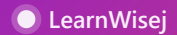


Esercizio di coding con l'IA

Costruisci il progetto IconDesk di questo modulo con ChatGPT o Claude partendo da un prompt già pronto che indirizza il modello alla specifica del modulo, poi rivedi, esegui ed estendi quello che ti restituisce.



Lo stesso modulo, costruito come costruirai davvero le cose.

Tutte le altre sezioni di questo modulo spiegano come funziona "Risorse incorporate e resource.wx". Questa ti chiede di produrlo: dirigendo un assistente con la specifica del modulo stesso, poi verificando che il risultato giri e faccia quello che descrive la traccia del lab. È il flusso di lavoro che userai sul lavoro vero, provato su qualcosa di cui conosci già la risposta giusta.

Che cosa stai chiedendo

Aggiungi all'applicazione IconDesk una cartella Resources che contiene logo.svg, status-ok.svg, status-warning.svg, photo.png e badge.gif, e imposta su ciascuno di essi Build Action Embedded Resource. Costruisci una pagina ResourceLab che li mostra tutti e cinque tramite sorgenti immagine resource.wx, scrivi una di quelle sorgenti antepoendo al file il nome dell'assembly dell'applicazione perché la ricerca non possa essere ambigua e lascia le altre non qualificate. Poi metti un file sostitutivo accanto all'applicazione distribuita per sovrascrivere una risorsa incorporata e cattura il prima e il dopo. Concludi sbagliando di proposito un URL di risorsa e annotando i passi che hanno individuato l'errore.

Il modulo chiede questi deliverable, e il prompt qui sotto li chiede all'assistente esattamente così:

- ✓ Cartella Resources con logo.svg, status-ok.svg, status-warning.svg, photo.png e badge.gif tutti marcati Embedded Resource
- ✓ Pagina ResourceLab che mostra ogni asset incorporato tramite una sorgente immagine resource.wx
- ✓ Una sorgente immagine qualificata con il nome dell'assembly dell'applicazione e le altre lasciate non qualificate
- ✓ File distribuito che sovrascrive una risorsa incorporata, catturato prima e dopo
- ✓ Nota di troubleshooting che elenca i passi con cui è stato individuato un URL di risorsa scritto male

Perché è fatto così

Fino a qui ogni asset di IconDesk è stato qualcosa che l'applicazione ha trovato su disco o ha chiesto al tema. Funziona finché qualcuno non distribuisce il sito dimenticando una cartella: il browser chiede un'icona, il server risponde 404 e l'utente vede un rettangolo vuoto dove dovrebbe esserci il badge di stato. In revisione non se ne accorge nessuno, perché sulla macchina di sviluppo il file c'è ancora.

Questi sono i termini che l'assistente deve azzeccare. Se quello che ti restituisce li usa con leggerezza, non ha capito il modulo:

Termine	Che cosa significa qui
Build Action = Embedded Resource	La proprietà di file di Visual Studio che compila un file dentro il manifest dell'assembly invece di copiarlo accanto all'eseguibile, così l'asset viene spedito dentro la DLL.
handler resource.wx	L'handler HTTP di Wisej.NET che serve al browser le risorse incorporate; quello che assegna è una stringa di sorgente immagine, mai un oggetto immagine lato server.
URL di risorsa non qualificato	resource.wx/company-logo.svg chiede a Wisej.NET di cercare fra gli assembly caricati una risorsa con quel nome: comodo e leggermente ambiguo.
URL di risorsa qualificato con l'assembly	resource.wx/MyCompany.Branding/company-logo.svg indica l'assembly (senza.dll) così che la ricerca non possa finire sul file omonimo di qualcun altro.
Tipi di risorsa consentiti	L'handler documenta esattamente nove estensioni: .css, .js, .html, .jpg, .jpeg, .png, .gif, .svg e .bmp; qualsiasi altra cosa torna indietro come 404.
Override tramite file distribuito	Un file distribuito insieme all'applicazione può essere servito al posto della risorsa incorporata allo stesso percorso, il che permette a un'installazione di ri-brandizzare o correggere al volo la grafica senza ricompilare l'assembly.
Nome della risorsa nel manifest	Il nome che .NET memorizza davvero, derivato dal nome dell'assembly, dal namespace predefinito e dalla cartella in cui si trova il file; tirarlo a indovinare è il motivo abituale per cui un URL di risorsa non restituisce nulla.

Costruiscilo — scegli il tuo assistente

Entrambi i pulsanti aprono una nuova scheda con lo **stesso prompt già pronto**. Il prompt indirizza il modello alla specifica di questo modulo — un unico file Markdown con la lezione, la traccia del lab e i deliverable — così lavora sul materiale vero invece di tirare a indovinare che cosa sia il Modulo 5. Puoi aprire tu stesso [module-05.md](#) per vedere esattamente che cosa leggerà.



Usa ChatGPT

Apri [chatgpt.com](#) con il prompt di questo modulo già caricato. Funziona con GPT-5 o Codex; qualsiasi piano che possa navigare il web recupererà la specifica.



Usa Claude

Apri [claude.ai](#) con lo stesso prompt già caricato. Chi usa Claude Code può invece incollarlo in un terminale dentro una cartella vuota.



Mandalo così com'è e ottieni un progetto, non un frammento

Il prompt si chiude chiedendo l'intera soluzione in un unico `.zip` con i passi per compilarla ed eseguirla. Scaricalo, apri la soluzione in Visual Studio, premi F5 e IconDesk si apre su `localhost`. Ti servono l'SDK .NET e il pacchetto NuGet `Wisej-4`; tutto il resto il prompt chiede all'assistente di scriverlo da sé.

Sto seguendo il Modulo 5 - "Risorse incorporate e resource.wx" - del corso LearnWisej "Icane e immagini in Wisej.NET", e voglio costruire con te il progetto del modulo. Rispondimi in italiano.

Comincia leggendo la specifica del modulo. È un unico file Markdown (in inglese) che contiene l'intera lezione, la traccia del lab e i deliverable:

<https://learnwisej.com/assets/courses/icons-and-images/specs/module-05.md>

Poi costruisci un progetto Wisej.NET 4 completo che io possa aprire in Visual Studio:

1. Progetto "IconDesk", C#, net8.0 o successivo, pacchetto NuGet Wisej-4 4.1.x, host Kestrel che chiama `app.UseWisej()`.
2. Costruisci quello che chiede il lab della specifica: Aggiungi all'applicazione IconDesk una cartella Resources che contiene `logo.svg`, `status-ok.svg`, `status-warning.svg`, `photo.png` e `badge.gif`, e imposta su ciascuno di essi Build Action Embedded Resource. Costruisci una pagina ResourceLab che li mostra tutti e cinque tramite sorgenti immagine `resource.wx`, scrivi una di quelle sorgenti antepoendo al file il nome dell'assembly dell'applicazione perché la ricerca non possa essere ambigua e lascia le altre non qualificate. Poi metti un file sostitutivo accanto all'applicazione distribuita per sovrascrivere una risorsa incorporata e cattura il prima e il dopo. Concludi sbagliando di proposito un URL di risorsa e annotando i passi che hanno individuato l'errore.
3. Produci questi deliverable e dimmi dove si trova ciascuno nella soluzione:
 - Cartella Resources con `logo.svg`, `status-ok.svg`, `status-warning.svg`, `photo.png` e `badge.gif` tutti marcati Embedded Resource
 - Pagina ResourceLab che mostra ogni asset incorporato tramite una sorgente immagine `resource.wx`
 - Una sorgente immagine qualificata con il nome dell'assembly dell'applicazione e le altre lasciate non qualificate
 - File distribuito che sovrascrive una risorsa incorporata, catturato prima e dopo
 - Nota di troubleshooting che elenca i passi con cui è stato individuato un URL di risorsa scritto male
4. Segui la specifica ovunque nomini una classe, una proprietà o un evento di Wisej.NET - usa esattamente quello. Dove lascia una scelta aperta, dimmi quale API hai scelto e perché.
5. Tieni l'autorità sul server: stato e validazione vivono in .NET, il browser disegna e riferisce. Non spostare mai una decisione in JavaScript solo perché lì è più comodo.
6. Rendi il risultato dimostrabile: devo poterlo eseguire e vedere il comportamento di cui parla questo modulo, compreso che cosa succede quando qualcosa va storto.

Prima di scrivere codice, riformula con parole tue che cosa costruirai e dove starà ogni pezzo, così posso correggere il progetto finché costa poco.

Quando il codice è pronto, impacchetta l'intera soluzione in un unico file .zip scaricabile e dammi i passi esatti per compilarla ed eseguirla.

Infine rispondi a queste domande della specifica, usando come prova la tua implementazione:

- Due assembly referenziati incorporano entrambi un file chiamato `logo.svg`. A che cosa si risolve una sorgente non qualificata, e che cosa cambieresti perché la risposta smetta di dipendere dall'ordine di caricamento?
- Il tuo override funziona sul server di test ma non dopo il rilascio successivo. Quali parti del processo di deploy potrebbero averlo rimosso, e dove avrebbe dovuto essere registrato?
- Un collega incorpora `icon.ico` e segnala che `resource.wx` è rotto. Che cosa sta succedendo

davvero, e qual è la modifica più piccola che lo risolve?

Chiedimi qualsiasi cosa non ti sia chiara prima di cominciare.

Poi rivedi quello che ti è tornato indietro

Codice che compila non è ancora codice giusto. Le domande interessanti sono domande di progetto: è questo il controllo Wisej.NET che il compito richiede davvero, segue il ciclo di vita che il framework ti offre, lo stato autorevole sta sul server o è stato parcheggiato nel browser perché era comodo? Leggi il risultato da revisore, non da cliente.

Questo modulo ti dà la checklist. Rispondi a ciascuna domanda guardando il codice che hai appena ricevuto:

- ✓ Due assembly referenziati incorporano entrambi un file chiamato logo.svg. A che cosa si risolve una sorgente non qualificata, e che cosa cambieresti perché la risposta smetta di dipendere dall'ordine di caricamento?
- ✓ Il tuo override funziona sul server di test ma non dopo il rilascio successivo. Quali parti del processo di deploy potrebbero averlo rimosso, e dove avrebbe dovuto essere registrato?
- ✓ Un collega incorpora icon.ico e segnala che resource.wx è rotto. Che cosa sta succedendo davvero, e qual è la modifica più piccola che lo risolve?

Tutto ciò che hai dovuto correggere vale la pena di scriverlo. Una breve nota con le convenzioni e le decisioni a cui sei arrivato, incollata nel primo messaggio della sessione successiva, vale più di qualsiasi prompt ingegnoso.



Per questo corso non c'è ancora un esempio di riferimento — il tuo è il primo

Il [repository degli esempi LearnWisej](#) contiene build già fatte per quasi tutti i corsi, ma non ancora per questo. Significa che non c'è niente con cui confrontarsi e nessuna scorciatoia: la specifica e la tua revisione sono tutta la rete di sicurezza. Tieni quello che costruisci — e se viene bene, ci farebbe piacere vederlo.

Tre prompt per andare oltre

Punta il tuo assistente sul progetto che hai appena costruito e prova questi. Ognuno è un esercizio di revisione ed estensione, non una riscrittura.

- ✓ **Aggiungi il secondo caso.** *"Estendi questo progetto IconDesk con un secondo caso che segue lo stesso schema del primo — un'altra entità, schermata o controllo della stessa forma, collegato allo stesso modo. Estrai in una classe base o in un servizio ciò che è davvero condiviso, ma solo dove se lo merita; due cose simili non chiedono sempre un'astrazione. Mostrami il diff e giustifica ogni cosa che hai estratto."*
- ✓ **Dimostra che funziona davvero.** *"Passa in rassegna questo progetto deliverable per deliverable rispetto alla specifica del modulo su <https://learnwisej.com/assets/courses/icons-and-images/specs/module-05.md> e mostrami la prova che ciascuno è davvero fatto — il percorso nel codice, su che cosa cliccare e che cosa dovrei vedere. Dove un deliverable è soddisfatto solo in parte, dillo chiaramente invece di arrampicarti sugli specchi. Poi aggiungi quello che manca."*

- ✓ **Rivedilo come faresti con una pull request.** *"Accompagnami attraverso questo codice IconDesk come se stessi rivedendo la pull request di un collega e non l'avessi scritto tu. Dove vive lo stato, che cosa succede dall'inizio alla fine su una singola azione dell'utente, che cosa rimanderesti indietro per modifiche, e quali parti non ti sentiresti tranquillo a mettere davanti a un cliente? Sii specifico e cita le righe di cui parli — non riassumermi il progetto."*



Portalo dentro il lab

Il lab pratico che chiude questo modulo chiede gli stessi deliverable che hai appena fatto produrre all'assistente. Usa quello che hai qui come forma della tua risposta e come prova a sostegno — non come la risposta in sé. Il lab viene valutato sul tuo ragionamento, e quella parte deve essere tua.