Next Framework
Меньше кода, быстрее разработка
Версия 10.1 от 2021 г.

Содержание

1 Особенности Next Framework

Next Framework – это высокоуровневый фреймворк, эффектно решающий ряд актуальных вопросов разработки приложения.

Next Framework вводит дополнительный слой абстракции, разделяя бизнес-логику и функциональную логику. Таким образом, Next Framework продолжает вектор абстрагирования, введенный HMVC, создавая новый слой абстракции.

Бизнес-логика отвечает за поведение приложения с точки зрения пользователя. Правила бизнес-логики отвечают на вопросы “что сделать”.

Функциональная логика отвечает за реализацию бизнес-логики. Императивы функциональной логики отвечают на вопросы “как сделать”.

Новизна Next Framework заключается в том, что он вводит принципиальную дифференциацию между бизнес-логикой и функциональной логикой.

1.1 Преимущества Next Framework

1. Значительное **ускорение** разработки, доработки и сопровождения, до десятков раз.
2. Принципиальное увеличение уровня безопасности разрабатываемых приложений.
3. Повышение производительности приложений, разработка HighLoad-систем.
4. Упрощение процесса разработки за счёт сильной стандартизации структуры и внутренних протоколов приложения.
5. Next Framework полностью упраздняет необходимость проектирования базы данных и ряд других затратных процессов разработки.

Next Framework никак не ограничивает функциональность, скорость, сферу применения и любые другие свойства приложений.

1.2 Действия

Next Framework вводит понятие **действий** (action). Бизнес-логика приложения буквально состоит из **действий**, которые совершает приложение. Декларации действий хранится в конфигурационных файлах. По умолчанию в поддиректории application/config/appconfig.

Каждое действие привязано к одной сущности, или не привязано ни к одной из сущностей. Если действие является **сложным**, и его выполнение затрагивает несколько сущностей, это простое действие разбивается (**декомпозируется**) на несколько простых – атомарных действий, каждое из которых привязано не более, чем к одной сущности.

1.3 Контекст

Для декомпозиции действий вводится понятие **контекста**. При объединении нескольких действий в цепочки, одно из них всегда вызывается агентом пользователя (user agent) напрямую, а другие – выполняются в контексте первого действия.

Контекстные действия нельзя вызвать напрямую агентом пользователя (user agent). Для того, чтобы указать, что данное действие является исключительно контекстным (и таким образом в том числе реализовать требуемый протокол безопасности), нужно добавить в декларацию действия параметр *context* со значением *TRUE*.

1.4 Обработчики

В каждом действии указывается обработчик. Это метод модели HMVC, который указывается в конфигурации действия, который запускается при выполнении данного действия. При запуске обработчика ему передаётся конфигурация действия. Обработчик может получать любые другие данные через модель значений (Value), а также обмениваться информацией с базой данных через слой абстракции базы данных (DAL – database abstraction layer) и кеш.

Модель указывается в параметре конфигурации действия *model*.

Обработчик (метод модели) указывается в параметре конфигурации действия *method*.

1.5 Виртуальный контроллер

Next Framework расширяет модель HMVC, вводя понятие виртуального контроллера.

Технически, Next Framework содержит всего два контроллера – Virtual и Test. Контроллер Test используется для тестирования приложения при его разработке, а также для развертывания приложения.

Контроллер Virtual делает вид, что он является множеством HMVC-контроллеров. Виртуальные контроллеры Next Framework являются результатом перемножения матриц конфигурации действий и обработчиков.

$$[Бизнес-логика]=[Конфигурация\,действий]*[Обработчики]$$

Таким образом, бизнес-логика приложения формируется из деклараций действий и их обработчиков.

2 Контроль доступа

Правила контроля доступа в обязательном порядке указываются в параметре *access* каждого действия. Нельзя продекларировать действие без указания правил контроля доступа.

Есть два специальных правила контроля доступа * и none, которые делают действие доступным для всех (удобно для формирования контекстных цепочек) и недоступным ни для кого соответственно.

- * – действие доступно для всех (за исключением ограничений контекста)
- none – действие НЕ доступно

Также используется множество других правил контроля доступа, которые могут расширяться разработчиками произвольно.

Правила контроля доступа – это действия. Как любые другие действия, они декларируются и используют указанные обработчики для выполнения.

2.1 Базовые правила контроля доступа

Next Framework поставляется с набором базовых правил для контроля доступа.

*	Можно всем
none	Нельзя никому
guest	Только для неавторизованных пользователей (к примеру, форма входа)
registered	Только для авторизованных пользователей
root	Только для пользователей, у которых свойство root = 1
admin	Только для пользователей, у которых свойство admin = 1
manager	Только для пользователей, у которых свойство manager = 1
not_manager	Только для пользователей, у которых свойство manager = 0
item_exist_cache	Элемент есть в кеше. Используется для логики Many-to-many
item_unexist_cache	Элемента нет в кеше. Используется для логики Many-to-many
item_exist	Элемент существует. Используется для логики Many-to-many
undeleted	Элемент не удален. Используется для отображения кнопки “Удалить”.
unactive	Элемент удален. Используется для отображения кнопки “Восстановить”.
is_topicality	Свойство topicality установлено в 1
no_topicality	Свойство topicality установлено в 0

owner	В свойстве user_id данного элемента записан идентификатор текущего пользователя. К примеру, для того, чтобы разрешать редактирование только автору записи.
not_owner	В свойстве user_id данного элемента записан НЕ идентификатор текущего пользователя. К примеру, для того, чтобы разрешать отвечать на объявления или заявки пользователям, за исключением владельца объявления или заявки.
manager_undeleted	Если элемент не удален, действие может совершить только пользователь со свойством manager = 1
manager_unactive	Если элемент удален, действие может совершить только пользователь со свойством manager = 1
owner_undeleted	Если элемент не удален, действие может совершить только владелец элемента
owner_unactive	Если элемент удален, действие может совершить только владелец элемента
admin_undeleted	Если элемент не удален, действие может совершить только пользователь со свойством admin = 1
admin_unactive	Если элемент удален, действие может совершить только пользователь со свойством admin = 1
none_undeleted	Никто не может совершить действие с этим элементом, если он не удален
none_unactive	Никто не может совершить действие с этим элементом, если он удален

Разработчик может изучить простую реализацию этих методов контроля доступа для того, чтобы быстро создавать собственные правила контроля доступа.

2.2 Несколько правил контроля доступа

Правила контроля доступа для действия всегда указываются в массиве в параметре *action*. Таким образом, возможно указать несколько правил контроля доступа.

При выполнении проверки права доступа последовательно проверяется каждое правило. Если результатом выполнения действия для проверки очередного правила является FALSE (булева ложь) – проверка прекращается, и интерпретатор не выполняет указанное действие.

2.3 Контроль доступа и транзакции

По очевидным причинам контроль доступа не выполняется внутри транзакции.

Контроль доступа должен быть выполнен перед транзакцией.

Контроль доступа может быть выполнен в контекстной цепочке после транзакции.

2.4 Адаптивный доступ

Адаптивный доступ Next Framework является мощнейшим средством, которое позволяет формировать бизнес-логику приложения без каких-либо ограничений по её размеру или сложности.

Правило контроля доступа может изменяться в зависимости от состояния элемента.

Примеры реализации таких правил можно посмотреть в undeleted, unactive, owner_undeleted, admin_undeleted, admin_unactive и другие. Декларация действия такого правила – это всего несколько строк, потому разработчик может легко повторить их, действуя по аналогии.

2.5 Создание новых правил контроля доступа

Для большинства случаев используется модель Access. Исходный код крайне минималистичен, поэтому можно изучить его и создавать новые правила контроля доступа, даже самые сложные, просто действуя по аналогии.

Свойство *properties* содержит условия проверки. Доступны указания методов сравнения.

= (не указывается по умолчанию), !=, <, >, <=, >=. Сравнивать можно любые свойства любых объектов – исходные данные получаются из модели Value.

Свойство *get_item* (получить элемент), установленное в true, приведет к попытке получения элемента из базы данных. При этом **элемент будет закеширован**, и следующий обработчик скорее всего получит его из кеша (если явно не указано обратное).

Свойство *mode* (режим) используется для изменения правила доступа (адаптивного контроля доступа) при выполнении указанных условий.

Свойство *reply* (ответ) при установке в false используется для глобальной инверсии правила контроля доступа.

3 Базовые обработчики

3.1 Высокоуровневые

1. Access коллекция методов для авторизации и проверки прав доступа
2. Create создание новых записей в БД, с формой и без неё
3. Edit редактирование записей в БД, через форму и без ней
4. Item вывод одной записи из базы данных
5. Initial загрузка данных по умолчанию при установке приложения
6. Install установка приложения, очистка и создание таблиц в базе данных, очистка директорий для пользовательских файлов
7. Listing вывод списка записей из базы данных
8. Template технически не делает ничего. Содержит низкоуровневую обёртку для вывода
9. Update упаковывает данные в JSON для AJAX-клиентов. Позволяет реализовать сценарии REST и RPC.

3.2 Низкоуровневые

1. Action выполнение действий
2. Appconfig загружает конфигурацию приложения, действия, таблицы и поля
3. Button создаёт кнопки
4. Cookie работает с куками
5. Entity низкоуровневая работа с базой данных через DAL (Database Abstraction Layer)
6. Form низкоуровневая работа с формами и с валидацией
7. Random выделенные из Value методы для работы с хешами и случайными значениями
8. Time методы для работы со временем
9. Value модель для получения данных

3.3 Значения из модели Value

{сущность}.{свойство}	Свойство любой сущности – два ключевых слова, указываемых через точку. Для вывода используется шаблон, указанный в справочнике полей.
data	Значения из конфигурации текущего действия

caller	Значения из вызывающего действия (когда текущее действие является контекстным)
button	Создаёт кнопку
access	Вызов методов из модели Access
password	Проверка пароля
time	Получение времени, см. ниже
sha1	Получает свойство из элемента (указанного в текущем действии), возвращает через функцию SHA1
random	Случайные значения, см. ниже
URI	Получение сегментов URI. После точки должен быть указан номер сегмента
POST	Получение данных из POST. После точки должен быть указан ключ массива POST
context	Выполняет контекстное действие и получает результат его выполения в массиве
content	Выполняет контекстное действие и получает результат его выполения в строке – рендеринг шаблона действия. К примеру, полезно для адресных писем или двухфакторной авторизации
lang	Получает строку из языкового файла. Встроенный механизм локализации поддерживает склонение числительных и другие функции.

3.3.1 Получение времени

Осуществляется через модель Value с ключевым словом time. К примеру, time.today

time.today	Timestamp полуночи текущего дня
time.yesterday	Timestamp полуночи вчерашнего дня
time.tommorow	Timestamp полуночи завтрашнего дня
time.month	Timestamp полуночи первого числа текущего месяца
time.quarter	Timestamp полуночи первого числа первого месяца текущего квартала
time.year	Timestamp полуночи первого января текущего года
time.current_year	Текущий год, число
time.current_month	Текущий месяц, число

time.current_day	Текущая дата, число

3.3.2 Получение случайных значений

Осуществляется через модель Value с ключевым словом random. К примеру, random.hash

random.hex	Вернёт 32 случайных байта
random.hash	Вернёт хеш из строчных и заглавных букв и цифр (как видео Youtube). Исключает букву O, цифру 0 и другие знаки, которые могут быть прочитаны человеком не однозначно.
random.mk5	Вернёт хеш MD5 от случайного числа
random.uuid	Вернёт UUID версии 4

3.4 Параметры

Для программирования с использованием базовых обработчиков возможно использовать следующие параметры.

Параметр	Обработчик	Объяснение
access = [‘*’]	Все	Правила контроля доступа
context = TRUE	Все	Объявляет действие контекстным. Это делает невозможным вызов действия напрямую.
name	Все	Содержит копию имени действия
entity	Все	Содержит копию имени сущности, или default или access для действий без сущностей
model	Все	Модель, в которой находится обработчик действия
method	Все	Метод модели, который является обработчиком действия
out	Все	Массив с данными для пользовательского вывода
template	Все	Путь до отображения (шаблона) для вывода
form	create, edit	Массив с полями и кастомными правилами валидации для форм. Если опустить – добавление и редактирование будет происходить мгновенно, без формы.
set	create, edit	Массив с полями и их значениями для безусловной записи
fields	listing	Массив с полями для формирования вывода

		списка
reverse = FALSE	listing	Обратить порядок вывода списка
pagination, limit	listing	Ограничение списка
sql	listing	SQL-запрос
like	listing	Режим поиска
pagination_source	listing	Откуда брать смещение для ограничения списка
row	listing	Путь для отображения для строки списка
where	item, edit	Массив со сведениями для получения элемента из базы данных
button	Все	Массив с данными для формирования кнопок

4 Кнопки

При отображении кнопки проверяются правила контроля доступа к действию, на которое указывает кнопка. Если действие недоступно – кнопка не будет отображаться.

Действия первого уровня (за исключением контекстных действий) могут вызываться нажатием кнопок в интерфейсе приложения. Возможно выбирать дизайн и принцип работы этих кнопок, указывая соответствующие параметры в конфигурации действия, на которое ведет кнопка.

При создании кнопки выполняется синтетическое (не указанное явно в конфигурации) действие. Параметры этого действия указываются в параметре button конфигурации оригинального действия.

- *template* – указывает на шаблон (отображение) для вывода кнопки в интерфейс
- *model* – модель для синтетического действия, к примеру template для списков или формы добавления или item для элементов, формы редактирования, удаления или восстановления, изменения свойств записи и т.д.
- *method* – метод модели для отображения кнопки
- *where* – опциональное указание, каким образом получить элемент для идентификации
- *out* – опциональное указание перечня данных для отображения кнопки (к примеру, хеш-идентификатор элемента, смещение при выводе списка и т.д.)

5 Отображения

Разработчик может использовать любую структуру отображений и шаблонизаторы, по своему усмотрению.

Next Framework предлагает базовую структуру отображений, которую разработичики Next Framework сочли наиболее удобной.

5.1 Указание отображений

Указание локального (от директории application) пути до отображения происходит в параметре template.

- В декларации поля сущности (в справочнике полей) указывается параметр template.
- В декларации действия указывается параметр template.
- В декларации действия в ветке button указывается параметр template для шаблона кнопки.
- В декларации действия listing указывается параметр row – шаблон для одной строки списка.

5.2 Структура отображений

В поддиректории application/views/browser/

- actions/ - отображения для типичных действий
- buttons/ - отображения для кнопок
- fields/ - отображения для полей (к примеру, преобразование unixtime и числительных в человекочитаемый формат)
- form/ - компоненты форм
- func/ - функциональные компоненты
- output/ - вывод HTML-страниц
- test/ - отображения для тестирования

5.3 Параметры отображений

Разработчик может использовать любые параметры отображений по своему усмотрению.

Разработчики Next Framework приняли следующую систему параметров, как удобную.

- title – то, что выводится в заголовке блока (страницы, логического блока, модального окна и т.п.)

- buttons – блок кнопок
- fields – основной блок информации
- service – дополнительные данные (к примеру, для формирования сложных ссылок M2M)
- context – дополнительный блок информации

В указанных ветках параметра out конфигурации действий содержатся вложенные массивы с данными для вывода (понятными модели Value).

Вывод веток title, buttons, fields находится в зоне действия AJAX-контроллера. Поэтому в нём следует с осторожностью относиться к применению форм – содержимое заполняемых пользователем форм может быть удалено вместе с автоматическим обновлением страницы, если явно не указать обратное.

Вывод ветки context находится в статичной зоне, поэтому в нём допускается применение форм.

5.4 Реактивный интерфейс

Next Framework поставляется в комплекте с реактивным интерфейсом, который имеет следующие возможности и особенности.

- Автоматическое обновление фрагментов страницы, эффект “живого” интерфейса. Полностью автоматическая система AJAX-интерфейса не требует внесения дополнений на серверсайде.
- Фотоаппарат и диктофон
- Загрузка файлов с проверкой типа, размера, преобразованием разрешения и другими функциями
- Полнофункциональный кроссбраузерный WYSIWYG-редактор. Не заимствование, полностью собственная разработка
- AJAX-колбеки и формы

Для активации интерфейса достаточно включить соответствующий тег в HTML страницы и использовать классы CSS.

- nf_editor – активация редактора (кастомизация полей редактора осуществляется через параметр в теге, смотрите примеры)
- nf_photo – фотоаппарат
- nf_dictaphone – диктофон
- nf_update – компонент живого интерфейса

- nf_quiet – AJAX-кнопка
- nf_form – AJAX-формы
- nf_modal – модальные окна Bootstrap
- и так далее.

Реактивный интерфейс написан на ES6 и не требует сборщиков или каких-то других дополнительных компонентов.

6 Примеры типовых решений

6.1 Получение списка записей пользователя

Необходимы два действия.

Первое с обработчиком Item, получает запись пользователя, и запускает второе действие в контексте.

Второе с обработчиком Listing, получает идентификатор пользователя из первого действия, получает из БД список записей и выводит их.

6.2 Многие-ко-многим

Вводится дополнительная сущность для хранения связей многие-ко-многим.

Управление связями как минимум включает в себя следующие действия:

- Создание, если её нет
- Удаление, если она есть и активна
- Восстановление, если она есть и она была удалена

Таким образом, возможно создать следующий набор действий

1. Элемент первой сущности
    1. Кнопка для управления связями
    2. В контексте – список активных связей
        1. В контексте – ссылки на записи второй сущности
2. Элемент второй сущности
    1. Кнопка для управления связями
    2. В контексте – список активных связей
        1. В контексте – ссылки на записи первой сущности
3. Элемент любой сущности
    1. В контексте - список всех доступных (или фильтр) элементов другой сущности
        1. В контексте - в каждой строке мультикнопка (с AJAX интерфейсом)
4. Мультикнопка
    1. (действие в контексте выше получает список имеющихся связей)
    2. Если связь не существует

        1. Показать кнопку для создания связи
    3. Если связь существует
        1. Если связь не активна
            1. Показать кнопку для восстановления связи
        2. Если связь активна
            1. Показать кнопку для удаления связи
5. Создание связи
6. Удаление связи
7. Восстановление ранее удаленной связи

6.3 Кейсы генератора

Генератор содержит примеры множества кейсов из разработки корпоративного и универсального ПО, включая веб-приложения, десктопные и мобильные приложения.

7 Генератор

Разработчик описывает требования к программе в виде формальной спецификации – генератор создаёт нужную программу.

В настоящий момент поддерживается генерация:

- Веб-приложений, в том числе с мобильным и кастомными интерфейсами
- Приложений для Windows и Linux
- Приложений для Android

В разработке

- Приложения для iOS
- Приложения для MacOS

Генератор не входит в комплект поставки Next Framework.

Программы, создаваемые генератором, не имеют никаких функциональных или иных ограничений.

8 Развертывание

8.1 Окружения

Документация по развертыванию окружений поставляется в комплекте с исходным кодом Next Framework.

8.2 Установка фреймворка

Фреймворк поставляется с предустановленным окружением для разработки.

8.3 Установка приложения

До начала установки следует создать пустую базу данных Postgres.

Установка приложения на Next Framework производится из командной строки через контроллер Test из директории document root веб-сервера (www). Используется следующая команда:

php index.php test test_install

Что делает установщик?

1. Очищает базу данных
2. Создаёт таблицы в базе данных
3. Загружает информацию по умолчанию в базу данных
4. Очищает публичную и приватную директории для хранения пользовательских файлов

9 История

Разработка Next Framework начата в 2013 с тестирования гипотезы расширения MVC и HMVC концепциями действий, контекста и адаптивного доступа.

Первый релиз вышел в 2015. С 2015 до 2021 на базе Next Framework создано более 50 программных продуктов, которыми миллионы пользователей пользуются каждый день.

Развитие продолжается по настоящее время.

10 Лицензирование

Next Framework поставляется на следующих условиях.

Приложения на базе Next Framework – в соответствии с лицензией на каждое конкретное приложение. Next Framework не является неотъемлемым компонентом, в рамках не предоставляется гарантия и поддержка.

Пакет для разработчиков Next Framework – даёт право выпускать приложения на базе Next Framework. Предоставляется гарантия и поддержка непосредственому покупателю лицензии. Лицензия неисключительная и НЕ даёт покупателю права дистрибуции. Запрещено распространять Next Framework и его части иначе как в составе пользовательских приложений.

Генератор приложений Next Framework – даёт право выпускать приложения на базе Next Framework. Предоставляется гарантия и поддержка непосредственому покупателю лицензии. Лицензия неисключительная и не даёт покупателю права дистрибуции. Запрещено распространять генератор и его документацию в любом виде.

Не допускается заимствование частей кода или его идей.

Next Framework, его компоненты, документация, концепции и идеи являются собственностью его владельца, и подлежат соответствующей охране. Несанкционированное копирование или дистрибуция подлежат уголовному преследованию.