1.7 Перетаскивание и перенос (Drag and Drop)

Перетаскивание и перенос (Drag and

Drop)

Механика перетаскивания и переноса в Flet достаточно проста - пользователь

начинает перетаскивать управляющий элемент и "бросает" его

Draggable

на . Если и Draggable , и DragTarget находятся в одной

DragTarget

группе, DragTarget вызывает обработчик событий и передает ID

on\_accept

управляющего элемента в качестве данных события. В этом

Draggable

случае переноса.

Draggable

служит источником "данных" для операции перетаскивания и

Давайте посмотрим на следующий пример. В приведенной ниже программе вы можете перетащить левый управляющий элемент, отображающий "1", на верхнюю часть правого управляющего элемента, отображающего "0", и когда операция перетаскивания завершится, левый элемент будет заменен на "0", а правый элемент станет "1":

import flet as ft

def main(page: ft.Page):

page.title = "Drag and Drop example" # Пример перетаскивания и

переноса

def drag\_accept(e):

# получаем перетаскиваемый (источниковый) элемент управления

по его ID

src = page.get\_control(e.src\_id)

# обновляем текст внутри перетаскиваемого элемента управления

src.content.content.value = "0"

# обновляем текст внутри целевого элемента управления

перетаскивания

e.control.content.content.value = "1"

page.update()

page.add(

ft.Row(

[

ft.Draggable(

group="number",

content=ft.Container(

width=50,

height=50,

bgcolor=ft.colors.CYAN\_200,

border\_radius=5,

content=ft.Text("1", size=20),

alignment=ft.alignment.center,

),

),

ft.Container(width=100),

ft.DragTarget(

group="number",

content=ft.Container(

width=50,

height=50,

bgcolor=ft.colors.PINK\_200,

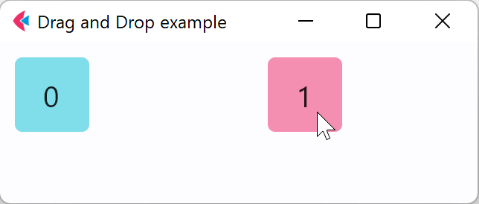
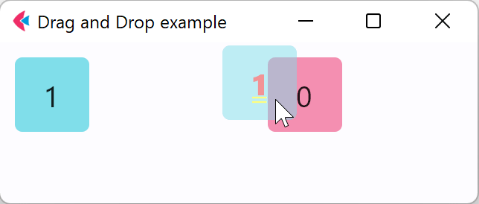
border\_radius=5,

content=ft.Text("0", size=20),

alignment=ft.alignment.center,

),

on\_accept=drag\_accept,



),

]

)

)

ft.app(target=main)

Таким образом, разработчик сам определяет, что происходит с "источником"

( Draggable ) и "назначением" ( DragTarget ) при событии on\_accept .

Существуют дополнительные свойства и обработчики событий, чтобы сделать операцию перетаскивания и переноса еще более интерактивной. Например, у

Draggable

content\_when\_dragging

элемента

есть свойство

, которое позволяет

отобразить другой элемент управления вместо во время перетаскивания.

content

Также есть свойство content\_feedback , которое позволяет отобразить другой элемент управления под указателем. По умолчанию, тот же элемент управления content , но с прозрачностью 50%, отображается под курсором при

перетаскивании.

Давайте модифицируем в нашем примере, чтобы отобразить "пустоту"

Draggable

на месте перетаскиваемого элемента и только "1" под курсором во время перетаскивания:

...

ft.Draggable(

group="number",

content=ft.Container(

width=50,

height=50,

bgcolor=ft.colors.CYAN\_200,

border\_radius=5,

content=ft.Text("1", size=20),

alignment=ft.alignment.center,

),

content\_when\_dragging=ft.Container(

width=50,

height=50,

bgcolor=ft.colors.BLUE\_GREY\_200,

border\_radius=5,

),

content\_feedback=ft.Text("1"),



),

...

У целевого элемента управления есть дополнительные обработчики

DragTarget

событий и on\_leave , которые помогают лучше визуализировать,

on\_will\_accept

когда можно "бросить" что-то на цель. Давайте модифицируем в нашем

DragTarget

примере, чтобы нарисовать рамку вокруг целевого элемента управления, когда он готов принять входящее перетаскивание:

import flet as ft

def main(page: ft.Page):

page.title = "Drag and Drop example 2" # Пример перетаскивания и

переноса 2

def drag\_accept(e):

# получаем перетаскиваемый (источниковый) элемент управления

по его ID

src = page.get\_control(e.src\_id)

# обновляем текст внутри перетаскиваемого элемента управления

src.content.content.value = "0"

# сбрасываем группу источника, чтобы он больше не мог быть

брошен на цель

src.group = ""

# обновляем текст внутри целевого элемента управления

перетаскивания

e.control.content.content.value = "1"

# сбрасываем границу

e.control.content.border = None

page.update()

def drag\_will\_accept(e):

# черная граница, когда разрешено бросить, и красная, когда

нет

e.control.content.border = ft.border.all(

2, ft.colors.BLACK45 if e.data == "true" else

ft.colors.RED

)

e.control.update()

def drag\_leave(e):

e.control.content.border = None

e.control.update()

page.add(

ft.Row(

[

ft.Draggable(

group="number",

content=ft.Container(

width=50,

height=50,

bgcolor=ft.colors.CYAN\_200,

border\_radius=5,

content=ft.Text("1", size=20),

alignment=ft.alignment.center,

),

content\_when\_dragging=ft.Container(

width=50,

height=50,

bgcolor=ft.colors.BLUE\_GREY\_200,

border\_radius=5,

),

content\_feedback=ft.Text("1"),

),

ft.Container(width=100),

ft.DragTarget(

group="number",

content=ft.Container(

width=50,

height=50,



bgcolor=ft.colors.PINK\_200,

border\_radius=5,

content=ft.Text("0", size=20),

alignment=ft.alignment.center,

),

on\_accept=drag\_accept,

on\_will\_accept=drag\_will\_accept,

on\_leave=drag\_leave,

),

]

)

)

ft.app(target=main)



Используя все эти функции и возможности, вы можете создать достаточно сложные и интерактивные сценарии перетаскивания и переноса в своих приложениях.