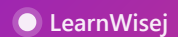


Temi, risorse, localizzazione e modernizzazione della UI

Approfondisci i temi: il cambio chiaro/scuro a runtime, chip di stato riutilizzabili con CSSStyle, la gestione delle risorse e una localizzazione consapevole della cultura.



Obiettivo applicato

Modernizzare da cima a fondo la dashboard operativa di TicketOps: cambiare un tema chiaro/scuro a runtime, costruire un controllo chip di stato riutilizzabile con stile tramite CSSStyle, gestire icone e immagini come risorse e localizzare etichette e formattazione per due culture.

1. Cambiare tema a runtime con Application.Theme

La TicketOps Console include un interruttore nell'intestazione con cui gli operatori passano da un tema chiaro per i turni diurni a un tema scuro per il presidio operativo notturno. Assegna il nome di un tema a `Application.Theme` e Wisej.NET ridisegna ogni finestra aperta; non devi ricostruire tu i controlli.

Tieni i nomi dei temi in un unico punto e salva la scelta dell'operatore, così la dashboard si riapre con lo stesso tema dopo un aggiornamento della sessione.

```
// Intestazione di TicketOps: un solo interruttore governa l'intera console
private void themeSwitch_CheckedChanged(object sender, EventArgs e)
{
    // Nomi dei temi integrati di Wisej.NET distribuiti con l'app
    var theme = themeSwitch.Checked ? "BootstrapDark-4" : "Bootstrap-4";

    Application.Theme = theme; // applica il tema a tutte le finestre
    aperte, dal vivo
    Application.Session["ticketops.theme"] = theme; // ricordalo per sessione
}

// All'avvio, ripristina l'ultimo tema prima di mostrare la dashboard
protected override void OnLoad(EventArgs e)
{
    if (Application.Session["ticketops.theme"] is string saved)
        Application.Theme = saved;

    base.OnLoad(e);
}
```



Regola di produzione

Non spargere mai i nomi dei temi come letterali tra le schermate. Tieni `"Bootstrap-4"` e `"BootstrapDark-4"` in un'unica costante, così un rebranding è una modifica di una riga.

2. Un chip di stato riutilizzabile con stile tramite `CSSStyle`, non con colori scritti nel codice

Ogni riga Work Order mostra uno stato: *Open, In Progress, Blocked, Resolved*. Invece di colorare a mano le etichette in ogni schermata, costruisci un unico UserControl `StatusChip`. Espone una proprietà `Status` e applica un token di classe `CSSStyle`; i colori veri vivono nel tema, così sia il tema chiaro sia quello scuro visualizzano il chip correttamente.

```
// StatusChip: un UserControl riutilizzabile per la dashboard operativa
public partial class StatusChip : UserControl
{
    private WorkOrderStatus _status;

    public WorkOrderStatus Status
    {
        get => _status;
        set
        {
            _status = value;
            labelText.Text = value.ToString();

            // Governa l'aspetto con un token CSSStyle, non con colori inline
            // Il tema definisce .chip-open, .chip-blocked, ecc. per ogni tema
            this.CssClass = "chip chip-" + value.ToString().ToLowerInvariant();
        }
    }
}
```

Poiché il chip fa riferimento a un nome di classe e il tema fornisce la palette, lo stesso controllo appare corretto di giorno e di notte senza un solo condizionale nel tuo codice.

3. Scegliere dove va uno stile

L'errore di modernizzazione più comune è mettere il colore nel livello sbagliato. Usa questa tabella per decidere dove vive ogni regola visiva prima di scriverla.

Tecnica	Ambito	Usala per	Evitala per
Tema / Theme Builder	Intera applicazione	Palette, tipografia, aspetto predefinito dei controlli, colori dei chip	Ritocchi una tantum su un singolo controllo
CSSStyle / CssClass	Un controllo o uno UserControl riutilizzabile	Varianti di componenti condivisi come il chip di stato	Decisioni sui colori a livello di app
BackColor / ForeColor inline	Una singola istanza di controllo	Quasi nulla nelle schermate moderne	Tutto ciò a cui dovrai riapplicare un tema in seguito



Errore comune: colori scritti nel codice

Un chip impostato con `chip.BackColor = Color.Red` resta rosso nel tema scuro e rompe il contrasto. Fai passare l'aspetto dal tema o da una `CssClass`, così entrambi i temi restano corretti.

4. Gestire icone e immagini come risorse

La toolbar della dashboard ha bisogno di un'icona di aggiornamento, un'icona di esportazione e un glifo di stato per ogni chip. Trattali come risorse gestite invece che come percorsi di file sparsi. Wisej.NET può risolvere immagini sensibili al tema, e riutilizzare un unico riferimento a una risorsa mantiene la toolbar coerente tra le schermate.

Tipo di risorsa	Ideale per	Come viene referenziata
Mixin di immagini del tema	Icone che cambiano per tema (glifi chiari o scuri)	Per nome dell'immagine del tema, risolta automaticamente
Risorsa incorporata / di progetto	Loghi e immagini che non cambiano mai con il tema	Tramite l'accessor delle risorse generato
File esterno / URL	Contenuti forniti dall'utente, come gli allegati	Per percorso o URL a runtime

```
// I pulsanti della toolbar attingono a risorse gestite, non a percorsi sparsi
refreshButton.ImageSource = "resource.wx/TicketOps.Icons.refresh.svg";
exportButton.ImageSource = "resource.wx/TicketOps.Icons.export.svg";

// Un unico riferimento riutilizzato ovunque mantiene coerente la toolbar
public static class Glyphs
{
    public const string Refresh = "resource.wx/TicketOps.Icons.refresh.svg";
    public const string Export = "resource.wx/TicketOps.Icons.export.svg";
}
```

5. Localizzare etichette e formattazione per due culture

TicketOps viene usato da team operativi inglesi e italiani. Due cose devono cambiare insieme: il **testo** delle etichette (tramite risorse localizzabili) e la **formattazione** di date e valute (tramite `CultureInfo`).

Localizzarne una senza l'altra produce parole inglesi accanto a numeri formattati all'italiana, e il risultato sembra rotto.

Marca come localizzabili i testi di form e controlli, così i file `.resx` tradotti forniscono ogni cultura. Per i valori, formatta esplicitamente secondo la cultura dell'operatore invece di affidarti al default del server.

```
// Formattazione sensibile alla cultura per il pannello di dettaglio del Work Order
using System.Globalization;

// en-US → 7/3/2026, $1,250.00   it-IT → 03/07/2026, 1.250,00 €
var culture = new CultureInfo(operatorCultureCode); // "en-US" o "it-IT"

createdLabel.Text = workOrder.CreatedOn.ToString("d", culture);
costLabel.Text    = workOrder.LaborCost.ToString("C", culture);
slaLabel.Text     = workOrder.DueBy.ToString("g", culture);

// Imposta la cultura della richiesta così si risolvono anche le risorse localizzabili
Application.CurrentCulture = culture;
Application.CurrentUICulture = culture;
```



Regola di produzione

Imposta insieme `CurrentUICulture` per il testo e `CurrentCulture` per la formattazione, poi ritesta l'etichetta più larga. Le stringhe italiane sono spesso più lunghe di quelle inglesi e possono essere tagliate in un chip o in un pulsante a larghezza fissa.

6. Una checklist di modernizzazione della UI applicata alla dashboard

Passa ogni schermata operativa al vaglio di queste domande prima di definirla moderna. Trasformano le regole astratte della guida della lezione in controlli concreti su schermate TicketOps reali.

- ✓ Ogni colore di stato arriva dal tema o da una `CssClass`, senza alcun `BackColor` inline?
- ✓ L'interruttore chiaro/scuro riapplica il tema all'intera console, chip e icone della toolbar compresi?
- ✓ Tutte le icone della toolbar sono risolte da risorse gestite invece che da percorsi sparsi?
- ✓ Ogni etichetta visibile è localizzabile, e il pannello formatta date e valute tramite `CultureInfo`?
- ✓ L'etichetta tradotta più larga entra ancora senza essere tagliata in entrambi i temi?
- ✓ Un futuro rebranding potrebbe cambiare la palette senza modificare le singole schermate?

Vocabolario chiave

- ✓ **Application.Theme** — assegna il nome di un tema per ridisegnare a runtime tutte le finestre aperte.

- ✓ **CssClass / CSSStyle** — associa un token di classe così che un controllo riutilizzabile erediti i colori del tema.
- ✓ **Chip di stato** — un unico UserControl che esprime lo stato del Work Order tramite una classe, non con colori inline.
- ✓ **Risorsa gestita** — un'icona o un'immagine referenziata per nome, così resta coerente e sensibile al tema.
- ✓ **Proprietà Localizable** — testo del controllo preso da risorse `.resx` specifiche per cultura.
- ✓ **CultureInfo** — governa la formattazione di date e valute sensibile alla cultura per ogni team operativo.