ФОРМЫ МОДЕЛЕЙ И ВАЛИДАТОРЫ

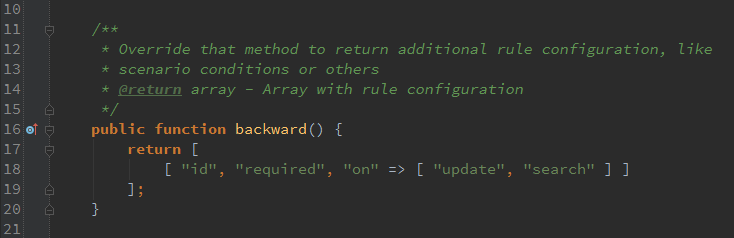
Валидаторы полей формы используется для проверки на соответствие введённых пользователем данных. По умолчанию в системе уже определены некоторые валидаторы, такие как «CRequiredValidator», «CNumericalValidator». По умолчанию все правила валидации «required» заменяются на «LRequiredValidator», который проверяет на наличие не -1 значение и полей типа «Выпадающий список» и «Список с множественным выбором». Каждый созданные валидатор расширяет стандартный класс Yii – «CValidator», документацию по которому можно найти в интернете.

Валидаторы используется в формах модели. Каждая модель формы хранится в папке «forms» и обязана расширять абстрактный класс «LFormModel», суть которого заключается в описании конфигурации и правил валидации для формы. Класс «LFormModel» имеет абстрактный метод «config», который возвращает массив с конфигурацией формы.

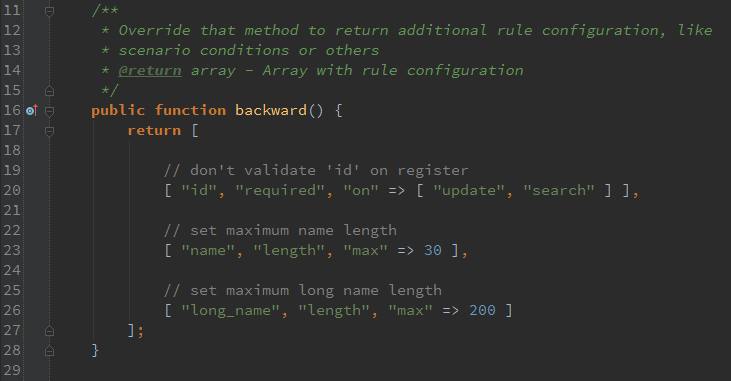
Структура конфигурации формы. В качестве ключа используется название поля в базе данных, а в качестве значения массив со следующими возможными полями (жирным отмечены обязательные):

1. «**label**» - Название поля на русском или другом языке, которое будет использоваться для отображения
2. «**type**» - Тип данных поля (ключ), список всех поле хранится в папке «fields». Ключ для каждого поля можно посмотреть в методе «getKey» этого класса соответственно. Список доступных полей описан в документе с полями.
3. «rules» - список правил для валидации поля, возможные варианты присваивания
   1. Строка, где правила для валидации разделены запятыми
   2. Массив, в котором каждое поле – отдельное правило, полностью соответствующее правилам Yii, например, если поля обязательно для заполнения, то правило будет «required» , если поле обязательно для заполнения только для сценариев поиска или обновления (первичный ключ), то можно указать такое правило: «[“required”, “on” => [“update”, “search]]»
4. “hidden” – является ли поле спрятанным
5. “table” – используется для конфигурации отображаемых данных (только для полей типа “DropDown” и “Multiple”) при автоматическом рендеринге формы, принимает следующие поля:
   1. “**name**” – название таблицы в БД вместе со схемой
   2. “**key**” – первичный ключ, который будет являться “value” <option>
   3. “**value**” – список полей, которые будут подгружены из БД, если указать одно значение, то это значение и будет отображаться, например
   4. “format” – строка или функция, которая форматирует полученные данные, например, при получении нескольких полей из БД (Фамилия, Имя, Отчество). Если полей больше 1, то необходимо указать формат для их отображения. Для обращения к полю БД используется следующая подстановка «%{name}», где «name» - название поля в БД. Пример отображения фамилии, имени и отчества: «%{surname} %{name} %{patronymic}». В случае, например, если нужно отобразить фамилию, имя и отчество пациента в формате Фамилия И.О, то можно использовать функцию для форматирования, которая принимает ассоциативный массив с полями и возвращает готовую строку для отображения

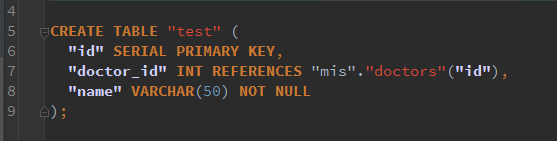
Класс “LFormModel” также включает в себя метод “backward”, который можно использовать для обратной совместимости с “Yii::CFormModel”, в котором уже можно возвращать конфигурацию в привычном формате. Также можно использовать, если нельзя вернуть конфигурацию через “config”, например, для валидации одного и тоже поля для разных сценариев использования, то можно указать отдельную конфигурацию для проверки каждого сценария, например, обязательность заполнения идентификатора записи при поиске или обновлении.



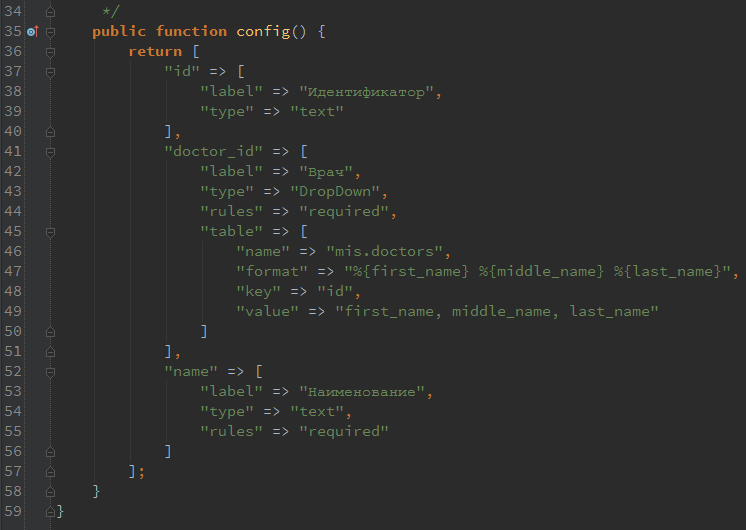
Также при создании модели формы необходимо указывать максимальный и минимальный предел для текстовых полей, которые имеют ограничения на уровне БД



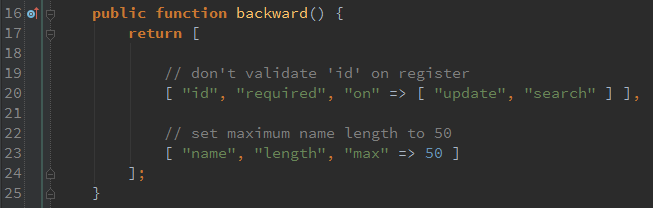
Пример создания тестовой формы, которая в БД имеет следующую структуру.



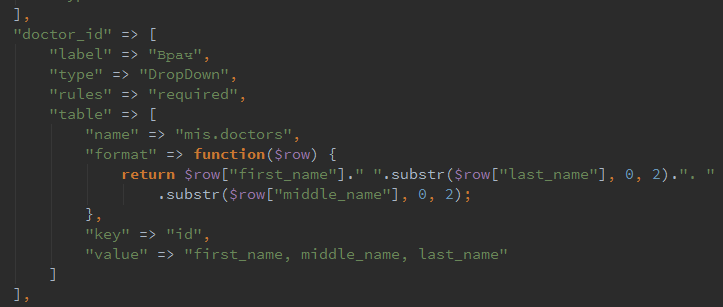
Для такой таблицы модель формы будет иметь следующую конфигурацию



И, соответственно, конфигурация для сценариев использования



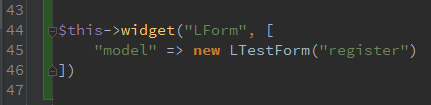
Пример конфигурации формы для отображения Фамилии И.О



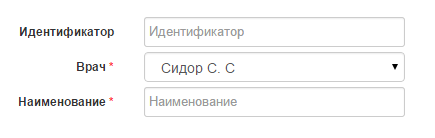
Для отображения формы используется виджет “LForm”, которые принимает следующие поля:

1. “**model**” – сущность формы модели для отображения, элемент также может являться массивом для слияния некоторых форм вместе, в этом случае будет создан «мост» между ними в виде адаптера с объединенными конфигурациями и на сервер будет отправляться объединенная модель, классом по умолчанию для которой являться «LFormModelAdapter»
2. “id” – идентификатор формы, который будет использоваться при отображении
3. “**url**” – ссылка для отправки формы на сохранение, если поля нет, то при отправке будет показано сообщение на уровне клиента с текстом, что отсутствует URL для отправки формы на сервер

Пример создания виджета для отображения формы

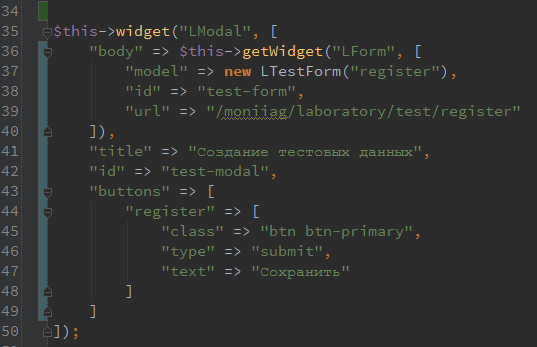


Результат отображения

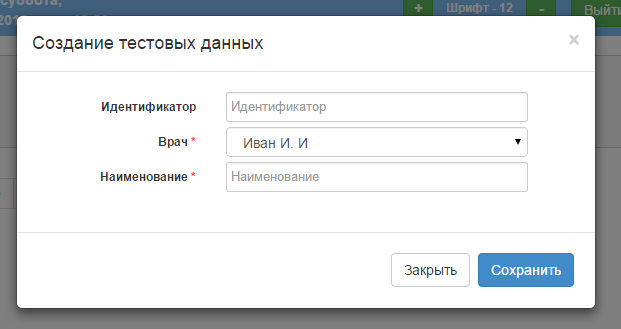


Формы уже умеют автоматически вешать событие для отправки на кнопку с типом данных “submit” при условии, что форма является телом для виджета “LPanel” или “LModal”.

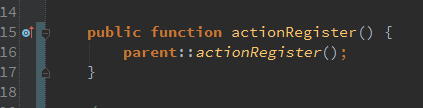
Пример создания модального окна с формой для заполнения и автоматической кнопкой для обработки действия



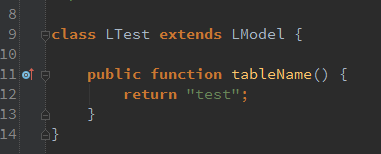
Отображаемый результат (форма по умолчанию не сбрасывается, и сбрасывать ее нужно вручную)



После нажатия на кнопку «Сохранить» на сервер (по указанной ссылке) будет отправлена сериализованная форма в виде строки. Метод для обработки запроса уже имеется, всё что нужно – это только его включить. Для того, чтобы включить метод регистрации (по умолчанию “actionRegister”) нужно его переопределить с вызовом родительского метода, но с указанием модификатора “public”



Для правильного сохранения полученной формы, контроллер имеет абстрактный метод “getModel”, который должен возвращать сущность модели для этой формы с обязательно переопределенным методом “tableName ” для получения имени таблицы БД



Возвращение модели

