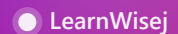


Geräte-APIs über die Hybrid-Bridge

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Kamera, Dateien, Geolocation, Push-Benachrichtigungen und Biometrie aus der App über die Shell – mit Capability-Checks und Berechtigungsabfragen. Was das für ein Team bedeutet, das eine Wisej.NET-App über den Browser-Tab hinaus ausliefert – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Kamera, Dateien, Geolocation, Push-Benachrichtigungen und Biometrie aus der App über die Shell – mit Capability-Checks und Berechtigungsabfragen.

1. Das Problem im Produktivbetrieb

Ein FieldOps-Techniker fotografiert ein korrodiertes Ventil, lässt sich eine Unterschrift auf dem Display geben und fährt zur nächsten Adresse – während der Disponent, der den Auftrag vergeben hat, denselben Arbeitsauftrag in einem Desktop-Browser verfolgt. Jedes Control in FieldOps ist ein C#-Objekt auf Ihrem Server, und der kommt weder an die Kamera noch an den GPS-Chip noch an den Benachrichtigungsbereich heran. Dieses Modul behandelt den Hin- und Rückweg, der diese Lücke schließt – und was der Bildschirm dort tut, wo es diesen Weg nicht gibt.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Geräte-Bridge	Der Nachrichtenkanal, den eine hybride Shell der gehosteten Seite bereitstellt: Ihr Server bittet die Shell, eine Geräteoperation auszuführen, und die Shell antwortet mit einer Nachricht, die die App wieder in ein serverseitiges Event verwandelt.
Host-Objekt	Das Objekt, das eine WebView in die gehostete Seite injiziert, damit JavaScript dort nativen Code aufrufen kann. Serverseitiger C#-Code erreicht es nur über das JavaScript-Interop der App.
Capability-Deskriptor	Die Funktionen, die eine Shell beim Start einer Session meldet – Kamera, Standort, Benachrichtigungen, Biometrie –, zwischengespeichert und abgefragt, bevor irgendeine Geräteaktion angezeigt wird.
Progressive Enhancement	Jeden Bildschirm so entwerfen, dass der Weg im einfachen Browser eine vollständige Funktion ist und der native Weg eine bessere Version davon – nie der einzige, der funktioniert.
Just-in-Time-Berechtigung	Die Plattform genau in dem Moment um Zugriff auf Kamera, Standort oder Benachrichtigungen bitten, in dem der Benutzer den Button drückt, der ihn braucht – direkt nach einem Satz, der erklärt, warum.
Bridge-Ergebnis-Event	Die asynchrone Antwort – Fotodaten, Koordinaten oder eine Ablehnung –, die mit einer späteren Anfrage ankommt als der Klick, der sie angefordert hat, und daher einen eigenen Push braucht.
Speicherung außerhalb des Web-Roots	Aufgenommene Fotos und Unterschriften dort ablegen, wo keine URL auf sie verweist, und sie über einen Handler ausliefern, der prüft, ob diese Session den Arbeitsauftrag sehen darf.

3. Der rote Faden der Lektion

Die Shell fragen und auf die Antwort warten

Der Button Take Photo öffnet keine Kamera. Sein Handler stellt eine Anfrage zusammen – ein Standbild, längste Kante 1600 Pixel, JPEG-Qualität 70 – und reicht sie über zwei Stationen weiter. Die erste stellt WISEJ.NET bereits bereit: Serverseitiger Code führt JavaScript in der gehosteten Seite aus und erhält einen Wert zurück – der Mechanismus hinter `Application.Call` und der Interop-Schicht für Widgets. Die zweite gehört der Shell: Dieses JavaScript ruft das von der WebView injizierte Host-Objekt auf, und die Shell startet die Kamera der Plattform. Die Antwort kommt auf demselben Weg zurück und wird als gewöhnliches .NET-Event mit einem Byte-Array ausgelöst.

Was das von einem normalen Klick unterscheidet, ist die Zeit. Der Techniker braucht vielleicht vierzig Sekunden, um das Motiv auszurichten, der Handler darf also nicht blockieren; er kehrt sofort zurück, und die Anfrage bleibt unter einem von Ihnen erzeugten Token in `Application.Session` offen. Die Antwort kommt mit einer anderen Anfrage an, daher läuft der Code, der die Bytes speichert und die Bildanzeige füllt, außerhalb des ursprünglichen Events, und seine Änderungen an Controls erreichen den Browser nicht

von allein – schließen Sie mit `Application.Update` ab, genau wie bei Arbeit mit `Application.StartTask`.

Abbruch und Fehlschlag sind ebenfalls Ergebnisse, keine Exceptions: Der Benutzer hat abgebrochen, die Anfrage ist in ein Timeout gelaufen, die Shell wurde suspendiert und fortgesetzt. Alles kommt über dasselbe Event an, und das Token ermöglicht es Ihnen, eine Antwort zu verwerfen, auf die niemand mehr wartet.

```
private void btnPhoto_Click(object sender, EventArgs e)
{
    if (!Shell.Current.Has("camera")) { uploadPhoto.Visible = true; return; }
    var token = Shell.Current.RequestPhoto(maxEdge: 1600, quality: 70);
    this.pendingPhotos[token] = this.workOrder.Id;
}

private void OnPhotoCaptured(object sender, ShellResultEventArgs e)
{
    if (!this.pendingPhotos.TryGetValue(e.Token, out var id)) return;
    this.photoStore.Save(id, e.Bytes);
    this.picPhoto.Image = Image.FromStream(new MemoryStream(e.Bytes));
    Application.Update(this);
}
```

Ein Bildschirm, vier Shells

FieldOps wird an einen Browser, eine PWA, ein WebView2-Desktop-Fenster und eine mobile .NET-MAUI-Shell ausgeliefert, und die Seite für Arbeitsaufträge ist in allen vier dieselbe. Was sie anzeigt, ergibt sich aus einem Capability-Check, der einmal pro Session durchgeführt wird: Die App fragt die Seite, ob ein Host-Objekt vorhanden ist und was die Shell als ihre Fähigkeiten meldet, und diese Antwort wandert in die Session, wo jeder Bildschirm sie lesen kann. Was Sie nicht tun dürfen, ist den User-Agent zu parsen – er beschreibt den Browser, nicht, ob dieser Build Ihrer Shell eine Kamera bereitstellt.

Die Matrix hat eine Zeile pro Fähigkeit und eine Spalte pro Shell. Die Kamera ist in der mobilen Shell nativ und im Smartphone-Browser das `Upload`-Control, das der Browser direkt aus der Kamera-App bedient; die Desktop-Shell bietet stattdessen einen Dateidialog. Der Standort ist auf Mobilgeräten nativ und anderswo die Geolocation-API des Browsers, erreicht über JavaScript-Interop. Benachrichtigungen kommen in den beiden Shells vom Betriebssystem und in Browser und PWA per Web-Push. Biometrie gibt es nur dort, wo die Plattform einen Authenticator hat, daher nutzt FieldOps sie, um eine Session wieder zu entsperren – nie als einzigen Zugang.

Take Photo und `Upload` sind daher zwei Darstellungen ein und derselben Aktion, und der Rest des Bildschirms erfährt nie, welche ausgeführt wurde. Eine Fähigkeit, die die Shell nicht meldet, erscheint schlicht nicht: Ein deaktivierter Button ohne Erklärung ist schlechter als ein fehlender, und einer, der beim Drücken eine Exception wirft, ist schlechter als beide.

Berechtigung im Moment der Nutzung

Vor jeder Gerätefunktion stehen zwei Schranken: die Plattformberechtigung, bei der Betriebssystem oder Browser den Benutzer bittet, Kamera, Standort oder Benachrichtigungen zu erlauben, und Ihre eigene

Erklärung, die zuerst kommt. FieldOps zeigt einen Satz – der Standort zeigt die Entfernung zum nächsten Einsatzort an und vermerkt die Ankunftszeit – und löst erst dann die Abfrage der Plattform aus.

Der Zeitpunkt entscheidet über die Antwort. Eine Standortabfrage auf dem Anmeldebildschirm, bevor der Techniker überhaupt einen Arbeitsauftrag gesehen hat, wird mit „Nein“ beantwortet, und auf mehreren Plattformen ist dieses Nein endgültig; dieselbe Abfrage, ausgelöst durch Navigate bei einem Auftrag, dessen Adresse auf dem Bildschirm steht, wird fast immer mit „Ja“ beantwortet. Deshalb fragt FieldOps nach Benachrichtigungen erst nach der ersten Zuweisung der Session und nach der Kamera, wenn Take Photo gedrückt wird.

- Erteilt: fortfahren und die Erlaubnis für diese Session merken.
- Einmal abgelehnt: Aktion und Begründung beibehalten; beim nächsten Auftrag fällt die Antwort vielleicht anders aus.
- Dauerhaft abgelehnt: auf die manuelle Alternative umschalten – ein vorhandenes Foto anhängen, die Adresse eintippen – und zeigen, wo sich die Einstellung befindet.

Das eine Verhalten, das eine dauerhafte Ablehnung garantiert, ist das wiederholte Nachfragen in einer Schleife. Erfassen Sie das Ergebnis jeder Anfrage pro Fähigkeit und pro Session, protokollieren Sie es und behandeln Sie eine Ablehnung als gewöhnlichen Zustand des Bildschirms, nicht als Fehler.

Die Bytes nach Hause bringen

Ein Foto von einem aktuellen Smartphone hat vier bis acht Megabyte, und ein Techniker mit einer ländlichen Verbindung hängt pro Auftrag drei davon an. Verkleinern Sie, bevor die Bytes die Bridge passieren: Die Shell-Anfrage enthält eine begrenzte längste Kante und eine JPEG-Qualität, und der Browser-Fallback zeichnet die gewählte Datei mit derselben Begrenzung in ein Canvas und kodiert sie neu, bevor **Upload** sie sendet. Aus sechs Megabyte werden etwa dreihundert Kilobyte – der Unterschied zwischen einer brauchbaren und einer unbrauchbaren App bei schwachem Signal.

Auf dem Server streamen statt puffern: Kopieren Sie den eingehenden Stream in den Speicher und behalten Sie nur die Kennung im Session-Zustand, denn ein Byte-Array, das an einem Bildschirmobjekt hängt, vervielfacht sich mit jedem gleichzeitig arbeitenden Techniker. Schreiben Sie die Datei unter einem zufälligen Namen außerhalb des Web-Roots und liefern Sie sie über einen Handler aus, der prüft, ob diese Session für diesen Arbeitsauftrag berechtigt ist.

Die Unterschrift durchläuft dieselbe Pipeline mit einer anderen Quelle: Eine Zeichenfläche sammelt die Striche, das zusammengeführte Bild wird als PNG exportiert und gesendet, und der Server speichert es mit der Kennung des Arbeitsauftrags, dem eingetippten Namen des Unterzeichners und einem vom Server erzeugten Zeitstempel – niemals mit einer Uhrzeit oder Identität, die der Client liefert. Der Standort ist die kleinste und sensibelste Nutzlast: Speichern Sie Breitengrad, Längengrad, Genauigkeitsradius und Serverzeit zusammen, denn eine Ortung auf zwei Kilometer genau ist kein Beweis für die Ankunft, und erfassen Sie ihn nur bei einer ausdrücklichen Aktion.



Packaging-Grundsatz

Die App bleibt eine WISEJ.NET-Web-App, die von Ihrem Server ausgeliefert wird; die Shell ist nur das Transportmittel. Sorgen Sie dafür, dass jede Funktion in einem einfachen Browser funktioniert, und ergänzen Sie Gerätefähigkeiten dann als progressive Erweiterungen hinter einem Capability-Check.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ Gerätefunktionen aus serverseitigem Code über die Shell-Bridge aufrufen und die Ergebnisse als Events empfangen.
- ✓ Fähigkeiten zur Laufzeit prüfen, damit derselbe Bildschirm in einem Browser, einer PWA und einer nativen Shell funktioniert.
- ✓ Berechtigungen im richtigen Moment anfordern und eine Ablehnung souverän behandeln.
- ✓ Fotos, Unterschriften und Standortdaten sicher und effizient auf den Server übertragen.

Lab- und Prüfungsleitfaden

Was Sie im Praxis-Lab bauen, das vorgeschlagene Vorgehen und die geforderten Ergebnisse.



Praxis-Lab

Ergänzen Sie FieldOps um Gerätefunktionen: eine Aktion Take Photo, die in der nativen Shell die Gerätekamera nutzt und im Browser auf ein Upload-Control zurückfällt, eine Unterschriftserfassung, die als Bild hochgeladen wird, eine Aktion Navigate, die den Gerätestandort liest und die Karten-App öffnet, eine Push-Benachrichtigung bei der Zuweisung eines Arbeitsauftrags und einen Capability-Check, der jede Aktion pro Shell ein- oder ausblendet.

Ausgangspunkt

Sie öffnen FieldOps im Stand nach den Modulen 1 bis 4: Die Liste der Arbeitsaufträge, die Detailseite und der Ablauf zur Statusaktualisierung laufen alle in einem einfachen Browser, die PWA installiert und aktualisiert sich selbst, die WebView2-Desktop-Shell hostet die App in einem nativen Fenster, und die .NET-MAUI-Shell führt sie unter Android und iOS aus. Die Detailseite hat bereits ein Photos-Panel, ein Signature-Panel und einen Navigate-Button – allesamt funktionslose Platzhalter –, und die Shells stellen eine Bridge bereit, die bisher nichts in der Web-App aufgerufen hat.

Vorgeschlagenes Vorgehen

- ✓ Schreiben Sie einen Capability-Service, der einmal beim Start der Session läuft: Fragen Sie die gehostete Seite, ob das Host-Objekt der Shell vorhanden ist, erfassen Sie die gemeldeten Funktionen – Kamera, Standort, Benachrichtigungen, Biometrie – in der Session und stellen Sie sie jedem Bildschirm über eine einzige Methode bereit.
- ✓ Implementieren Sie Take Photo als eine serverseitige Aktion mit zwei Wegen dahinter: in den Shells eine Bridge-Anfrage, begrenzt auf eine längste Kante von 1600 Pixeln und JPEG-Qualität 70, und im Browser ein **Upload** -Control, das vor dem Senden verkleinert; speichern Sie die zurückgegebenen Bytes außerhalb des Web-Roots und zeigen Sie das Vorschaubild über einen autorisierten Handler an.
- ✓ Fügen Sie das Unterschriftenfeld als Zeichenfläche in die Seite ein, erfassen Sie die Striche, exportieren Sie das zusammengeführte PNG und laden Sie es mit der Kennung des Arbeitsauftrags, dem eingetippten Namen des Unterzeichners und einem vom Server erzeugten Zeitstempel hoch.
- ✓ Verbinden Sie Navigate mit einer Standortanfrage, die zuerst ihre Begründung zeigt, speichern Sie Breitengrad, Längengrad, Genauigkeitsradius und Serverzeit im Besuchsdatensatz, und übergeben Sie dann über die Shell an die Karten-App der Plattform – oder öffnen Sie einen Kartenlink in einem neuen Browser-Tab, wenn es keine Bridge gibt.
- ✓ Lösen Sie eine Benachrichtigung aus, wenn ein Arbeitsauftrag zugewiesen wird – über die Shell, wo das Betriebssystem eine bereitstellt, sonst per Web-Push –, fordern Sie die Benachrichtigungsberechtigung nach der ersten Zuweisung der Session statt bei der Anmeldung an, und protokollieren Sie jedes Berechtigungsergebnis pro Fähigkeit.

Geforderte Ergebnisse

- ✓ Kameraaufnahme mit Upload-Fallback im Browser
- ✓ Unterschriftserfassung und Upload
- ✓ Geolocation und Übergabe an die Karten-App
- ✓ Push-Benachrichtigung bei Zuweisung
- ✓ Capability-Check-Matrix über alle Shells

Wiederholungsfragen

- ✓ Das Kameraergebnis kommt vierzig Sekunden nach dem Klick an, der es angefordert hat. Erklären Sie, mit welcher Anfrage es ankommt, welchen Session-Zustand es vorfinden muss, um zu wissen, was mit den Bytes zu tun ist, und warum die Bildanzeige ohne expliziten Push nicht im Browser erscheint.
- ✓ Ein Techniker hat die Standortberechtigung vor drei Wochen auf seinem eigenen Smartphone dauerhaft verweigert. Beschreiben Sie genau, was ein Druck auf Navigate heute bewirken soll, und nennen Sie die eine Sache, die die App darauf niemals tun darf.
- ✓ Take Photo funktioniert in der mobilen Shell, und der Upload-Fallback funktioniert in Chrome auf dem Desktop. Nennen Sie einen Punkt, der sich für den Benutzer zwischen diesen beiden Wegen wirklich unterscheidet, und begründen Sie, ob dieser Unterschied für einen Außendiensttechniker akzeptabel ist.



Packaging-Grundsatz

Die App bleibt eine Wisej.NET-Web-App, die von Ihrem Server ausgeliefert wird; die Shell ist nur das Transportmittel. Sorgen Sie dafür, dass jede Funktion in einem einfachen Browser funktioniert, und ergänzen Sie Gerätefähigkeiten dann als progressive Erweiterungen hinter einem Capability-Check.