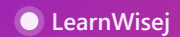


Wiederverwendbare Navigation und Berechtigungen

Die vier Ansichten, die Sie bauen, eine Shell, die die Navigation verantwortet, und ein erster Vorgeschmack auf Berechtigungen.



Lernziel

Lernen Sie das Ziel kennen, das Sie im Lab bauen: eine wiederverwendbare Shell, deren Inhaltsbereich zwischen vier Ansichten wechselt – Dashboard, Tickets, Customers und Settings –, während Kopfzeile, Navigation und Statusleiste gleich bleiben.

1. Die vier Ansichten

Die äußere Shell ändert sich nie. Jeder Navigations-Button lädt ein anderes **UserControl** in das Inhalts-Panel.

Ansicht	Was sie zeigt
DashboardView	Übersichtskarten (Open Tickets, Customers, System Status) und eine Schnellaktion wie <i>Run Health Check</i> .
TicketsView	Eine Liste mit Support-Tickets und den Aktionen <i>New Ticket</i> und <i>Close Selected</i> .
CustomersView	Eine Kundenliste mit den Aktionen <i>Add Customer</i> und <i>View Selected</i> .
SettingsView	Die aktuelle Rolle und einen Button <i>Save Settings</i> , an dem sich das Verhalten von Berechtigungen zeigt.



Warum UserControls

Wenn jede Seite in einem eigenen UserControl steckt, bleibt wiederkehrende UI geordnet und wiederverwendbar, und die Shell bleibt klein – sie verantwortet nur die Navigation und das Inhalts-Panel.

2. Eine Shell verantwortet die Navigation

Die genauen Namen können variieren, das Prinzip bleibt gleich: Die Shell erzeugt die passende Ansicht, dockt sie füllend in das Inhalts-Panel und aktualisiert Breadcrumb und Status.

```
// Main shell owns the navigation buttons and the content panel.
public partial class MainPage : Page
{
    private string currentRole = "Support Agent";

    public MainPage()
```

```

{
    InitializeComponent();
    NavigateTo("Dashboard");
    ApplyPermissions();
}

private void NavigateTo(string pageName)
{
    pnlContent.Controls.Clear();

    UserControl view;
    switch (pageName)
    {
        case "Tickets": view = new TicketsView(); break;
        case "Customers": view = new CustomersView(); break;
        case "Settings": view = new SettingsView(currentRole); break;
        default: view = new DashboardView(); break;
    }

    view.Dock = DockStyle.Fill;
    pnlContent.Controls.Add(view);
    lblBreadcrumb.Text = "Home / " + pageName;
    lblStatus.Text = DateTime.Now.ToString("hh:mm:ss tt") + " - Opened " + pageName + "
page.";
}

private void ApplyPermissions()
{
    // Beginner version: a Support Agent can view Settings but not save.
    // Later this grows into real authentication and authorization.
}
}

```

3. Settings nur zum Ansehen – die erste Berechtigung

In der Settings-Ansicht werden Berechtigungen sichtbar. Für einen **Support Agent** zeigen Sie die Einstellungen an, deaktivieren aber das Speichern – der Button *Save Settings* ist ausgegraut, daneben steht ein Hinweis wie *"Settings are view-only for this role."* Ein Manager dürfte speichern. So führen Sie Berechtigungen ein, **bevor** es überhaupt ein Anmeldesystem gibt.

4. Checkliste für den Designer

- Verwenden Sie **Dock** oder **Anchor** für Kopfzeile, linke Navigation, Inhalts-Panel und Statusleiste.
- Entfernen Sie bei einem Window/Form1 die Minimieren-Schaltfläche, außer Sie bauen eine Taskleiste bzw. einen Bereich zum Wiederherstellen.
- Bevorzugen Sie eine **Page**, wenn die Shell den Browser wie eine normale Web-App ausfüllen soll.
- Halten Sie die Navigationslogik zentral in der Shell, statt sie in jeder Seite zu duplizieren.
- Verwenden Sie UserControls für wiederkehrende Inhalte – **DashboardView**, **TicketsView**, **CustomersView**, **SettingsView**.



Sie sind bereit für das Lab

Als Nächstes: Sehen Sie sich den Video-Walkthrough zur Navigationsshell an, bestehen Sie den Wissenstest und bauen Sie dann die wiederverwendbare Navigationsshell – Kopfzeile, vier Navigations-Buttons, einen wechselnden Inhaltsbereich, ein Status-Log und Settings nur zum Ansehen für einen Support Agent.