Помимо непосредственной разработки системы, также было необходимо настроить сервер (работает под управлением ОС Ubuntu для серверов), настроить систему управления версиями (используется Git), настроить и установить Apache (веб-сервер), MySQL (СУБД) и PHP (скриптовый язык программирования).

Для разработки системы было решено использовать фреймворк Yii из-за сочетания в нем простоты, функциональности и очень хорошо реализованной расширяемости. Важнейшим преимуществом Yii является то, что он является MVC фреймворком. MVC (Model, View, Controller) — это методология проектирования, когда система разбивается на три части: модель для работы с данными, представление для вывода данных и контроллер для ответа на запросы пользователей системы и связывания моделей с представлениями.

Связывание пути запроса с его обработчиком. Связывание путей запросов с их обработчиками в Yii задается в конфигурационных файлах фреймворка. В данном случае конфигурации прописаны в файле /proctected/config/main.php.

Рассмотрим подробнее строку конфигурации связывания пути вида /api/processes/<id процесса>/events с его обработчиком. При запросе по данному пути система выводит список событий процесса.

```
array('events/list',
 'pattern'=>'api/processes/<pid:\d+>/events',
 'verb'=>'GET'),
```

Здесь `'pattern'=>'api/processes/<pid:\d+>/events'` означает шаблон пути. Строка `<pid:\d+>` означает, что часть пути, располагающаяся на ее месте, должна удовлетворять регулярному выражению `\d+`, и данная часть будет передана обработчику как параметр под названием `pid` (process id). То есть этот параметр — уникальный идентификатор процесса системы.

Выражение `'verb'=>'GET'` означает, что данное связывание должно срабатывать только в случае, если метод HTTP запроса — GET. Помимо GET используются также методы POST для создания сущностей, PUT для редактирования сущностей и DELETE для удаления сущностей.

И наконец, `'events/list'` означает обработчик данного пути. Здесь `events` означает, что контроллер для данного пути — EventsController.php, а `list` означает, что метод данного котроллера — actionList.

Обработчики запросов. После того, как Yii находит нужное связывание по заданному пути запроса, запускается метод соответствующего контроллера. Для примера рассмотрим контроллер, который находится в файле /protected/controllers/EventsController.php. Контроллер представляет собой класс с названием `EventsController`, наследуемый от класса `ERestController` и состоящий из следующих методов: `actionList` — обработчик запросов на получение событий процессов; `actionUpdate` — обработчик запросов на обновление событий; `actionCreate` — обработчик запросов на создание новых событий; `actionDelete` — обработчик запросов на удаление событий.

Рассмотрим подробнее метод `actionList`. Он отвечает на запросы о получении событий процессов. В самом начале идет проверка доступа пользователя.

```
$this->checkUserFilled();
```

Далее, из базы данных загружаются данные процесса с уникальным идентификатором, переданном в параметре `pid`. Если процесс не найден в базе данных, контроллер выводит сообщение об ошибке. Если процесс найден в базе данных, но пользователь не имеет доступа на его чтение, то также выводится ошибка.

```
$process = Process::model()->findByAttributes(
array('id' => Yii::app()->request->getQuery('pid')));
if (is_null($process)) {
    $this->responseError(404, 'Process not found.');
}
if (!$process->userCanRead()) {
    $this->responseError(403, 'No access.');
}
```

Потом создаются критерии поиска событий в базе данных. Например, одним из критериев является то, что событие должно быть из указанного процесса.

```
$criteria = new CDbCriteria();
$criteria->compare('process_id', $process->id);
$criteria->order = 't.date DESC, t.id DESC';
$q = Yii::app()->request->getQuery('q', null);
if ($q) {
    $criteriaQ = new CDbCriteria();
    $criteriaQ->addSearchCondition('t.name', $q, true,
        'OR');
    $criteriaQ->addSearchCondition('t.description', $q,
        true, 'OR');
    $criteria->mergeWith($criteriaQ);
}
```

Следующим шагом является загрузка событий из базы данных, и их дополнительная обработка перед выводом.

```
$provider = new CActiveDataProvider($eventsModel, array(
    'criteria' => $criteria,
    'pagination' => array(
        'pageSize' => Yii::app()->params['rest']
            ['processes']['eventsPerPage'],
        'currentPage' =>
            Yii::app()->request->getQuery('page', 1)-1
    )
));
$events = $provider->getData();
```

```
if (empty($events)) {
    $events = array();
}
for ($i = 0; $i < count($events); $i += 1) {
    $events[$i]->prepareComments();
    $events[$i]->prepareTags(false);
    $events[$i]->prepareUploads(false);
}
```

И последний шаг в ответе на запрос на получение списка событий процесса — это вывод данных.

```
$this->outputHelper(array(
    'events' => $events,
    'count' => $provider->getTotalItemCount()
));
```

Вывод данных в формате JSON. Серверная и клиентская части общаются через протокол RESTfull API в формате данных JSON. Он основывается на представлении объектов в языке JavaScript и его чтение не составляет никакого труда. Посмотрим на один пример ответа сервера.

Если отправить запрос на путь /api/processes/<id процесса>/events, серверная часть выведет список событий для процесса с id равным <id процесса> в формате JSON.

Ответ содержит массив событий. У каждого события есть идентификатор, название, описание, идентификатор процесса, дата создания и информация об опубликованности данного события.

```
{
  "events": [{
    "id": "87",
    "name": "Event name",
    "description": "Event desription.",
    "process_id": "24",
    "date": "2013-04-21",
    "published": "0",
```

Комментарии состоят из массива комментариев и поля count — количества комментариев.

```
"comments": {
  "data": [{
```

Каждый комментарий состоит из данных о комментарии, таких как идентификатор, текст, дата создания, информация о пользователе, создавшем комментарий и о событии, которому принадлежит комментарий.

```
    "comment": {
      "id": "84",
      "text": "Comment.",
      "created": "2013-04-29 18:00:01",
      "user": {
        "id": "27",
        "name": "Alimov",
        "surname": "Rustam",
        "second_name": "Fakhriddinovich"
      },
      "event": {
        "id": "87",
        "date": "2013-04-21",
        "process": {
          "id": "24",
          "start_date": "2013-04-14",
          "subscribed": 0,
          "permissions": {
            "subscribe": false,
            "read": true
          }
        }
      }
    },
    "permissions": {
      "canDelete": true
    }
  }],
  "count": "1"
},
```

Далее идет информация о создателе события, список его тегов и список файлов, прикрепленных к событию.

```
    "creator": {
      "id": "27",
      "name": "Alimov",
      "surname": "Rustam",
      "second_name": "Fakhriddinovich"
    },
    "tags": ["tag1", "tag2"],
    "uploads": [{
      "id": "245",
      "real_name": "file1.png",
      "size": "15813",
      "type": "image\/png"
    }]
  }],
  "count": "1"
}
```

В приведенном примере присутствует только одно событие, его поля, комментарии, файлы, теги, информация о создавшем событие пользователе и настройки доступа к данному событию для текущего пользователя.

Модели. Yii предоставляет классы, абстрагирующие доступ к базе данных в объектно-ориентированной форме. Наследуемые от них классы называются моделями. В системе для каждой из таблиц в базе данных существует по одной модели.

Модели существенно упрощают работу и с базой данных, и с данными. Среди функций моделей можно особо выделить следующие:

- запросы к базе данных на сохранение, изменение и получение данных в объектно-ориентированной форме;
- валидация данных, поступающих в систему;
- преобразование данных перед их сохранением в базу данных и наоборот, после получения из базы данных.

В качестве примера использования модели Yii, можно привести модель событий, находящуюся в файле /protected/models/Event.php.

Класс модели наследуется от класса RestActiveRecord и содержит различные методы, обеспечивающие вышеперечисленную функциональность. Например, метод `afterFind` преобразует дату события после того, как данные загружаются из базы данных.

Модели можно дополнять методами, которых нет в других моделях: `userCanComment`, `userCanEditEvent`, `userCanDeleteEvent`, `prepareComments`, `prepareTags`, `prepareUploads`, которые используются для проверки доступа пользователя к данному событию, а также для подготовки события к выводу, написаны непосредственно для данной модели и не переопределяют методы родительского класса.

Структура базы данных

В качестве СУБД была использована реляционная СУБД MySQL — проверенная временем, надежная и современная.

Рассмотрим таблицу пользователей системы: user. В ней хранятся учетные записи пользователей и некоторая информация о них. Записи создаются при регистрации пользователей. Таблица состоит из следующих полей: id — уникальный идентификатор, name — имя, surname — фамилия, second_name — отчество, sex — пол, birth_date — дата рождения, email — электронная почта, password — sha1 хэш пароля в комбинации с «солью», salt — «соль» для пароля (для большей безопасности, представляет собой хэш от некоторой случайной строки), description — некоторая информация о себе, degrees — научные степени и достижения, created — дата создания записи, modified — дата последнего изменения записи, filled_profile — информация о заполненности всех необходимых полей записи (система становится доступной после заполнения таких полей, как имя, фамилия, отчество, дата рождения и пол), social_id — уникальный идентификатор от стороннего сервиса авторизации (например, Facebook, Google и т. п.), type — тип регистрации.

Далее идет таблица научно-образовательных процессов системы: process. Каждая запись процесса является объединяющей для его событий. Состоит из следующих полей: id — уникальный идентификатор, name — название, super_user_id — уникальный идентификатор пользователя, создавшего процесс, start_date — дата начала процесса, end_date — дата окончания процесса, is_public — доступ к процессу (открытый, закрытый), published — опубликованность про-

цесса, created — дата создания записи, modified — дата последнего изменения записи.

Следующей рассмотрим таблицу руководителей процессов: process_master. Она является связывающей таблицей пользователей с процессами (связь много ко многим). Состоит из следующих полей: id — уникальный идентификатор, process_id — уникальный идентификатор процесса, user_id — уникальный идентификатор пользователя, created — дата создания записи, modified — дата последнего изменения записи.

Таблица process_member участников закрытых процессов является связывающей таблицей пользователей с процессами (связь много ко многим). Состоит из следующих полей: id — уникальный идентификатор, process_id — уникальный идентификатор процесса, user_id — уникальный идентификатор пользователя, created — дата создания записи, modified — дата последнего изменения записи.

Таблица подписок на процессы process_subscriber является связывающей таблицей пользователей с процессами (связь много ко многим). Состоит из следующих полей: id — уникальный идентификатор, process_id — уникальный идентификатор процесса, user_id — уникальный идентификатор пользователя, created — дата создания записи, modified — дата последнего изменения записи.

Табица тегов: tag. Теги используются в системе для краткого описания событий и процессов. Состоит из следующих полей: id — уникальный идентификатор записи, name — название тега, created — дата создания записи, modified — дата последнего изменения записи.

Таблица загрузок файлов upload содержит такую информацию о каждом загруженном файле, как оригинальное название, mime тип, размер и т. д. Состоит из следующих полей: id — уникальный идентификатор записи, user_id — уникальный идентификатор пользователя, загрузившего файл, is_avatar — информация о том, является ли файл аватаром для какого-нибудь пользователя, name — названия файла в файловой системе сервера, real_name — название файла при загрузке, size — размер файла в байтах, type — mime тип файла, tmp — временный ли файл (если временный, то удаляется через некоторое время), created — дата создания записи, modified — дата последнего изменения записи, is_small_avatar — если аватар, то является ли он малым.

Таблица событий event состоит из следующих полей: id — уникальный идентификатор записи, name — название, description — описание, process_id — уникальный идентификатор процесса, creator_id — уникальный идентификатор пользователя, создавшего событие, date — дата события, published — опубликовано ли событие, created — дата создания записи, modified — дата последнего изменения записи.

Таблица комментариев событий event_comment состоит из следующих полей: id — уникальный идентификатр записи, user_id — уникальный идентификатор пользователя, создавшего комментарий, event_id — уникальный идентефикатор события, text — текст, created — дата создания записи, modified — дата последнего изменения записи.

Таблица тегов событий event_tag является связывающей таблицей тегов с событиями (связь много ко многим). Состоит из следующих полей: id — уникальный идентификатор записи, event_id — уникальный идентификатор события, tag_id — уникальный идентификатор тега, created — дата создания записи, modified — дата последнего изменения записи.

Таблица загруженных фалов событий event_upload является связывающей таблицей загрузок файлов с событиями (связь много ко многим). Состоит из следующих полей: id — уникальный идентификатор записи, event_id — уникальный идентификатор события, upload_id — уникальный идентификатор файла, created — дата создания записи, modified — дата последнего изменения записи.

Клиентская часть

Для системы разработан веб-клиент, доступный с любого компьютера с современным браузером и подключением к интернету. Он предоставляет удобный интерфейс для работы с системой и позволяет использовать ее функциональность в полной мере. К клиентской части предъявлены следующие требования: она должна представлять собой «одностраничное» приложение, отказ от сложных для заполнения форм, максимальное следование правилу, по которому работа с сущностями выполняется «на лету». Каждое действие, как например создание нового процесса, должно состоять из минимального числа действий (желательно одним кликом). Также приложение

должно быть доступно для любого компьютера с современным браузером.

«Одностраничное» приложение — это такой сайт, который не перезагружается каждый раз, когда осуществляется переход на различные разделы. Например, Gmail — это одностраничное приложение.

После некоторых сравнений и тестов, для разработки веб-клинета были выбраны следующие фреймворки и библиотеки (и языки программирования, на которых они написаны):

- AngularJS — MVVM JavaScript фреймворк для создания современных «одностраничных» приложений (JavaScript, HTML);
- jQuery 2.0 — библиотека для упрощения работы с различными браузерами, помогающая с решением проблемы несовместимости API различных браузеров (JavaScript);
- Bootstrap — библиотека для верстки страниц (HTML5, CSS, JavaScript);
- HTML5 Boilerplate — библиотека для основы верстки страниц (HTML5, CSS, JavaScript);
- Modernizr — библиотека для проверки функциональности браузеров. Помогает в учете различий в API браузеров (JavaScript).

Фреймворк AngularJS

Основной рабочей лошадкой среди перечисленных фреймворков и библиотек является MVVM JavaScript фреймворк AngularJS. MVVM (Model, View, View, Model) — это тип фреймворка, который позволяет связывать данные с их представлением в обе стороны (при изменении данных в JavaScript — изменяется и представление, и наоборот). В некотором смысле подходы к решению вопросов на самом высоком уровне AngularJS и Yii схожи. AngularJS также предоставляет возможность связывать пути с их обработчиками, у него есть такие понятия, как контроллер и модель. Но вместе с тем они сильно отличаются: у них совершенно разные методы представления данных, AngularJS является более динамичным по сравнению с Yii. Также AngularJS поддерживает внедрение зависимостей, что сильно упрощает код приложений, так как это позволяет не задумываться о загрузке модулей и т. п. Разрабатывается сотрудниками компании Google и распространяется под лицензией MIT.

Описание приложения AngularJS. Приложение AngularJS в системе описано в файле /js/app.js.

```
"use strict";
angular.module('dvinemnauku', ['ngResource',
 'strap.directives', 'infinite-scroll', 'ui']).
    config(['$routeProvider', function ($routeProvider) {/*
 */}]).
    config(['$httpProvider', function($httpProvider) {/*
 */}]).
    run(function ($rootScope, AccessControl, User,
 StartUser, $location) {/* */});
```

Из кода видно, что создаваемое приложение называется `dvinemnauku`, и зависит от таких модулей, как `ngResource`, `strap.directives`, `infinite-scroll`, `ui`. Эти модули выполняют следующие задачи:

- ngResource — предоставляет удобные средства для работы с RESTfull API;
- strap.directives — директивы для компонентов Bootstrap;
- infinite-scroll — директива для создания бесконечного скролла, когда при достижении конца страницы загружаются новые данные;
- ui — дополнительные полезные директивы, как например, установщики обработчиков событий `blur` и `focus`.

Методы `config` и `run`, соответственно, вызываются во время инициализации приложения и после него.

Связывание путей с обработчиками. Как и в случае с Yii, в AngularJS связывание путей с обработчиками задается в конфигурациях приложения. Приведем пример некоторых из них.

```
config(['$routeProvider', function ($routeProvider) {
    $routeProvider.
        when('/processes', { templateUrl:
 'views/processes.html', controller: 'ProcessesCtrl' }).
```

```
        when('/processes/create', { templateUrl:
 'views/processes.create.html', controller:
 'ProcessesCreateCtrl' }).
        when('/processes/view/:id', { templateUrl:
 'views/processes.view.html', controller:
 'ProcessesViewCtrl' }).
}])
```

Для примера рассмотрим связывание /processes/view/<id процесса> с его обработчиком.

```
when('/processes/view/:id', { templateUrl:
'views/processes.view.html', controller:
'ProcessesViewCtrl' }).
```

Здесь `'/processes/view/:id'` означает, что если путь имеет вид `/processes/view/:id`, то должен вызваться обработчик, указанный в параметре `controller: 'ProcessesViewCtrl'` с шаблоном вида `templateUrl: 'views/processes.view.html'`.

Контроллеры. Важным понятием в AngularJS является контроллер, который служит как связующее звено между данными и их представлением. Действия пользователя на странице, такие как, клик на какой-либо элемент, нажатие на клавишу и т. п., обрабатываются в контроллерах.

В качестве примера приведем контроллер для вывода и редактирования процесса из файла /js/controllers/processes.view.js.

```
"use strict";
angular.module('dvinemnauku').controller('ProcessesViewCtrl',
function ($scope, $routeParams, Processes) {
    $scope.createEvent = function () {/* */};
    $scope.nextEventsPage = function () {/* */};
    $scope.saveMaster = function (user, success, error) {/*
 */};
    $scope.deleteMaster = function (user, success, error) {/*
 */};
    $scope.saveMember = function (user, success, error) {/*
```

```
 */};
    $scope.deleteMember = function (user, success, error) {/*
 */};
});
```

Данный контроллер называется `ProcessesViewCtrl` и зависит от следующий модулей:

- $scope — объект для связывания данных и представления;
- $routeParams — параметры пути;
- Processes — сервис для работы с RESTfull API для процессов, событий и комментариев.

Объект `$scope` нужен для двухстороннего связывания данных с их представлениями. Например, если создать в `$scope` параметр и привязать к нему какое-нибудь текстовое поле, то при изменении текстового поля, также поменяется и параметр и наоборот, при изменении параметра, меняется и содержимое текстового поля. Это очень сильно упрощает разработку приложения, так как не нужно вручную синхронизировать данные.

`$scope` также используется для внедрения функций в представления приложения, чтобы можно было указать их в качестве обработчиков событий. Например, рассмотрим функцию `$scope.createEvent` для создания нового события в процессе из приведенного ранее кода.

```
$scope.createEvent = function () {
    $scope.creatingNewEvent = true;
    Processes.event.create({
        pid: $scope.process.id,
        empty: 1
    }, {}, function (data) {
        $scope.clearSearch();
        $scope.creatingNewEvent = false;
    }, function (error) {
        alert('Can not create new event.');
        $scope.creatingNewEvent = false;
    });
};
```

При вызове данной функции осуществляется запрос к RESTfull API на создание пустого события в текущем процессе. В случае успеха запроса обновляется список событий, а в случае ошибки выводится сообщение пользователю.

Сервисы. Сервисы в AngularJS — это то, что позволяет расширять возможности фреймворка. В клиенте сервисы использованы в качестве связующего звена между клиентом и сервером. Для примера приведем сервис Processes из файла /js/services/processes.js.

```
"use strict";
angular.module('dvinemnauku').factory('Processes',
 function ($resource, ApiUrl) {
    return {
        event: $resource(
ApiUrl + '/processes/:pid/events/:eid', {}, {
            create: { method: 'POST' },
            list: { method: 'GET' },
            update: { method: 'PUT' },
            remove: { method: 'DELETE' }
        }),
    };
});
```

Тут описывается сервис под названием `Processes`, который используется для работы с RESTfull API. В коде видно, что для пути запроса `'/processes/:pid/events/:eid'` существуют различные методы запросов, из названия которых ясно их предназначение.

Директивы. Еще одним важным компонентом AngularJS являются директивы. Они позволяют использовать различные элементы интерфейса многократно и помогают добиться максимальной интерактивности при работе с приложением. Это делает AngularJS более декларативным, чем императивным. То есть нужно писать, что делать, а не как это делать. Например, язык C — это императивный язык, а язык SQL — декларативный. AngularJS находится где-то посередине.

Приведем пример директивы комментариев событий, которая находится в файле /js/directives/dv-comments.js.

```
"use strict";
angular.module('dvinemnauku').directive('dvComments',
function ($timeout, Processes) {
    return {
        restrict: 'A',
        scope: {
            process: '=dvCommentsProcess',
            event: '=dvCommentsEvent',
            comments: '=dvCommentsArray',
            count: '=dvCommentsCount'
        },
        templateUrl: 'views/dv-comments.html',
        link: function (scope, elem, attrs) {/* */ }
    };
});
```

Директива — это объект, который может содержать различные поля и методы, но в данном случае описаны следующие:

- restrict — задает правило применения директивы (в данном случае, директиву можно применять только в качестве атрибута);
- scope — задает двустороннее связывание директивы и ее окружающего представления;
- templateUrl — адрес шаблона для директивы;
- link — функция для связывания кода JavaScript и представления директивы.

Остальные библиотеки

jQuery 2.0. jQuery — это кросс-браузерная библиотека, разработанная для упрощения разработки приложений на JavaScript и HTML. Распространяется под лицензией MIT и вляется самой распространенной JavaScript библиотекой на сегодняшний день.

jQuery существенно облегчает работу при работе с DOM документа, при создании анимации и различных эффектов на веб-странице, обработке событий и при отправке AJAX запросов, так как учитывает все различия в API браузеров. Таким образом, нет необходимости писать лишний код для поддержки различных браузеров.

Bootstrap. Bootstrap — это библиотека, разрабатываемая сотрудниками компании Twitter и распространяемая под лицензией Apache License v2.0. Очень сильно упрощает верстку веб-страниц. Предоставляет множество различных компонент и плагинов. Поддерживает большое число браузеров, что явялется большим преимуществом перед остальными подобными библиотеками.

HTML5 Boilerplate и Modernizr. Данные библиотеки разрабатываются сотрудниками компании Google и служат основой для верстки веб-страниц и для проверки функциональности браузера. Последнее необходимо для сайтов, которые поддерживают некоторые старые версии браузеров (для чего существует возможность использования старых методов, если не поддерживается какая-либо функция) или выводят некоторые соответсвующие сообщения пользователю о том, что его браузер не поддерживается. Также библиотека помогает в структурировании HTML5 приложений.

Вспомогательные модули

Серверная часть

На данный момент в Yii нет поддержки создания RESTfull API. Но его можно расширить так, чтобы он поддерживал данный протокол. В Yii существует поддержка расширения функционала в объектно-ориентированной форме. Для системы был написан расширенный класс контроллера, расширенный класс модели и конвертирование данных в массивы.

Расширенный класс контроллера

Стандартный класс контроллера Yii разработан для вывода сгенерированных HTML страниц. Однако для системы нужно, чтобы контроллер выводил данные в формате JSON и при этом выставлял необходимые HTTP статусы.

Вывод ошибок. Расширенный контроллер выводит ошибки в формате JSON и выставляет при этом правильные HTTP статусы, так как зачастую приложения-клиенты сервисов только благодаря им и

узнают о случившейся ошибке. За вывод ошибок отвечают три метода: `responseError`, `onException` и `actionError`.

Метод `responseError` получает на входе HTTP статус, сообщение об ошибке и некоторые дополнительные данные для вывода. В конце своей работы этот метод завершает работу приложения Yii.

```
public function responseError($code, $message = "",
$info = null)
{
    if ($code < 400 || $code > 599) {
        $code = 500;
    }
    $this->HTTPStatus = $this->getProperHttpStatus($code,
500);
    $this->renderJson(array(
        'message' => $message,
        'info' => $info
    ));
    Yii::app()->end();
}
```

Метод `onException` вызывается в случае неотловленного иключения. Он получает данные об исключении и вызывает метод `responseError` с соответсвующими параметрами.

```
public function onException($event)
{
    $message = '';
    if (!$this->developmentFlag &&
        ($event->exception->statusCode == 500 ||
        is_null($event->exception->statusCode))) {
        $message = 'Internal Server Error';
    } else {
        $message = $event->exception->getMessage();
    }
    $errorCode = (!isset($event->exception->statusCode) ||
        is_null($event->exception->statusCode)) ? 500 :
        $event->exception->statusCode;
```

```
    $this->responseError($errorCode, $message);
}
```

Метод `actionError` отвечает за вывод ошибок в логике Yii. Например, когда пользователь отправляет запрос на несуществующий путь, вызывается именно данный метод. Он, как и `onException`, использует метод `responseError` для вывода информации об ошибках.

```
public function actionError()
{
    $error = Yii::app()->errorHandler->error;
    if ($error) {
        $this->responseError($error['code'],
        $error['message'], $error);
    }
}
```

Для вывода ошибок используется метод `responseError`, а остальные методы это просто переопределенные методы Yii, выводящие информацию об ошибках в формате JSON.

Вывод данных в ответ на успешный запрос. Для вывода ответов на успешные запросы используется метод `outputHelper`. В отличии от метода `responseError`, данному методу можно передать модели Yii, которые он в дальнейшем сконвертирует в массив и выведет в формате JSON.

```
public function outputHelper($results, $HTTPStatus = 200)
{
    $this->HTTPStatus = $HTTPStatus;
    $converter = new EToArrayConverter();
    $this->renderJson($converter->convert($results,  array(
        'relname' => '',
        'scenario' => 'output'
    )));
    Yii::app()->end();
}
```

Расширенный класс модели

Стандартный класс для моделей в Yii — это класс `CActiveRecord`. Однако часто необходимо к стандартным полям модели добавить еще и дополнительные поля, например, информацию о настройках доступа к сущности. Для этого был написан класс RestActiveRecord, добавляющий необходимую функциональность.

```
class RestActiveRecord extends CActiveRecord
{
    public $additionalFields = array();
    public function addField($name, $value)
    {
        $this->additionalFields[$name] = $value;
    }
    public function removeField($name)
    {
        if (isset($this->additionalFields[$name])) {
            unset($this->additionalFields[$name]);
        }
    }
    public function getAttributes($names = true)
    {
        $attributes = parent::getAttributes($names);
        return array_merge($this->additionalFields,
        $attributes);
    }
}
```

Метод `addField` добавляет поле, `removeField` удаляет поле, а метод `getAttributes` в дополнение к остальным полям выводит и те, которые были добавлены программистом.

Клиентская часть

В AngularJS повторно используемые компоненты интерфейса создаются в виде директив. Часто также приходится оборачивать в директивы какие-либо плагины и элементы пользовательского интерфейса, такие как, например, календари. Для системы ДвинемНауку

было создано несколько директив, некоторые из них в качестве обертки для уже существующих элементов.

Списки руководителей и пользователей процессов

Списки руководителей и списки пользователей по реализации как две капли похожи друга на друга. Поэтому, было решено для списка пользователей создать отдельную директиву без жесткой привязки к какому-либо из списков пользователей. Директива называется `dvInputUsers`. Она необходима для создания интерфейса добавления, вывода и удаления пользователей системы в каких-либо списках. Непосредственные запросы на добавление и удаление осуществляются в контроллерах. Для добавления пользователя в директиве был создан автокомплит с подсказками для удобной и быстрой работы.

Директива находится в файле /js/directives/dv-input-users.js.

Теги событий

Списки тегов внешне во многом похожи на списки руководителей и пользователей процессов. Это сделано с целью минимизировать различные виды элементов интерфейса, что упрощает работу с системой.

Директива находится в файле /js/directives/dv-input-tags.js.

Выбор дат процессов и событий

Одним из главных требований к веб-клиенту является минимальность количества необходимых действий для работы с системой. Поэтому выбор календаря был важным вопросом. После некоторого просмотра доступных вариантов, в кандидатах на добавление в систему остались два календаря: Bootstrap Datepicker и Pickadate. Для первого уже существовала директива AngularJS, а второй наиболее лучшим образом подходил под требование минимальности действий. Было решено выбрать второй календарь и написать для него новую директиву под названием dvPickadate.

Директива находится в файле /js/directives/dv-pickadate.js.

Комментирование событий

Комментирование событий встречается в веб-клиенте в двух местах. При этом код для комментирования довольно сложный. Поэто-

му очевидным было решение написать отдельную директиву. Директива поддерживает такие функции, как подгрузка всех комментариев (в начале загружаются только 3 последних комментария), удаление комментария и добавление нового комментария. Решение оказалось правильным и помогло избежать дублирования кода и траты дополнительного времени.

Директива находится в файле /js/directives/dv-comments.js.

Загрузка файлов

Загрузка файлов состоит из двух частей: из серверной и из клиентской. Это довольно сложная задача. Особенно если учесть требования к системе. Было решено использовать для этого дополнение к jQuery под названием jQuery-File-Upload. Рассмотрим его серверную и клиентскую части отдельно.

Серверная часть. Так как серверная часть расширения является сильно обобщенной, к ней пришлось добавлять некоторые модификации и дополнения. Формирование названий файлов было полностью переписано под нужды системы ДвинемНауку. Также добавлена интеграция с существующей моделью для загрузок. В базе данных хранятся настоящие названия файлов, их mime тип, размер и название в файловой системе сервера. На сервере файлы хранятся под случайно сгенерированными названиями из 7 символов.

Модифицированный класс серверной части расширения находится в файле /protected/components/UploadHandler.php.

Клиентская часть. Для расширения на стороне клиента была написана новая директива. Ее особенность заключается в том, что работа с файлами требует минимальных действий. Достаточно простого перетаскивания файлов в специальную область на веб-странице, после чего они автоматически загружаются на сервер и добавляются к событию. Директива называется dvFileUpload.

Заключение

В результате работы создана система онлайн-сопровождения научно-образовательных процессов. Процессы создаются одним на-

жатием кнопки и редактируются максимально быстрым и простым способом: достаточно лишь кликнуть на поле, внести изменения и кликнуть в другом месте на странице. Таким же способом осуществляется редактировани событий. Кроме того, к событиям можно загружать файлы простым перетаскиванием их в определенную область события. Каждое событие можно комментировать.

Разработана общая архитектура системы, ее основные компоненты и приняты технические решения для разделения серверной и клиентских частей. Данного рода разделение позволяет создавать клиенты к системе совершенно не затрагивая ее серверную часть. Протокол общения частей — RESTfull API с форматом данных JSON. На серверной стороне используется фреймворк Yii и СУБД MySQL. В клиенсткой части используется фреймворк AngularJS и другие библиотеки.

Разработаны вспомогательные модули. Для Yii были написаны такие модули, как расширенный класс контроллера, расширенный класс модели, конвертирование данных в массивы. Для клиенсткой части были написаны директивы для списков руководителей процессов, тегов событий, выбора дат и комментирования событий на странице процесса и на стене. Необходимо особо выделить модули для загрузки файлов. Они реализованы и на стороне клиента, и на стороне сервера.

Список литературы

[1] Masse M. REST API Design Rulebook. — O'Reilly Media, 2011.

[2] Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming. — No Starch Press, 2011.

[3] Makarov A. Yii 1.1 Application Development Cookbook. — Packt Publishing, 2011.

[4] Conrad A. 3 Reasons to Choose AngularJS for Your Next Project. — 2012. http://net.tutsplus.com/tutorials/javascript-ajax/3-reasons-to-choose-angularjs-for-your-next-project/.

[5] Ford B. Building Huge Apps with AngularJS. — 2012. http://briantford.com/blog/huuuuuge-angular-apps.html.