18. Учебный проект: большие перемены (Часть 1)

Рабочая ветка **module6-task1**

Задача

В этом задании мы добавим в наши компоненты работу с данными и сделаем так, чтобы при изменении информации внутри компонента, изменялись данные, из которых эти компоненты были созданы.

Сперва разберёмся с кодом, который скопился у нас в `BoardController` и отвечает за смену задачи на форму редактирования задачи. Для этого создадим `TaskController`:

1. В нем нужно описать конструктор и метод `render`

2. Конструктор должен принимать `container` — элемент, в который контроллер будет всё отрисовывать.

3. Метод `render` должен принимать данные одной задачи. Также в него должен переехать код, который отвечает за отрисовку задачи, ее замену на форму редактирования и наоборот, а также установка связанных с этим обработчиков событий.

Настроим частичный датабиндинг

Реализуем обработку кликов на кнопках «Favorites», «Archive» у карточки задачи. Обработчики должны изменять данные задачи — добавлять или удалять из избранного и архивировать или разархивировать задачу соответственно, и на основе изменённых данных перерисовывать компонент.

Для этого:

1. В компоненте задачи добавьте методы для установки обработчиков клика для каждой кнопки.

2. В конструкторе `TaskController` добавьте новый аргумент — функцию `onDataChange`. Эта функция должна вызываться в обработчике клика и получать на вход старые и новые данные (задачу и измененную задачу).

Обратите внимание, пока что мы занимаемся обновлением только признаков задачи, поэтому при сохранении у нас по-прежнему форма просто заменяется на карточку задачи, а удаление не работает вовсе. С этим мы разберёмся в следующих заданиях.

3. В `BoardController` опишите метод `_onDataChange` с точно таким же интерфейсом, как функция `onDataChange`. Задача метода — обновить моки и вызывать метод `render` у конкретного экземпляра `TaskController` с обновлёнными данными.

4. Чтобы всё заработало, передайте метод `_onDataChange` в `TaskController` при создании его экземпляра.

Добавим интерактивности

Форма редактирования довольно сложный интерактивный компонент. Но это поведение — не часть бизнес-логики приложения. Это бизнес-логика самого компонента. Поэтому для реализации этой логики введем понятие `SmartComponent` — компонент, который может себя перерисовывать.

1. Создайте абстрактный класс `AbstractSmartComponent`, унаследовав его от `AbstractComponent`, с двумя методами:
    - абстрактный метод `recoveryListeners`, его нужно будет реализовать в наследнике. Его задача — восстанавливать обработчики событий после перерисовки;
    - обычный метод `rerender`, его задачи:
        - удалить старый DOM-элемент компонента;
        - создать новый DOM-элемент;

- поместить новый элемент вместо старого;

- восстановить обработчики событий, вызвав `recoveryListeners`.

2. Унаследуйте компонент формы редактирования от `AbstractSmartComponent` и объявите в компоненте метод `recoveryListeners`, пока пустым.

3. Теперь нужно реализовать перерисовку формы редактирования после взаимодействия с пользователем:

    - показ или скрытие поля ввода даты по клику на «Date», а так же смену «Yes» и «No»;

    - показ или скрытие полей для выбора дней повторения по клику на «Repeat», а так же смену «Yes» и «No»;

    - выбор цвета задачи.

Обратите внимание, что кнопку «Save» необходимо блокировать, если поля показаны, а дата или дни повторения не выбраны.

4. При перерисовке компонента все обработчики событий будут утеряны, поэтому их нужно навесить заново в методе `recoveryListeners`.

Отобразим только одну форму

Мы научились обновлять данные и создавать интерактивные компоненты. Осталось запретить открывать несколько форм редактирования одновременно. Мы реализуем это простым способом: в `BoardController` мы «прикажем» всем экземплярам `TaskController` вернуться в исходное состояние, когда пользователь открывает форму редактирования.

1. Добавьте метод `setDefaultView` в `TaskController` для отображения задачи вместо формы редактирования.

2. Добавьте в конструктор `TaskController` добавьте новый аргумент — функцию `onViewChange` и вызывайте её перед тем, как сменить задачу на форму редактирования.

3. В `BoardController` опишите метод `_onViewChange`, где вызовите у всех экземпляров `TaskController` метод `setDefaultView`.

4. В `BoardController` при создании экземпляров `TaskController` передайте и метод `_onViewChange`.