

# Adatbányászat a Gyakorlatban

## 6. Gyakorlat: Haladó Dash módszerek

Kuknyó Dániel  
Budapesti Gazdasági Egyetem

2024/25  
1.félév

- 1 Dinamikus felhasználói komponensek
- 2 URL manipuláció
- 3 Többlapos alkalmazások
- 4 Kliensoldali callback függvények

## 1 Dinamikus felhasználói komponensek

## 2 URL manipuláció

## 3 Többlapos alkalmazások

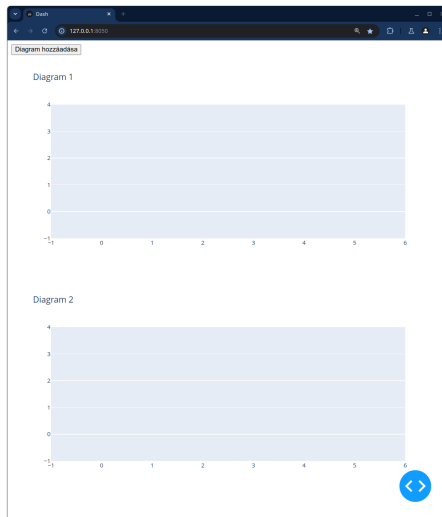
## 4 Kliensoldali callback függvények

# Dinamikus felhasználói komponensek (dyn\_component\_app\_v1.py)

Olyan dinamikus komponensek, amelyek nem állandóak, hanem felhasználói interakcióra kerülnek hozzáadásra az alkalmazáshoz, és el is lehet őket távolítani.

Ehhez tartozóan az első alkalmazás elrendezése:

```
1 app.layout = html.Div([  
2     dbc.Button("Diagram hozzáadása", id='  
3         dyn_component_button'),  
4     html.Div(id='dyn_component_output',  
5         children=[]),  
6 ])
```



# Callback függvény felhasználói komponensek hozzáadására

A callback a `dyn_component_output` Div children komponensét frissíti.

Ha a gombot megnyomták, létrehoz egy új oszlopdiagramot, és a diagram címében megjeleníti a gombnyomások számát. Az új diagramot hozzáadja a children listához.

Visszatér a frissített children listával, amely tartalmazza az új diagramot.

```
1 @app.callback(  
2     Output('dyn_component_output', '  
3         children'),  
4     Input('dyn_component_button', '  
5         n_clicks'),  
6     State('dyn_component_output', '  
7         children')  
8 )  
9 def add_new_chart(n_clicks, children):  
10     if not n_clicks:  
11         return no_update  
12     new_chart = dcc.Graph(figure=px.bar(  
13         title=f"Diagram {n_clicks}")  
14     )  
15     children.append(new_chart)  
16     return children
```

# Az alkalmazás callback gráfja

A callback gráf ebben az esetben egy speciális hurkot mutat. Ez azt jelenti, hogy a függvény egy olyan komponenst térít vissza, amit megkapott paraméterül és módosított a függvénytörzsben.

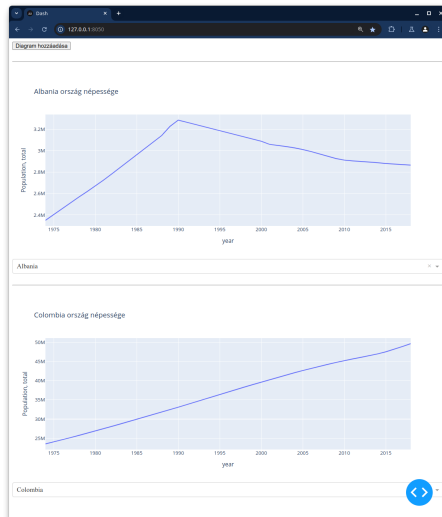
Ez a diagram változatlan marad a hozzáadott komponensek számától függetlenül.



# Mintaillesztő callback függvények

Az alkalmazás következő verziója interaktivitást ad a felhasználói komponenseknek, mintaillesztő callback függvények segítségével.

Minden diagramnak van egy saját legördülő menüje, ahol ki lehet választani egy országnevet, majd kirajzolja a népességet a diagramra.



# Dinamikus komponens elnevezések

Dash rendszerben egy komponens `id` paramétere bármilyen hash-képes (egy függvényen keresztül egyértelműen leképezhető) objektum lehet, így pl. egy szótár is.

Az alkalmazásban a `id` attribútum egy szótár, amely tartalmazza a `type` és `index` kulcsokat. Ez a struktúra lehetővé teszi komponenseket dinamikus azonosítását az alkalmazásban. Például több diagram létrehozása és módosítása esetén mindegyiket egyedi névvel látja el.

```
1 # Új diagram létrehozása
2 new_chart = dcc.Graph(
3     id={'type': 'chart', 'index':
4         n_clicks},
5     figure=px.bar(title=f"Diagram {
6         n_clicks}")
7 )
8 # Legördülő lista opciók létrehozása
9 countries = poverty[poverty['is_country']
10                      ][['Country Name']].drop_duplicates
11                      ().sort_values()
12 # Legördülő lista létrehozása
13 new_dropdown = dcc.Dropdown(
14     id={'type': 'dropdown', 'index':
15         n_clicks},
16     options=[{'label': c, 'value': c} for
17               c in countries],
18     placeholder='Ország kiválasztása'
19 )
```



# Dash MATCH

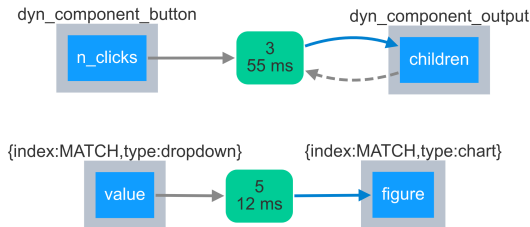
A `dash.dependencies.MATCH` lehetővé teszi, hogy a callback egy adott komponenscsoport egy adott példányára vonatkozzon.

Például ha több dinamikus objektum közül mindegyiknek van egy egyedi `id` paramétere egy szótár formájában, akkor a `MATCH` segítségével meg lehet határozni, hogy a callback csak az adott indexnek megfelelő komponensre reagáljon.

```
1 @app.callback(  
2     Output({'type': 'chart', 'index':  
3         MATCH}, 'figure'),  
4     Input({'type': 'dropdown', 'index':  
5         MATCH}, 'value'),  
6 )  
7 def create_population_chart(country):  
8     if not country:  
9         return no_update  
10    # Adatkészlet szűrése  
11    df = poverty[poverty['Country Name']  
12        == country]  
13    # Diagram létrehozása  
14    fig = px.line(  
15        df,  
16        x='year',  
17        y='Population, total',  
18        title=f'{country} ország népessége'  
19    )  
20    return fig
```

# Az alkalmazás callback gráfja

A felső gráf megegyezik az alkalmazás előző verziójában látottakkal. Az alsó gráfon a MATCH jelzi, hogy csak a megfelelő indexű komponenseket frissítse.



A MATCH mellett az ALL használható minden komponens nevének illesztésére, az ALLSMALLER pedig azokra a komponensekre, amelyek a callback függvény indulása előtt lettek létrehozva.

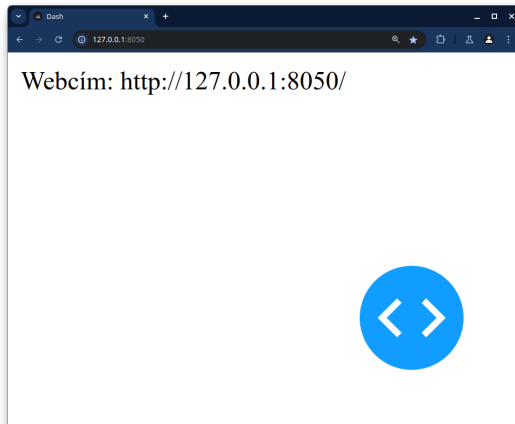
- 1 Dinamikus felhasználói komponensek
- 2 URL manipuláció
- 3 Többlapos alkalmazások
- 4 Kliensoldali callback függvények

## Location és Link komponensek (multi\_app\_v1.py)

Dash keretrendszerben a Location és Link komponensek az útválasztás fontos elemei, melyek lehetővé teszik az URL kezelést és a navigációt.

### Location

A Dash alkalmazásban történő kezelésére szolgál. A Location komponens adatai közvetlenül a böngésző URL-jéből származnak, és az alkalmazásban történő változásokat közvetlenül a böngésző URL-jébe írja vissza.



# Location és Link komponensek (multi\_app\_v1.py)

Dash keretrendszerben a Location és Link komponensek az útválasztás fontos elemei, melyek lehetővé teszik az URL kezelést és a navigációt.

## Link

A Link komponens egy hiperhivatkozást hoz létre a Dash alkalmazásban. Fontosabb attribútumai:

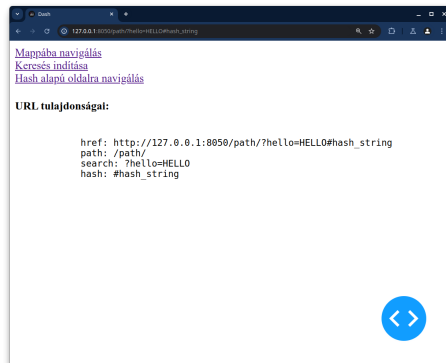
- href: Meghatározza a hivatkozás URL-jét
- target: Megadja, hogy a hivatkozást az aktuális vagy új ablakban kell-e megnyitni
- refresh: Ha igaz, akkor az URL megnyitásakor frissül az oldal



# Location és Link komponensek (multi\_app\_v2.py)

Az alkalmazás a Link komponens felhasználásával megváltoztatja az URL-t és a Location komponens segítségével pedig kivonatol tetszőleges attribútumokat.

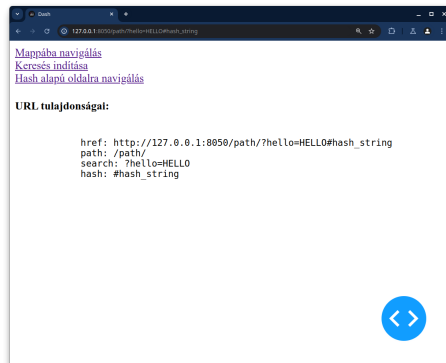
```
1 app.layout = html.Div([
2     dcc.Location(id='location', refresh=False),
3     html.A(href='/path', children='Mappába navig
4        álás'),
5     dcc.Link(href='/path/search?one=1&two=2',
6         children='Keresés indítása'),
7     dcc.Link(href='/path/?hello=HELLO#
8         hash_string', children='Hash alapú
        oldalra navigálás'),
9     html.H4("URL tulajdonságai:"),
10    html.Div(id='location_output')
11 ])
```



# Location és Link komponensek (multi\_app\_v2.py)

A callback függvény bemenetében a Location több paramétere található, és a formázást megőrző `html.Pre` majd kiíratódnak a `Div` tárolóra.

```
1 @app.callback(  
2     Output('location_output', 'children'),  
3     Input('location', 'pathname'),  
4     Input('location', 'search'),  
5     Input('location', 'href'),  
6     Input('location', 'hash'),  
7 )  
8 def show_url(pathname, search, href, hash):  
9     return html.Div([  
10         html.Pre([  
11             f"""href: {href}  
12             path: {pathname}  
13             search: {search}  
14             hash: {hash}"""  
15         ])  
16     ])
```



- 1 Dinamikus felhasználói komponensek
- 2 URL manipuláció
- 3 Többlapos alkalmazások
- 4 Kliensoldali callback függvények



# Többlapos alkalmazások struktúrája

Többlapos alkalmazások esetén az elrendezés csontvázát egy fő elrendezés komponens adja, amelyben egy üres Div található. A Div tartalma attól függően fog változni, hogy a felhasználó milyen oldalra navigál az alkalmazáson belül.

## ❶ Importok (boilerplate):

```
1 import dash
2 from dash import dcc
3 ...
```

## ❷ Alkalmazás példányosítása:

```
1 app = dash.Dash(__name__)
```

## ❸ Alkalmazás elrendezése:

```
1 app.layout = html.Div([
2     ...
3 ])
```

## ❹ Callback függvények:

```
1 @app.callback()
2     ...
3 @app.callback()
4     ...
```

## ❺ Alkalmazás futtatása:

```
1 if __name__ == '__main__':
2     app.run_server(debug=True)
```

# Többlapos alkalmazás: fő elrendezés

Ez a komponens szolgál az alkalmazás csontvázaként.

Tartalmaz egy NavbarSimple navigációs sávot és az országokat tartalmazó legördülő menüt. Az elrendezés még magában foglalja a lapos elrendezést az oldal alján.

A lap törzse tartalmaz egy Location komponenst és egy üres Div objektumot, ami majd az elrendezést fogja tartalmazni.

```
1 main_layout = html.Div([
2     dbc.NavbarSimple([
3         ...
4     ]),
5     dcc.Location(id='location'),
6     html.Div(
7         id='main_content',
8         children=[...],
9     ),
10    dbc.Tabs([
11        ...
12    ]),
13 ])
14
15 ...
16
17 app.layout = main_layout
```

## Többlapos alkalmazás: Indikátorok műszerfala

Ez az az elrendezés komponens, ami az eddigi alkalmazásokban került fejlesztésre. Azzal a különbséggel, hogy egy új változóba kerül elmentésre, és átadódik a `main_content` Div komponensnek, ha a megfelelő feltételek teljesülnek (az URL nem tartalmaz egy országnévet).

```
1 indicators_dashboard = html.Div([  
2     # Minden eddigi komponens  
3 ])
```

# Többlapos alkalmazás: Országok műszerfala

Ez a komponens is egy külön változóba kerül elmentésre, és akkor jelenítődik meg, amikor az URL tartalmaz egy országnevet. Ebben a komponensben az elrendezés (grafikon és táblázat) az URL-ben szereplő országnévtől függően változhat.

```
1 country_dashboard = html.Div([
2     html.H1(id='country_heading'),
3     dcc.Graph(id='country_page_graph'),
4     dcc.Dropdown(id='
5         country_page_indicator_dropdown'),
6     dcc.Dropdown(id='
7         country_page_contry_dropdown'),
8     html.Div(id='country_table')
9 ])
```

## Többlapos alkalmazás: Validációs elrendezés

Ez a komponens egy egyszerű lista egy Dash komponens formájában, ami az előző három elrendezést tartalmazza.

Ennek a jelentősége, hogy megadja az alkalmazás számára, hogy milyen elrendezések érhetőek el a számára. Amikor az alkalmazás csak egy elrendezésből jelenít meg komponenseket és a többiből nem, némely komponensek nem lesznek az alkalmazás része, és néhány callback függvény eltörik.

A validációs elrendezés egy egyszerű módszert ad az alkalmazás egésze számára elérhető elrendezések kezelésére.

```
1 app.validation_layout = html.Div([  
2     main_layout ,  
3     indicators_dashboard ,  
4     country_dashboard ,  
5 ])
```

# Tartalom megjelenítése az URL-től függően

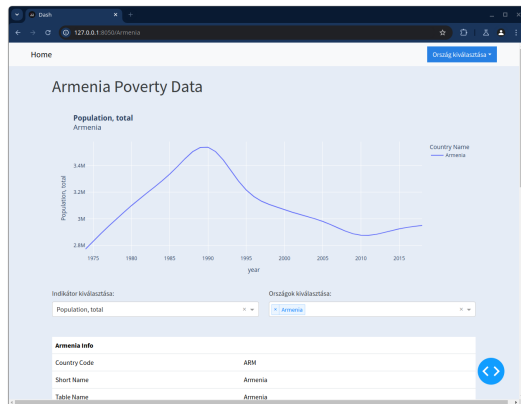
A következő callback függvény ellenőrzi, hogy a Location komponens egyike-e az elérhető országoknak vagy sem. Ha igen, visszatéríti a country\_dashboard elrendezést, egyébként pedig az indicators\_layout elrendezést.

```
1 countries = poverty[poverty['is_country'  
    ']][ 'Country Name'].drop_duplicates  
    ().sort_values().tolist()  
2 ...  
3 @app.callback(  
4     Output('main_content', 'children'),  
5     Input('location', 'pathname')  
6 )  
7 def display_content(pathname):  
8     if unquote(pathname[1:]) in countries:  
9         return country_dashboard  
10 else:ar  
11     return indicators_dashboard
```

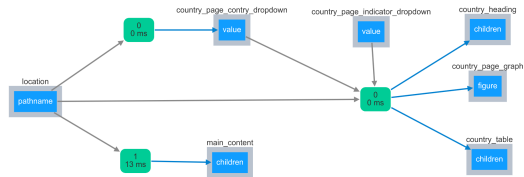
# Alkalmazás többlapos támogatással (app\_v6\_1.py)

Az alkalmazás következő verziójához az összes eddig implementált funkcionalitást be kell építeni a korábban létrehozott struktúrába.

Ennek eredménye egy konstans navigációs sáv, amelyben kiválasztható a megfelelő ország. A sávon kiválasztható az ország, amely átnavigálja a felhasználót egy új oldalra, amihez tartozóan az országhoz tartozó specifikus adatok fognak megjelenni.



# Az alkalmazás callback gráfja



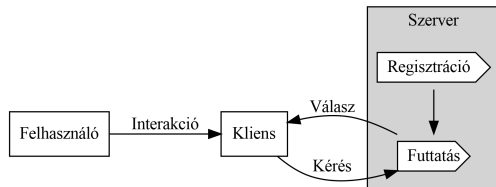


- 1 Dinamikus felhasználói komponensek
- 2 URL manipuláció
- 3 Többlapos alkalmazások
- 4 Kliensoldali callback függvények

# Szerveroldali callback függvények

A szerveroldali függvények regisztrációjának és futtatásának folyamata a következőképpen írható le:

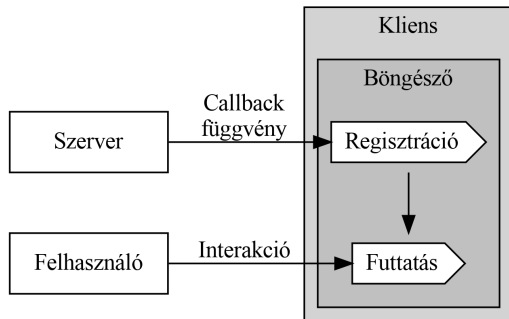
- 1 Függvény definiálása
- 2 Szerver regisztrálja a callback függvényt
- 3 Felhasználói interakció
- 4 Callback futtatása a szerveren
- 5 Eredmény visszaküldése a kliensnek



# Kliensoldali callback függvények

A kliensoldali függvények folyamata:

- 1 Függvény definiálása
- 2 Szerver elküldi a függvényt a kliens oldalra
- 3 Regisztráció a kliens oldalon
- 4 Felhasználói interakció
- 5 Callback futtatása a böngészőben



# Kliensoldali callback függvények

A következő két programrészlet ugyanazt a callback függvényt mutatja be szerver- és kliens oldalon.

```
1 from dash import Input, Output
2
3 @callback(
4     Output('out-component', 'value'),
5     Input('in-component1', 'value'),
6     Input('in-component2', 'value')
7 )
8 def large_params_function(largeValue1,
9                             largeValue2):
10     largeValueOutput = someTransform(
11         largeValue1, largeValue2)
12 return largeValueOutput
```

```
1 from dash import clientside_callback,
   Input, Output
2
3 clientside_callback(
4     """
5     function(largeValue1, largeValue2) {
6         return someTransform(largeValue1,
7                               largeValue2);
8     }
9     """,
10    Output('out-component', 'value'),
11    Input('in-component1', 'value'),
12    Input('in-component2', 'value')
13 )
```