

# Virtual Theme Manager — Integrációs Útmutató

Ez az útmutató leírja, hogyan kell integrálni a futásidejű (on-the-fly) virtuális témaváltást egy meglévő SAPUI5/TypeScript projektbe. A WMS projektbe való beépítés tapasztalatai alapján készült, frissített verzió.

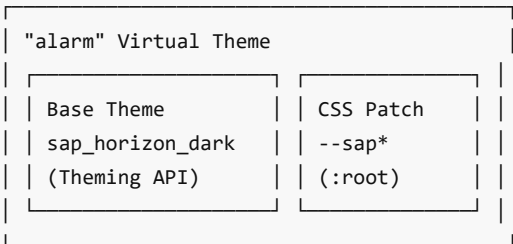
## Tartalomjegyzék

- [Konceptió: Miért "virtuális" témák?](#)
- [Előfeltételek](#)
- [Beépítési útmutató \(setup checklist\)](#)
  - [Szükséges forrásfájl](#)
  - [Mappastruktúra](#)
  - [Módosítandó fájl: Component.ts](#)
  - [Ellenőrző lista](#)
  - [Már integrált projektben \(nincs setup\)](#)
- [Elérhető témák](#)
- [Teljes API](#)
- [Gyors indulás](#)
- [Példák](#)
  - [Feltételes témaváltás hibakezelésnél](#)
  - [Warning téma időkorlátos művelethez](#)
  - [Éjszakai műszak váltás gombbal](#)
  - [Témaválasztó Select építése](#)
  - [Egyedi téma regisztrálása és használata](#)
  - [WebSocket esemény alapján](#)
- [Működés részletesen](#)
- [Használható CSS változók](#)
- [Mire figyelj](#)
- [Fájlok összefoglalása](#)

## Konceptió: Miért "virtuális" témák?

Az UI5 beépített `sap.ui.getCore().applyTheme()` API-ja csak valódi, előre buildelt témákat tud váltani (pl. `sap_horizon` ↔ `sap_horizon_dark`). Ez viszonylag kevés variációt kínál.

A **virtuális téma** megközelítés erre épít, de kibővíti: egy valódi alap témát kombinál **CSS custom property patch-ekkel**. A modern SAP témák (Horizon, Fiori 3) CSS változókat használnak a színekhez ( `--sapBrandColor` , `--sapBackgroundColor` , stb.), és ezek futásidőben felülírhatók a `document.documentElement.style.setProperty()` hívással.



Így egyetlen alap témából tetszőleges számú vizuális variánst hozhatunk létre build lépés nélkül.

A téma **NEM perzisztálódik** – oldal újratöltéskor visszaáll `normal` -ra. Ez szándékos, mert a témaváltás eseményvezérelt figyelmeztetésre szolgál, nem felhasználói preferencia.

## Előfeltételek

- SAPUI5 1.84+ (a VTM a legacy `sap.ui.getCore()` API-t használja a széles kompatibilitás érdekében)
- TypeScript alapú UI5 projekt
- `sap_horizon` és `sap_horizon_dark` témák elérhetők (ui5.yaml-ban `themelib_sap_horizon` )

## Beépítési útmutató (setup checklist)

Ha a VirtualThemeManager-t egy **új kontrollerben / modulban** akarsz használni, semmit nem kell másolnod – minden a helyén van. Ha viszont egy **másik UI5 projektbe** portolod, kövesd az alábbi lépéseket.

### Szükséges forrásfájl

Fájl	Útvonal a POC-ban	Cél
<b>VirtualThemeManager.ts</b>	<code>wms-integration/m/VirtualThemeManager.ts</code>	A témaváltó motor

### Mappastruktúra

```
webapp/  
├─ m/  
│   └─ VirtualThemeManager.ts    ← MÁSOLD IDE  
└─ Component.ts                 ← MÓDOSÍTANDÓ (import + init + accessor)
```

### Módosítandó fájl: `webapp/Component.ts`

3 helyen kell hozzányúlni:

**a) Import hozzáadása** (a fájl elején, a többi import mellé):

```
// webapp/Component.ts - importok szekció  
import VirtualThemeManager from "../m/VirtualThemeManager";
```

**b) Inicializálás az `init()` metódusban** (a router inicializálás előtt):

```
// webapp/Component.ts - init() metódus belseje  
public init(): void {  
    super.init();  
  
    // ... egyéb inicializálás ...  
  
    // ▼▼▼ EZT ADD HOZZÁ ▼▼▼  
    VirtualThemeManager.getInstance().applyDefault();  
    // ▲▲▲ EZT ADD HOZZÁ ▲▲▲
```

```
this.getRouter().initialize();
}
```

**c) Accessor metódus hozzáadása** (a Component osztály publikus metódusai közé):

```
// webapp/Component.ts - publikus metódusok szekció

// ▼▼▼ EZT ADD HOZZÁ ▼▼▼
public getVirtualThemeManager(): VirtualThemeManager {
    return VirtualThemeManager.getInstance();
}
// ▲▲▲ EZT ADD HOZZÁ ▲▲▲
```

Ez biztosítja, hogy bármely kontrollerből egyszerűen elérhető legyen:

```
(this.getOwnerComponent() as Component).getVirtualThemeManager()
```

### Ellenőrző lista

- ☐ VirtualThemeManager.ts bemásolva → webapp/m/VirtualThemeManager.ts
- ☐ Component.ts – import sor hozzáadva
- ☐ Component.ts – applyDefault() hívás az init() -ben
- ☐ Component.ts – getVirtualThemeManager() accessor metódus hozzáadva
- ☐ Teszt: vtm.switchTheme("alarm") → piros képernyő, vtm.switchTheme("normal") → vissza

### Már integrált projektben (nincs setup)

Ha a projekten belül dolgozol, **semmi extrát nem kell csinálnod**. A Component.ts már tartalmazza az importot, az init() hívást és az accessor metódust. Egyből használhatod bármely kontrollerből:

```
// Bármely kontroller onInit()-jében vagy metódusában:
const vtm = (this.getOwnerComponent() as Component).getVirtualThemeManager();
vtm.switchTheme("warning");
```

## Elérhető témák

Kulcs	Base theme	Leírás	Patch	CSS override
normal	sap_horizon	Alapértelmezett világos	–	–
normal_branded	sap_horizon	Zöld branded verzió	5 var	–
warning	sap_horizon	Sárga/amber figyelmeztető	7 var	–
alarm	sap_horizon_dark	Vörös riasztás (kritikus)	24 var	6 szabály
nightshift	sap_horizon_dark	Sötét mód	–	–
nightshift_dimmed	sap_horizon_dark	Extra sötét éjszakai	6 var	–

## Teljes API

```
import VirtualThemeManager from "ntt/wms/m/VirtualThemeManager";

const vtm = VirtualThemeManager.getInstance();

// Téma váltás
vtm.switchTheme("alarm");           // boolean (true=sikeres, false=ismeretlen kulcs)

// Alapértelmezett (normal) téma alkalmazása
vtm.applyDefault();                 // void

// Aktuális téma lekérdezése
vtm.getActiveThemeKey();             // "alarm" | null
vtm.getActiveTheme();                // VirtualThemeDefinition | null

// Összes téma listázása (UI építéshez)
vtm.getThemes();                    // VirtualThemeInfo[]
// → [{ key: "normal", label: "Normal (Horizon Light)" }, ...]

// Egy téma definíciójának lekérdezése
vtm.getThemeDefinition("alarm");    // VirtualThemeDefinition | undefined

// Egyedi téma regisztrálása
VirtualThemeManager.register("my_theme", {
  base: "sap_horizon",
  label: "Saját téma",
  patch: { "--sapBrandColor": "#ff6600" }
});
```

## Gyors indulás

A `VirtualThemeManager` egy singleton, ami runtime CSS variable patch-eléssel és base theme váltással biztosít "virtuális témákat" az alkalmazásban. Nem kell CSS fájlokat szerkeszteni, nem kell oldalt újratölteni – egy metódushívás az egész.

### Hozzáférés kontrollerből

```
import type Component from "../Component";

// Bármely kontrollerben:
const vtm = (this.getOwnerComponent() as Component).getVirtualThemeManager();
```

### Téma váltás

```
vtm.switchTheme("alarm");    // → vörös riasztás téma
vtm.switchTheme("normal");   // → vissza az alapértelmezettre
```

Ennyi. A `switchTheme()` automatikusan:

1. Váltja a UI5 base theme-t ha szükséges (pl. `sap_horizon` → `sap_horizon_dark` )
2. Alkalmazza a CSS variable patch-eket a `document.documentElement` -en
3. Injektál CSS override `<style>` taget ha a téma igényli (pl. `alarm` )
4. Eltávolítja az előző téma összes maradványát

Visszatérési érték: `true` ha sikeres, `false` ha ismeretlen témakulcs.

---

## Példák

### 1. Feltételes témaváltás hibakezelésnél

```
// controller/MyPage.controller.ts

private async _onSave(): Promise<void> {
  try {
    await this._restService.post("/my-endpoint", oPayload);
    MessageToast.show("Mentve!");
  } catch (err: any) {
    // Vizuális figyelmeztetés a teljes képernyőn
    const vtm = (this.getOwnerComponent() as Component).getVirtualThemeManager();
    vtm.switchTheme("alarm");

    MessageBox.error(err.message, {
      onClose: () => {
        vtm.switchTheme("normal");
      }
    });
  }
}
```

### 2. Warning téma időkorlátos művelethez

```
private _onLowStock(iQuantity: number): void {
  const vtm = (this.getOwnerComponent() as Component).getVirtualThemeManager();

  if (iQuantity < 10) {
    vtm.switchTheme("warning");
    MessageToast.show(`Alacsony készlet: ${iQuantity} db`);

    // 5 mp után vissza normalra
    setTimeout(() => vtm.switchTheme("normal"), 5000);
  }
}
```

### 3. Éjszakai műszak váltás gombbal

XML view:

```
<Button
  icon="sap-icon://light-mode"
  press=".onToggleNightShift"
  tooltip="Éjszakai mód"
/>
```

Kontroller:

```
private _isNightShift = false;

public onToggleNightShift(): void {
  const vtm = (this.getOwnerComponent() as Component).getVirtualThemeManager();
  this._isNightShift = !this._isNightShift;
  vtm.switchTheme(this._isNightShift ? "nightshift" : "normal");
}
```

#### 4. Témaválasztó Select építése

XML view:

```
<Label text="Téma"/>
<Select id="themeSelect" change=".onThemeChange"/>
```

Kontroller:

```
public onInit(): void {
  // Select feltöltése az elérhető témákkal
  const vtm = (this.getOwnerComponent() as Component).getVirtualThemeManager();
  const oSelect = this.byId("themeSelect") as Select;

  vtm.getThemes().forEach(t => {
    oSelect.addItem(new sap.ui.core.Item({ key: t.key, text: t.label }));
  });

  // Aktuális téma kiválasztása
  const sActive = vtm.getActiveThemeKey() || "normal";
  oSelect.setSelectedKey(sActive);
}

public onThemeChange(oEvent: Event): void {
  const sKey = (oEvent.getParameter("selectedItem") as any).getKey();
  (this.getOwnerComponent() as Component).getVirtualThemeManager().switchTheme(sKey);
}
```

#### 5. Egyedi téma regisztrálása és használata

```
import VirtualThemeManager from "ntt/wms/m/VirtualThemeManager";
```

```
// Regisztrálás (Component.ts init-ben vagy a saját controller onInit-ben)
VirtualThemeManager.register("customer_xyz", {
  base: "sap_horizon",
  label: "XYZ Ügyfél",
  patch: {
    "--sapBrandColor": "#0d47a1",
    "--sapShell_Background": "#e3f2fd",
    "--sapPageHeader_Background": "#bbdefb",
    "--sapButton_Emphasized_Background": "#1565c0",
    "--sapButton_Emphasized_Hover_Background": "#0d47a1"
  }
});

// Ezután használható mint bármely beépített téma:
VirtualThemeManager.getInstance().switchTheme("customer_xyz");
```

Ha CSS override is kell (pl. `!important` szabályok):

```
VirtualThemeManager.register("customer_xyz_dark", {
  base: "sap_horizon_dark",
  label: "XYZ Ügyfél (Sötét)",
  patch: {
    "--sapBrandColor": "#42a5f5",
    "--sapBackgroundColor": "#0a1929"
  },
  cssOverrides: `
    .sapMPPage { background-color: #0a1929 !important; }
    .sapMBar { background-color: #0d2137 !important; }
  `
});
```

## 6. WebSocket esemény alapján

A `Component.ts` `onWebSocketMessage` metódusában:

```
private onWebSocketMessage(oWebSocketEvent: Event) {
  try {
    let oMessage = JSON.parse(oWebSocketEvent.getParameter('data')) as WebSocketMessage;

    // Témaváltás WebSocket üzenet alapján
    if (oMessage.command === "THEME_CHANGE" && oMessage.data?.themeKey) {
      VirtualThemeManager.getInstance().switchTheme(oMessage.data.themeKey);
    }
  }
  catch (err: any) {
    Log.error("WebSocket Message Error", err.message);
  }
}
```

# Működés részletesen

## Base theme váltás mechanizmusa

Ha az új téma más base theme-t használ (pl. `normal` → `alarm`, azaz `sap_horizon` → `sap_horizon_dark`):

1. A patch "pending" állapotba kerül
2. Meghívódik `sap.ui.getCore().applyTheme("sap_horizon_dark")` – ez aszinkron
3. A `themeChanged` event-re feliratkozva a pending patch automatikusan alkalmazódik
4. Eredmény: a base theme betöltődik, majd rákerülnek a CSS variable felülírások

Ha az új téma ugyanazt a base theme-t használja (pl. `normal` → `warning`):

1. A patch azonnal alkalmazódik, nincs várakozás

## CSS patch alkalmazása

A patch `document.documentElement.style.setProperty()` hívásokkal működik. Az összes SAP CSS variable felülírható.

## CSS override eltávolítás

Témaváltáskor az előző téma **teljes** maradványa törlődik:

- A `document.documentElement` style attribútuma eltávolításra kerül
- A `<style id="vtm-css-overrides">` elem törlődik
- Csak ezután kerül alkalmazásra az új téma

---

# Használható CSS változók

A leggyakrabban használt változók:

Változó	Hatás
<code>--sapBrandColor</code>	Fő brand szín (emphasized gombok, linkek)
<code>--sapBackgroundColor</code>	Oldal háttér
<code>--sapShell_Background</code>	Shell/header háttér
<code>--sapPageHeader_Background</code>	Page header háttér
<code>--sapButton_Emphasized_Background</code>	Kiemelt gombok
<code>--sapField_Background</code>	Input mezők háttere
<code>--sapField_BorderColor</code>	Input mezők kerete
<code>--sapList_Background</code>	Listák/táblázatok háttere
<code>--sapTile_Background</code>	Tile-ok háttere
<code>--sapErrorColor</code>	Hibajelző szín
<code>--sapHighlightColor</code>	Kijelölés, fókusz



--sapContent_LabelColor	Label szöveg szín
--sapButton_Reject_Background	Elutasító gomb háttér

A teljes lista elérhető: [SAP Theming Parameters](#)

## Mire figyelj

1. **Ne inicializáld a class property-ket közvetlenül** – az ui5-tooling-transpile elnyeli őket. Mindig `onInit()` -ben állítsd be a változóidat.
2. **A `switchTheme()` idempotens** – ha már aktív a kért téma, nem csinál semmit ( `return true` ).
3. **Egyszerre egy téma aktív** – nincs "rétegezés", az új téma mindig teljesen lecseréli az előzőt.
4. **CSS override csak ritkán kell** – a legtöbb vizuális változás megoldható a CSS variable patch-csel. Override-ot ( `cssOverrides` ) csak akkor használj, ha egy SAP kontrol figyelmen kívül hagyja a CSS változókat és hard-coded háttérszínt használ.
5. **Nincs perzisztencia** – a téma jelenleg nem mentődik el. Oldal újratöltésnél `normal` -ra resetel. Ha kell, a kontroller `onInit` -jében olvasd ki `localStorage`-ból és hívd meg a `switchTheme()` -t.
6. **A `getThemes()` a regisztrált témákat is tartalmazza** – ha `register()` -rel bővítesz, az automatikusan megjelenik a listában.
7. **`sap.ui.getCore().getConfiguration().getTheme()` a base theme-t adja vissza, NEM a virtuális kulcsot** – használd a `vtm.getActiveThemeKey()` metódust.
8. **Ne használd a modern `import Theming from "sap/ui/core/Theming"` API-t** – a `sap/ui/core/Theming` modul csak SAPUI5 1.118+ felett érhető el, és a `@sapui5/ts-types-esm` csomag sem tartalmazza a típusdefinícióját. A VTM ezért a legacy `sap.ui.getCore()` API-t használja, ami 1.84+-tól működik.

## Fájlok összefoglalása

Fájl	Művelet	Leírás
<code>webapp/m/VirtualThemeManager.ts</code>	<b>ÚJ</b>	Singleton téma manager modul
<code>webapp/Component.ts</code>	<b>MÓDOSÍTÁS</b>	+1 import, +1 sor az init()-ben, +1 getter metódus

A `Settings.view.xml` és `Settings.controller.ts` **NEM módosul** – nincs felhasználói UI a témaváltáshoz (kivéve ha szándékosan építesz egyet, lásd [4. példa](#)).