Взаимодействие с элементами мастера (.NET)

Для работы с элементами мастера используйте объект wizard. В редакторе начните набирать "wizard." — появится список доступных методов.

Пример структуры мастера

```
<wizard title="Wizard title" endAction="ToContext">
  <page name="page1" title="Page title">
    <row>
      <element name="ExampleStringInput" title="Пример текстового поля" type="string" width="3" value="" />
      <element name="ExampleButton" title="Кнопка" type="button" width="2">
        <![CDATA[
          // Скрипт, который выполнится по нажатию на кнопку
        ]]>
      </element>
    </row>
    <row>
      <element name="ExampleTable" title="Тестовая таблица" type="table" width="12" selectType="none">
        <column name="FullName" title="Полное имя" type="string" />
        <column name="Comment" title="Комментарий" type="string" />
      </element>
    </row>
  </page>
</wizard>
```

Быстрые примеры

- Получить значение поля

```
var text = (string)wizard.GetValue("ExampleStringInput");
```

- Установить значение поля

```
wizard.SetValue("ExampleStringInput", "Новый текст");
```

- Получить строки таблицы

```
var rows = wizard.GetTableValues("ExampleTable");
```

- Задать строки таблицы

```
wizard.SetTableValues("ExampleTable", new List<Dictionary<string, object>>
{
  new() { ["FullName"] = "Иванов Иван", ["Comment"] = "OK" },
  new() { ["FullName"] = "Петров Петр", ["Comment"] = "..." },
});
```

Получение данных

- object GetValue(string name) Возвращает текущее значение элемента по имени из WizardData. Примечание: при отсутствии ключа будет исключение KeyNotFoundException.
- string GetTitle(string name) Возвращает «человеко-читаемый» заголовок/подпись значения для ссылочных полей. Фактически берет WizardData[$"{name}__Title"].
- WizardFile GetFile(string name) Возвращает загруженный файл элемента типа "file". Пример:

  ```
  if (wizard.IsFileLoaded("ImportFile"))
  {
    var file = wizard.GetFile("ImportFile");
    var bytes = file.Data;
  }
  ```

- bool IsFileLoaded(string name) Проверяет, есть ли в мастере элемент типа "file" и загружены ли для него данные (размер > 0).

Работа с таблицами

- List<Dictionary<string, object>> GetTableValues(string name) Возвращает копию списка строк таблицы name. Важно: Возвращается новая коллекция. Изменения в ней не попадут в UI, пока вы явно не вызовете SetTableValues.
- List<Dictionary<string, object>> GetSelectedRows(string name) Возвращает копию только выбранных строк таблицы по SelectedIds. Сопоставление идет по внутреннему полю "__Id".
- List<string> GetSelectedIds(string name) Возвращает список выбранных идентификаторов строк таблицы (значения "__Id").
- void SetTableValues(string name, List<Dictionary<string, object>> tableValues) Полностью заменяет данные таблицы. Дополнительно:
  - Для каждой строки гарантирует наличие служебных полей:
    - "__Id": берется из поля "Id" (если есть) или генерируется Guid.
    - "__ParentId": берется из "ParentId" (если есть) или 0.
    - "__HasChildren": true, если у строки есть дочерние строки по "__ParentId".
  - Сбрасывает выбранные строки (SelectedIds = []). Пример:

```
var rows = wizard.GetTableValues("ExampleTable");
rows.Add(new() { ["FullName"] = "Новый", ["Comment"] = "" });
wizard.SetTableValues("ExampleTable", rows);
```

Запись значений и изменение описания элементов

- void SetValue(string name, object value)
- Пример:

```
wizard.SetValue("CustomerLink", 123); // для link также обновит CustomerLink__Title
```

- void SetFilter(string name, string filter) Устанавливает фильтр для элемента типа "link". Пример:

```
wizard.SetFilter("CustomerLink", "IsActive = 1");
```

- void SetLinkedTable(string name, string linkedTable) Для элемента типа "table" задает связанное имя таблицы (AdditionalInfo.LinkedTable) и помечает описание как обновленное.
- void RedrawWidget(string name) Перерисовывает виджет элемента. Пример:

```
wizard.RedrawWidget("ChartWidget");
```

Управление видимостью и доступностью

- void HideElement(string name) Скрывает элемент (IsInvisible = true).
- void HideElements(List<string> names) Скрывает сразу несколько элементов.
- void ShowElement(string name) Показывает элемент (IsInvisible = false).
- void ShowElements(List<string> names) Показывает сразу несколько элементов.
- void SetReadonly(string name, bool value) Переключает режим «только чтение» для элемента (IsReadonly).

Переходы между мастерами

- void OpenNextWizard(string guid) Планирует переход в другой мастер по указанному GUID (NextWizardGuid). Может использоваться, например, по завершении текущего мастера.

Вывод в «веб-консоль»

- void WriteToWebConsole<T>(T message) Пишет текстовое сообщение в WebConsoleMessages (message.ToString()). Удобно для отладки и вывода промежуточных данных.

Валидация и уведомления

- void AddValidationMessage(string elementName, string message) Добавляет сообщение валидации для конкретного элемента текущей страницы. Если есть хоть одно сообщение, PageValidation.Valid = false. Пример:

```
if (string.IsNullOrWhiteSpace((string)wizard.GetValue("ExampleStringInput")))
    wizard.AddValidationMessage("ExampleStringInput", "Поле обязательно для заполнения");
```

- void AddToast(string type, string title, string message) Добавляет всплывающее уведомление (toast). Тип может `default` или `error`. Пример:

```
wizard.AddToast("default", "Готово", "Операция выполнена");
```

Потоковый вывод (SignalR)

- void OpenSocketDialog() Инициирует открытие диалога/панели потокового вывода на клиенте. Устанавливает служебный флаг "__StartSocketStreaming" и помечает UpdatedData.
- void StartSocketStreaming(string title, Func<Task> action) Запускает фоновую задачу (Task.Run), подготавливает клиент к приему сообщений:
  - Устанавливает "__StartSocketStreaming" = true и "__StartSocketStreaming__Title" = title.
  - Выполняет action в фоне.
  - При исключении логирует ошибку и отправляет клиенту служебные сообщения: "Raised exception" и "**END**". Рекомендация: внутри action отправляйте сообщения через SendToClientSocket и самостоятельно завершайте поток отправкой "**END**" по окончании. Пример:

```
wizard.StartSocketStreaming("Импорт данных", async () =>
{
  await wizard.SendToClientSocket("Старт");
  // ... ваша логика ...
  await wizard.SendToClientSocket("__END__");
});
```

- Task SendToClientSocket(string message) Отправляет строковое сообщение текущему пользователю по каналу SignalR ("ReceiveWizardMessage"). Примечание: используйте "**END**" как маркер завершения потока вывода, если это ожидается на клиенте.

Дополнительные замечания

- Большинство методов, меняющих данные/описание, автоматически помечают изменения для фронтенда (UpdatedData, UpdatedDescription, UpdatedTables), поэтому обновления UI происходят без дополнительных действий.
- GetValue/GetTitle/GetFile предполагают, что соответствующие ключи уже инициализированы в WizardData. Иначе возможны исключения. Безопасный паттерн — сначала проверять наличие элемента и/или Инициализировать значения на этапе построения мастера.
- Табличные данные используют служебные поля "__Id","__ParentId" и "__HasChildren" для иерархий и выбора строк. Не удаляйте их из коллекции вручную; используйте SetTableValues.

Мини-примеры из практики

- Спрятать кнопку и сделать поле только для чтения:

```
wizard.HideElement("ExampleButton");
wizard.SetReadonly("ExampleStringInput", true);
```

- Отфильтровать link-справочник и проставить значение:

```
wizard.SetFilter("CustomerLink", "[IsActive] = 1");
wizard.SetValue("CustomerLink", 42);
var title = wizard.GetTitle("CustomerLink"); // Читаемый заголовок выбранного значения
```

- Работа с выбранными строками таблицы:

```
var selectedRows = wizard.GetSelectedRows("ExampleTable");
foreach (var row in selectedRows)
```

```
{
  // обработка выбранных строк
}
```