**Лабораторная работа №1**

# **Установка Yii**

Вы можете установить Yii двумя способами: используя [Composer](https://getcomposer.org/) или скачав архив.

### **Установка через Composer**

Если Composer еще не установлен это можно сделать по инструкции на [getcomposer.org](https://getcomposer.org/download/), или одним из нижеперечисленных способов.

На Windows, скачайте и запустите [Composer-Setup.exe](https://getcomposer.org/Composer-Setup.exe).

В случае возникновения проблем читайте [раздел "Troubleshooting" в документации Composer](https://getcomposer.org/doc/articles/troubleshooting.md). Если вы только начинаете использовать Composer, рекомендуем прочитать как минимум [раздел "Basic usage"](https://getcomposer.org/doc/01-basic-usage.md).

Предполагается, что Composer установлен [глобально](https://getcomposer.org/doc/00-intro.md#globally). То есть он доступен через команду composer. Если вы используете composer.phar из локальной директории, изменяйте команды соответственно.

Если у вас уже установлен Composer, обновите его при помощи *composer self-update.*

**Примечание:**Во время установки Yii Composer запрашивает довольно большое количество информации через Github API. Количество запросов варьируется в зависимости от количества зависимостей вашего проекта и может превысить ограничения **Github API**. Если это произошло, Composer спросит логин и пароль от Github. Это необходимо для получения токена для Github API. На быстрых соединениях это может прозойти ещё до того, как Composer сможет обработать ошибку, поэтому мы рекомендум настроить токен доступа до установки Yii. Инструкции приведены в [документации Composer о токенах Github API](https://getcomposer.org/doc/articles/troubleshooting.md#api-rate-limit-and-oauth-tokens).

После установки Composer устанавливать Yii можно запустив следующую команду в папке доступной через веб:

*composer create-project --prefer-dist yiisoft/yii2-app-basic basic*

Эта команда устанавливает последнюю стабильную версию Yii в директорию basic. Если хотите, можете выбрать другое имя директории.

**Информация:**Если команда composer create-project не выполняется нормально, попробуйте обратиться к [разделу "Troubleshooting" документации Composer](https://getcomposer.org/doc/articles/troubleshooting.md). Там описаны другие типичные ошибки. После того, как вы исправили ошибку, запустите composer update в директории basic.

**Подсказка:**Если вы хотите установить последнюю нестабильную ревизию Yii, можете использовать следующую команду, в которой присутствует [опция stability](https://getcomposer.org/doc/04-schema.md#minimum-stability):

*composer create-project --prefer-dist --stability=dev yiisoft/yii2-app-basic basic*

Старайтесь не использовать нестабильную версию Yii на рабочих серверах потому как она может внезапно поломать код.

Установка из архива

Установка Yii из архива состоит из трёх шагов:

1. Скачайте архив с [yiiframework.com](http://www.yiiframework.com/download/);
2. Распакуйте скачанный архив в папку, доступную из Web.
3. В файле config/web.php добавьте секретный ключ в значение cookieValidationKey (при установке через Composer это происходит автоматически):

*// !!! insert a secret key in the following (if it is empty) - this is required by cookie validation*

*'cookieValidationKey' => 'enter your secret key here',*

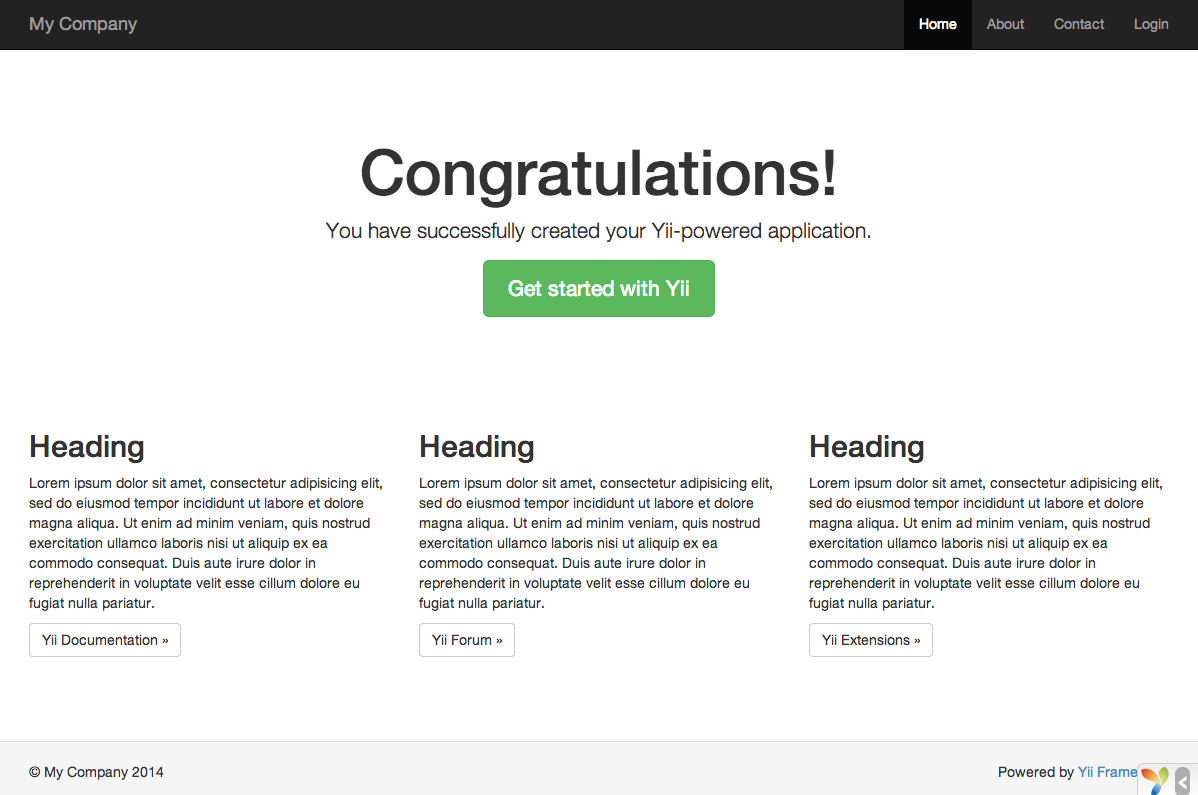
Проверка установки

После установки приложение будет доступно по следующему URL:

*http://localhost/basic/web/index.php*

Здесь подразумевается, что вы установили приложение в директорию basic в корневой директории вашего веб сервера сервер работает локально (localhost). Вам может потребоваться предварительно его настроить.

Внешний вид приложения приведен ниже.



Для корректной работы фреймворка вам необходима установка PHP, соответствующая его минимальным требованиям. Основное требование — PHP версии 5.4 и выше. Если ваше приложение работает с базой данных, необходимо установить [расширение PHP PDO](http://www.php.net/manual/ru/pdo.installation.php) и соответствующий драйвер (например, pdo\_mysql для MySQL).

Создание контроллера и представления

Рассмотрим, как создать новую страницу с надписью «Привет». В процессе решения задачи вы создадите действие контроллера и представление:

* Приложение обработает запрос и передаст управление соответствующему действию;
* Действие, в свою очередь, отобразит представление с надписью "Привет" конечному пользователю.

Создание Действия

Для нашей задачи потребуется [действие](https://www.yiiframework.com/doc/guide/2.0/ru/structure-controllers) say, которое читает параметр message из запроса и отображает его значение пользователю. Если в запросе не содержится параметра message, то действие будет выводить «Привет».

**Информация:**[Действия](https://www.yiiframework.com/doc/guide/2.0/ru/structure-controllers) могут быть запущены непосредственно пользователем и сгруппированы в [контроллеры](https://www.yiiframework.com/doc/guide/2.0/ru/structure-controllers). Результатом выполнения действия является ответ, который получает пользователь.

Действия объявляются в [контроллерах](https://www.yiiframework.com/doc/guide/2.0/ru/structure-controllers). Для простоты, вы можете объявить действие say в уже существующем контроллере SiteController, который определен в файле класса controllers/SiteController.php:

**<?php**

**namespace** **app**\**controllers**;

**use** **yii**\**web**\**Controller**;

**class** **SiteController** **extends** **Controller**

{

// ...существующий код...

**public** **function** **actionSay**($message = 'Привет')

{

**return** $this->render('say', ['message' => $message]);

}

}

В приведенном коде действие say объявлено как метод actionSay в классе SiteController. Yii использует префикс action чтобы различать методы-действия и обычные методы. Название после префикса action считается идентификатором соответствующего действия.

**Информация:**Идентификаторы действий задаются в нижнем регистре. Если идентификатор состоит из нескольких слов, они соединяются дефисами, то есть create-comment. Имена методов действий получаются путём удаления дефисов из идентификатора, преобразования первой буквы каждого слова в верхний регистр и добавления префикса action. Например, идентификатор действия create-comment соответствует методу actionCreateComment.

Метод действия принимает параметр $message, который по умолчанию равен "Привет". Когда приложение получает запрос и определяет, что действие say ответственно за его обработку, параметр заполняется одноимённым значением из запроса.

Внутри метода действия, для вывода отображения [представления](https://www.yiiframework.com/doc/guide/2.0/ru/structure-views) с именем say, используется метод [render()](https://www.yiiframework.com/doc/api/2.0/yii-base-controller" \l "render()-detail). Для того, чтобы вывести сообщение, в отображение передаётся параметр message. Результат отображения при помощи return передаётся приложению, которое отдаёт его пользователю.

Создание представления

[Представления](https://www.yiiframework.com/doc/guide/2.0/ru/structure-views) являются скриптами, которые используются для формирования тела ответа. Для нашего приложения вы создадите представление say, которое будет выводить параметр message, полученный из метода действия:

**<?php**

**use** **yii**\**helpers**\**Html**;

**?>**

**<?**= Html::encode($message) **?>**

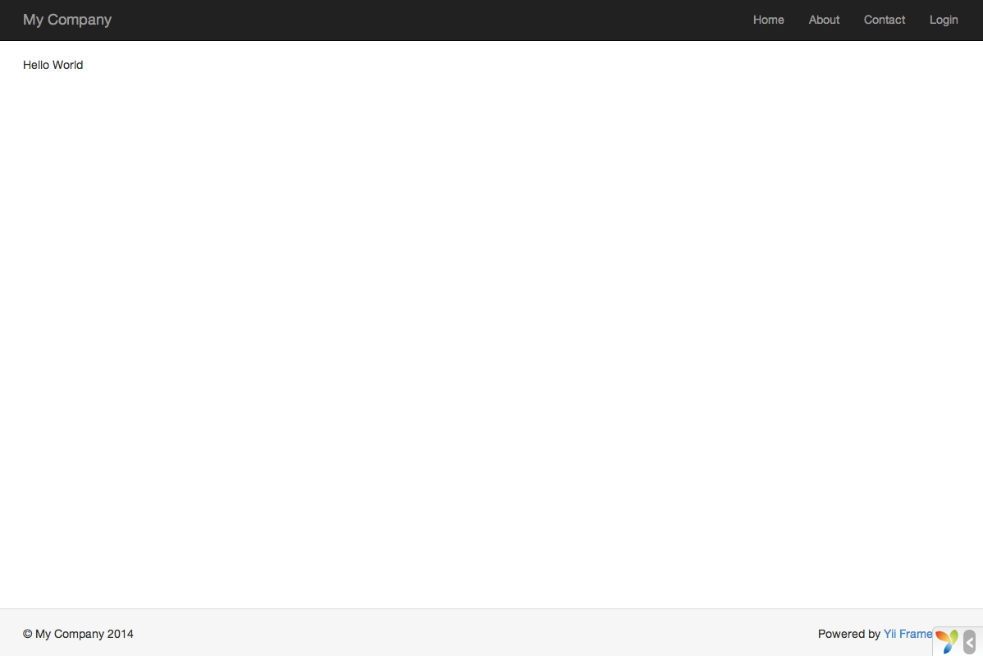
Представление say должно быть сохранено в файле views/site/say.php. Когда метод [render()](https://www.yiiframework.com/doc/api/2.0/yii-base-controller#render()-detail) вызывается в действии, он будет искать PHP файл с именем вида views/ControllerID/ViewName.php.

Стоит отметить, что в коде выше параметр message [экранируется для HTML](https://www.yiiframework.com/doc/api/2.0/yii-helpers-basehtml#encode()-detail) перед выводом. Это обязательно так как параметр приходит от пользователя, который может попытаться провести [XSS атаку](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D1%81%D0%B0%D0%B9%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%B8%D0%BD%D0%B3) путём вставки зловредного JavaScript кода.

Вы можете дополнить представление say HTML тегами, текстом или кодом PHP. Фактически, представление say является простым PHP скриптом, который выполняется методом [render()](https://www.yiiframework.com/doc/api/2.0/yii-base-controller" \l "render()-detail). Содержимое, выводимое скриптом представления, будет передано пользователю приложением.

После создания действия и представления вы можете перейти на новую страницу по следующему URL:

*http://hostname/index.php?r=site%2Fsay&message=Привет+мир*



Будет отображена страница с надписью «Привет мир». Она использует ту же шапку и футер, что и остальные страницы приложения. Если вы не укажете параметр message, то увидите на странице «Привет». Это происходит потому, как message передаётся в метод actionSay() и значение по умолчанию — «Привет».

**Информация:**Новая страница использует ту же шапку и футер, что и другие страницы, потому что метод [render()](https://www.yiiframework.com/doc/api/2.0/yii-base-controller" \l "render()-detail) автоматически вставляет результат представления say в, так называемый,[макет](https://www.yiiframework.com/doc/guide/2.0/ru/structure-views) views/layouts/main.php.

Параметр r в нашем URL требует дополнительных пояснений. Он связан с [маршрутом (route)](https://www.yiiframework.com/doc/guide/2.0/ru/runtime-routing), который представляет собой уникальный идентификатор, указывающий на действие. Его формат ControllerID/ActionID. Когда приложение получает запрос, оно проверяет параметр r и, используя ControllerID, определяет какой контроллер следует использовать для обработки запроса. Затем, контроллер использует часть ActionID, чтобы определить какое действие выполняет реальную работу. В нашем случае маршрут site/say будет соответствовать контроллеру SiteController и его действию say. В результате, для обработки запроса будет вызван метод SiteController::actionSay().

**Информация:**Как и действия, контроллеры также имеют идентификаторы, которые однозначно определяют их в приложении. Идентификаторы контроллеров используют те же правила именования, что и идентификаторы действий. Имена классов контроллеров получаются путём удаления дефисов из идентификатора, преобразования первой буквы каждого слова в верхний регистр и добавления в конец Controller. Например, идентификатор контроллера post-comment соответствует имени класса контроллера PostCommentController.

Работа с формами

Рассмотрим получение данных от пользователя. На странице будет располагаться форма с полями для ввода имени и email. Полученные данные будут показаны на странице для их подтверждения.

Чтобы достичь этой цели, помимо создания [действия](https://www.yiiframework.com/doc/guide/2.0/ru/structure-controllers) и двух [представлений](https://www.yiiframework.com/doc/guide/2.0/ru/structure-views) вы создадите [модель](https://www.yiiframework.com/doc/guide/2.0/ru/structure-models).

Создание модели

В файле models/EntryForm.php создайте класс модели EntryForm как показано ниже. Он будет использоваться для хранения данных, введённых пользователем.

**<?php**

**namespace** **app**\**models**;

**use** **yii**\**base**\**Model**;

**class** **EntryForm** **extends** **Model**

{

**public** $name;

**public** $email;

**public** **function** **rules**()

{

**return** [

[['name', 'email'], 'required'],

['email', 'email'],

];

}

}

Данный класс расширяет класс [yii\base\Model](https://www.yiiframework.com/doc/api/2.0/yii-base-model), который является частью фреймворка и обычно используется для работы с данными форм.

Класс содержит 2 публичных свойства name и email, которые используются для хранения данных, введённых пользователем. Он также содержит метод rules(), который возвращает набор правил проверки данных. Правила, объявленные в коде выше означают следующее:

* Поля name и email обязательны для заполнения;
* В поле email должен быть правильный адрес email.

Если объект EntryForm заполнен пользовательскими данными, то для их проверки вы можете вызвать метод [validate()](https://www.yiiframework.com/doc/api/2.0/yii-base-model" \l "validate()-detail). В случае неудачной проверки свойство [hasErrors](https://www.yiiframework.com/doc/api/2.0/yii-base-model" \l "hasErrors()-detail) станет равным true. С помощью [errors](https://www.yiiframework.com/doc/api/2.0/yii-base-model" \l "getErrors()-detail) можно узнать, какие именно ошибки возникли.

Создание действия

Далее создайте действие entry в контроллере site, точно так же, как вы делали это ранее.

<?php

namespace **app\controllers;**

use **Yii;**

use **yii\web\Controller;**

use **app\models\EntryForm;**

**class SiteController** **extends** **Controller**

{

// ...существующий код...

**public function** **actionEntry()**

{

$model = **new** EntryForm();

**if** ($model->load(Yii::$app->request->post()) && $model->validate()) {

// данные в $model удачно проверены

// делаем что-то полезное с $model ...

**return** $this->render('entry-confirm', ['model' => $model]);

} **else** {

// либо страница отображается первый раз, либо есть ошибка в данных

**return** $this->render('entry', ['model' => $model]);

}

}

}

Действие создает объект EntryForm. Затем оно пытается заполнить модель данными из массива $\_POST, доступ к которому обеспечивает Yii при помощи [yii\web\Request::post()](https://www.yiiframework.com/doc/api/2.0/yii-web-request" \l "post()-detail). Если модель успешно заполнена, то есть пользователь отправил данные из HTML-формы, то для проверки данных будет вызван метод [validate()](https://www.yiiframework.com/doc/api/2.0/yii-base-model" \l "validate()-detail).

Если всё в порядке, действие отобразит представление entry-confirm, которое показывает пользователю введенные им данные. В противном случае будет отображено представление entry, которое содержит HTML-форму и ошибки проверки данных, если они есть.

**Информация:**Yii::$app представляет собой глобально доступный экземпляр-одиночку [приложения](https://www.yiiframework.com/doc/guide/2.0/ru/structure-applications) (singleton). Одновременно это [Service Locator](https://www.yiiframework.com/doc/guide/2.0/ru/concept-service-locator), дающий доступ к компонентам вроде request, response, db и так далее. В коде выше для доступа к данным из $\_POST был использован компонент request.

Создание представления

В заключение создаём два представления с именами entry-confirm и entry, которые отображаются действием entry из предыдущего подраздела.

Представление entry-confirm просто отображает имя и email. Оно должно быть сохранено в файле views/site/entry-confirm.php.

<?php

**use yii\helpers\Html;**

?>

<p>Вы ввели следующую информацию:</p>

<ul>

<li><label>Name</label>: <?= Html::encode($model->name) ?></li>

<li><label>Email</label>: <?= Html::encode($model->email) ?></li>

</ul>

Представление entry отображает HTML-форму. Оно должно быть сохранено в файле views/site/entry.php.

<?php

**use yii\helpers\Html;**

**use yii\widgets\ActiveForm;**

?>

<?php $form = ActiveForm::begin(); ?>

<?= $form->field($model, 'name') ?>

<?= $form->field($model, 'email') ?>

<div class="**form-group**">

<?= **Html::submitButton**('Отправить', ['class' => **'btn btn-primary'**]) ?>

</div>

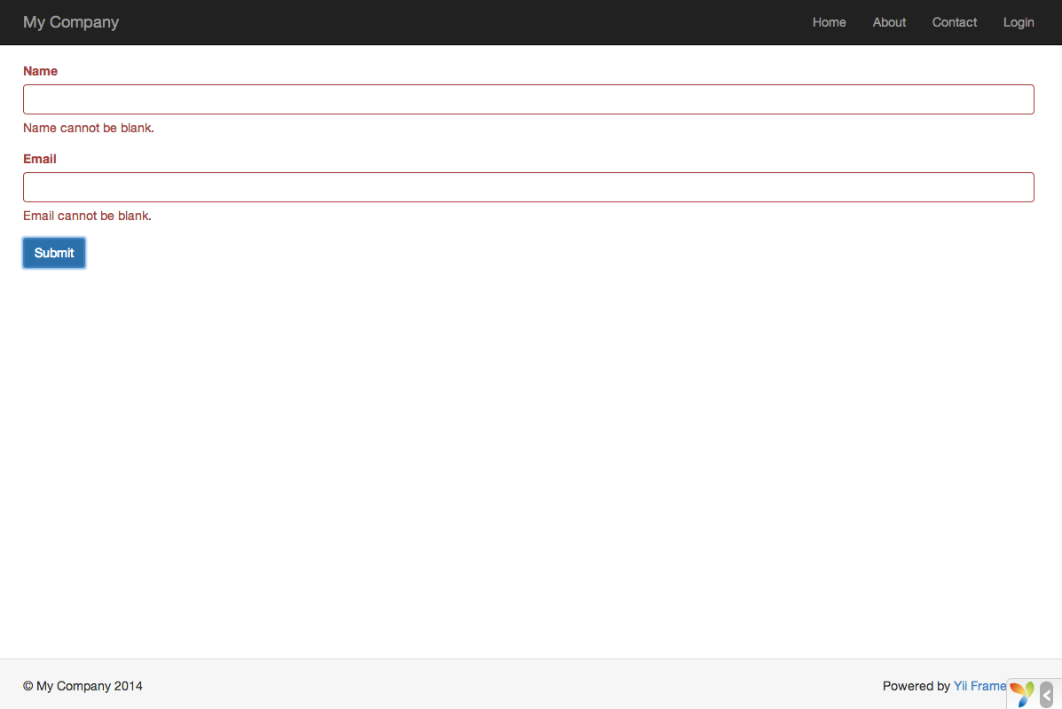
<?**php ActiveForm::end();** ?>

Для построения HTML-формы представление использует мощный [виджет](https://www.yiiframework.com/doc/guide/2.0/ru/structure-widgets) [ActiveForm](https://www.yiiframework.com/doc/api/2.0/yii-widgets-activeform). Методы begin() и end() выводят открывающий и закрывающий теги формы. Между этими вызовами создаются поля ввода при помощи метода [field()](https://www.yiiframework.com/doc/api/2.0/yii-widgets-activeform" \l "field()-detail). Первым идёт поле для "name", вторым — для "email". Далее для генерации кнопки отправки данных вызывается метод [yii\helpers\Html::submitButton()](https://www.yiiframework.com/doc/api/2.0/yii-helpers-basehtml" \l "submitButton()-detail).

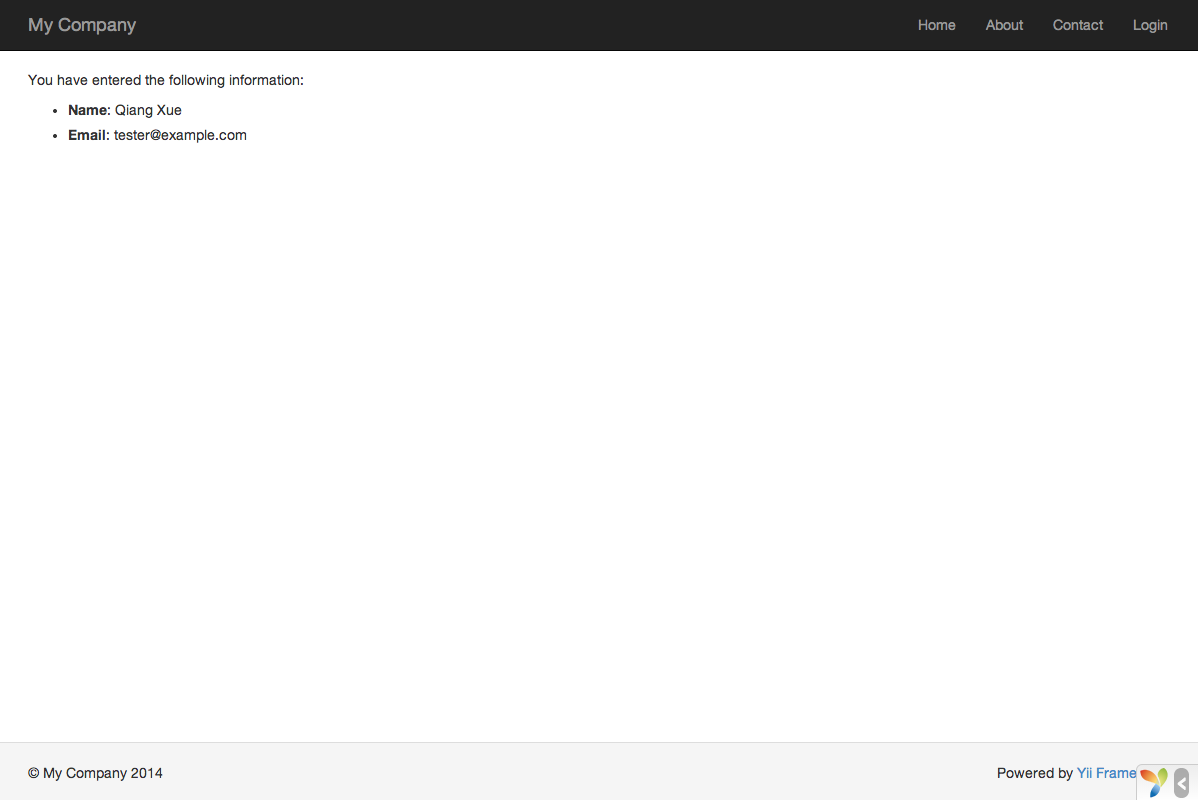
Чтобы увидеть всё созданное в работе, откройте в браузере следующий URL:

*http://hostname/index.php?r=site%2Fentry*

Вы увидите страницу с формой и двумя полями для ввода. Перед каждым полем имеется подпись, которая указывает, какую информацию следует вводить. Если вы нажмёте на кнопку отправки без ввода данных или если вы введете email в неверном формате, вы увидите сообщение с ошибкой рядом с каждым проблемным полем.



После ввода верных данных и их отправки, вы увидите страницу с данными, которые вы только что ввели.



Проверка данных на самом деле происходит и на стороне клиента при помощи JavaScript, и на стороне сервера.[yii\widgets\ActiveForm](https://www.yiiframework.com/doc/api/2.0/yii-widgets-activeform) достаточно продуман, чтобы взять правила проверки, которые вы объявили в EntryForm, преобразовать их в JavaScript код и использовать его для проведения проверок. На случай отключения JavaScript в браузере валидация проводится и на стороне сервера как показано в методе actionEntry(). Это даёт уверенность в том, что данные корректны при любых обстоятельствах.

Подписи для полей генерируются методом field(), на основе имён свойств модели. Например, подпись Name генерируется для свойства name. Вы можете модифицировать подписи следующим образом:

**<?**= $form->field($model, 'name')->label('Ваше имя') **?>**

**<?**= $form->field($model, 'email')->label('Ваш Email') **?>**

**Информация:**В Yii есть множество виджетов, позволяющих быстро строить сложные и динамичные представления. Многое из представлений можно вынести в виджеты, чтобы использовать это повторно в других местах и упростить тем самым разработку в будущем.

*Задания к лабораторной работе*

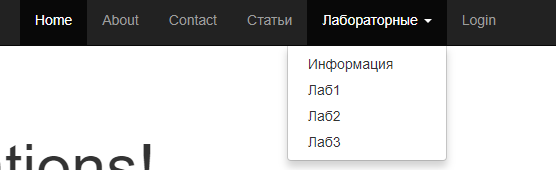
Максимальный балл за лабораторную работу – 3 балла.

Итоговый балл может быть понижен в следующих случаях: 1) задания выполнены некорректно, не в полном объеме, содержат ошибки, реализованы недостаточно оптимально и т.п.; 2) студент не может обосновать и аргументированно ответить на вопросы по заданиям, не владеет теоретическим материалом и т.п.

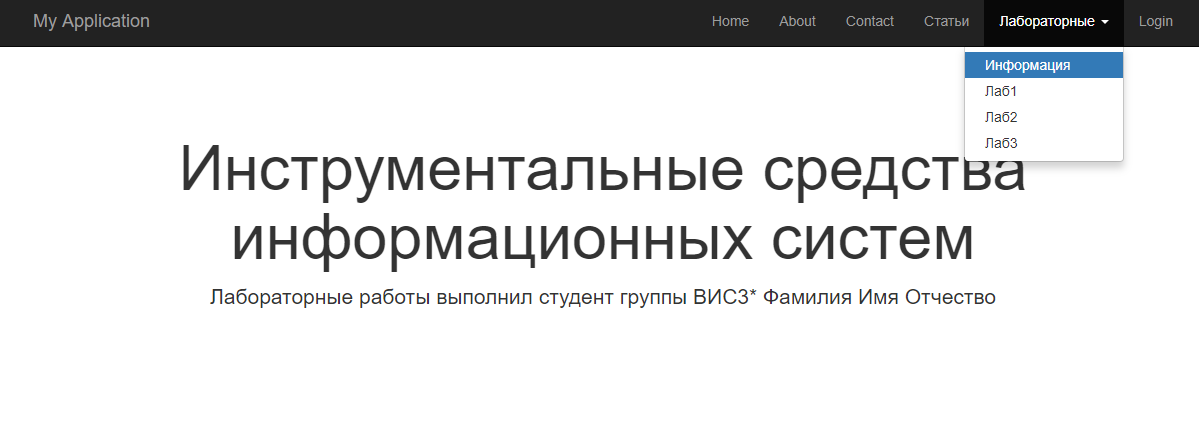
Задание 1

1. Выполните установку Yii. Можете воспользоваться установкой как из Composer, так и из архива. Установите приложение типа basic.
2. Протестируйте, что приложение Yii успешно установлено.
3. Настройте шаблон для дальнейших лабораторных работ.
   1. Добавьте в верхнее меню новый пункт меню «Лабораторные» с подпунктами «Информация». Создайте новый контроллер для работы с лабораторными работами. Добавьте каркас необходимый действий.
   2. Для каждой лабораторной работы создайте каркас представлений с заголовками «Лабораторная работа №1», «Лабораторная работа №2», «Лабораторная работа №3», а также в представлении «Информация» выведите информацию об авторе лабораторных работ (ФИО, группа).
   3. Протестируйте работоспособность ссылок в меню приложения, работу контроллера и вывод необходимых представлений.

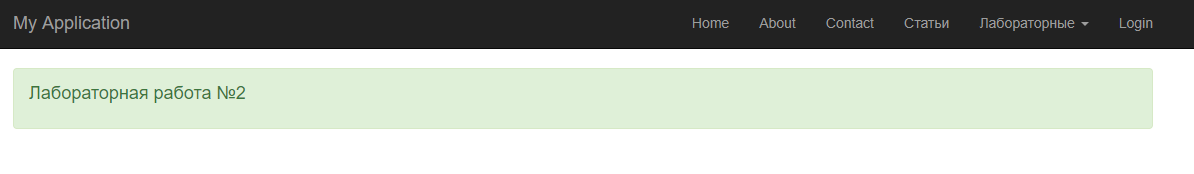
Примерный вид меню:



Страница информации о студенте:



Страница-заготовка лабораторной работы:



Задание 2

1. Разработайте форму, содержащую отзыв о ресторане. Выведите ее на экран пользователя. Необходимые поля:

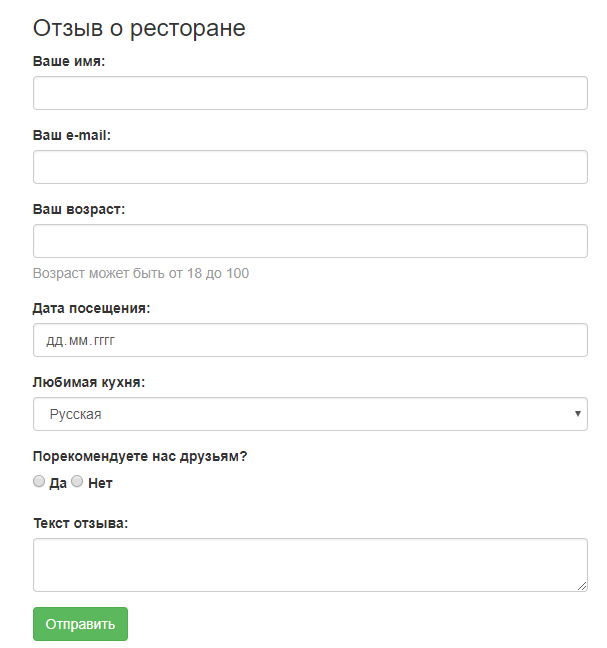
* Имя (текстовое поле)
* Электронная почта (текстовое поле)
* Дата посещения (дата)
* Возраст (число)
* Любимая кухня (select)
* Порекомендуете друзьям? (radio button)
* Текст отзыва (textarea)

1. Добавьте правила валидации:

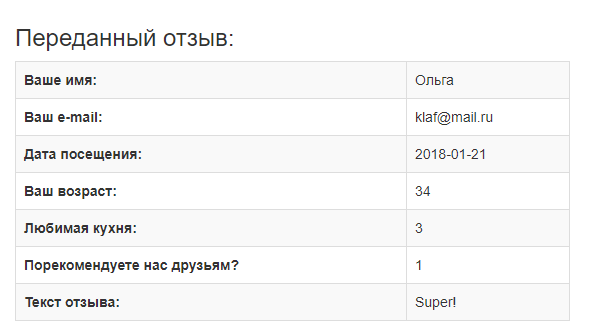
* Имя длиной от 5 до 30 символов
* Корректный адрес электронной почты
* Корректный формат даты посещения
* Возраст от 18 до 100
* Комментарий не содержит лишних пробелов
* Все поля обязательны для заполнения
* Свое сообщение об ошибке

1. Выполните отправку формы и выведите данные в произвольном виде на экран.

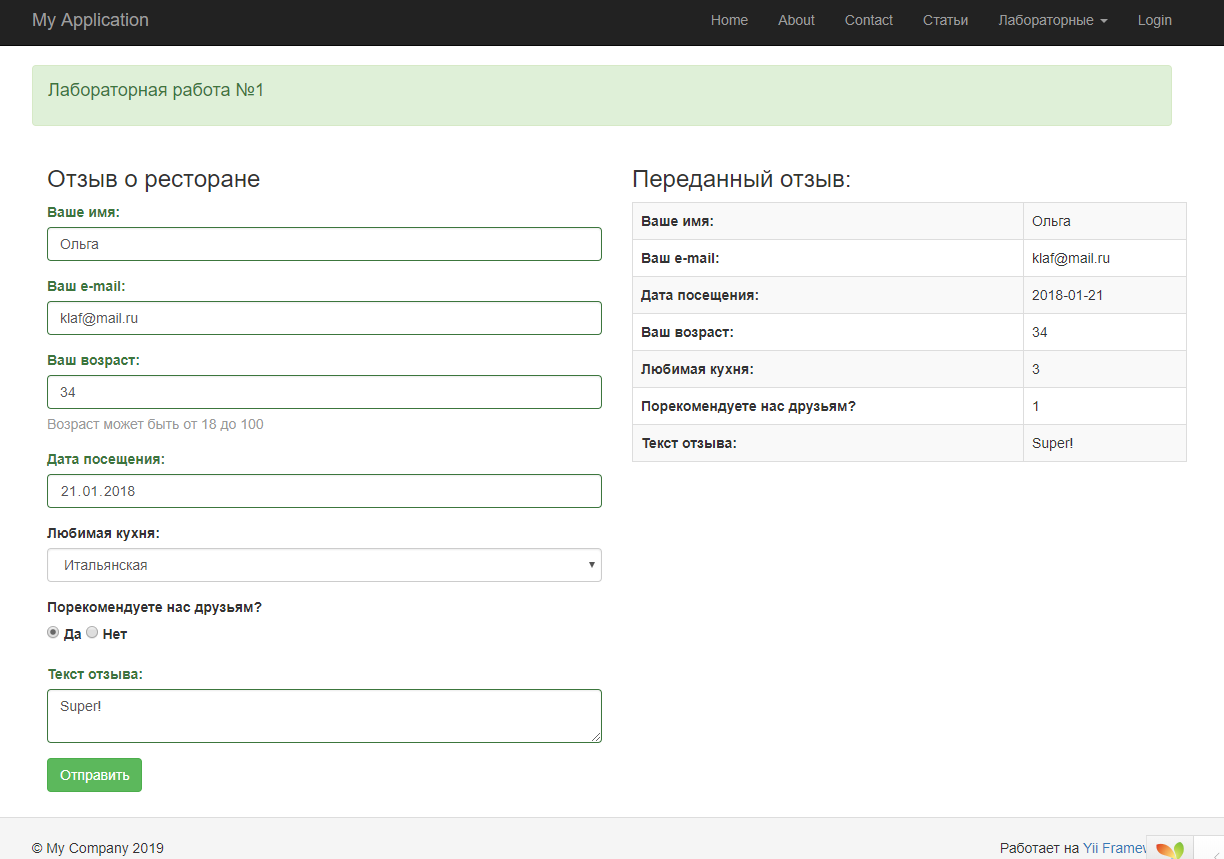
Примерный вид формы:



Примерный вывод:



Конечный вид:



*Контрольные вопросы*

1. Всегда ли необходимо указывать правила валидации?
2. Какой базовый класс необходимо наследовать для создания форм?
3. Какой метод отвечает за подключение вида?
4. Какое пространство имен необходимо указывать при создании контроллера?
5. Какой метод отвечает за открытие тега form?