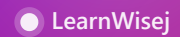


Configuration, Security & Deployment

Configuration & Secrets · Security & Deployment Readiness



Configuration & Secrets

Which configuration files to inspect before release, telling debug from production, and handling license keys and secrets safely.



Lesson goal

Know which configuration files to inspect before release, tell debug, staging and production apart, and handle license keys, connection strings and other secrets safely.

1. Configuration files to inspect

In a Wisej.NET project, configuration can live in several places. You don't need to memorize every advanced setting — but you must know where to look before deployment.

Area	What to check	Why it matters
<code>Program.cs</code> / startup	Which window/page starts, environment-specific startup logic, service setup.	The deployed app must open the correct entry point.
<code>Web.config</code> / hosting config	Debug flag, handlers, server settings, rewrite rules, error behavior.	IIS hosting depends heavily on correct configuration.
<code>Default.json</code> / app JSON	Wisej endpoint, theme, URL, window/page settings, static resource references.	Wrong URLs or themes can make a deployed app fail or look broken.
Themes & static files	Theme folder, CSS, images, icons, JavaScript files.	Resources must be included in the deployment package.
Logging config	Log destination, log level, protected error details.	Production needs troubleshooting without exposing internals.

2. Debug vs staging vs production

Debug settings help while building, but production settings should be safer and more controlled. Before release, review the environment and make sure debug behavior isn't accidentally left on.

- **Debug** — detailed errors, local testing, easier troubleshooting; not safe for public users.
- **Staging** — a production-like test area used before release.
- **Production** — safe error messages, protected secrets, stable configuration, logging enabled.



Safe release rule

Users should see a useful message; developers should see technical details in secure server-side logs. Never show stack traces, file paths, license keys or connection strings in the UI.

3. Secrets and license key handling

Sensitive values should not be hard-coded into UI code, button events, screenshots or source control — license keys, database connection strings, API keys, SMTP passwords and deployment credentials.

Unsafe practice	Safer practice
Hard-code a license key in a page or button event.	Use secure configuration or environment variables.
Commit connection strings to Git.	Keep secrets outside source control; use deployment secrets.
Display raw exception details to users.	Log details server-side and show a safe message.
Share production credentials in screenshots.	Redact or avoid showing secrets in training materials.



You're ready for the next reading

You can locate the key configuration files, separate debug from production, and keep secrets safe. Next: authentication and authorization, logging, and a deployment checklist.

Security & Deployment Readiness

Authentication vs authorization at a junior level, safe logging, deployment targets, and a required release checklist.



Lesson goal

Understand authentication vs authorization at a junior level, log safely, and build a deployment checklist for IIS, Kestrel or cloud hosting.

1. Authentication & authorization

Authentication answers *who is the user?* **Authorization** answers *what is this user allowed to do?* Module 9 introduces this as a design concept before full identity implementation.

- **Authentication:** sign-in, identity provider, session identity, user name, role claims.
- **Authorization:** role checks, page access, command access, hidden or disabled buttons.
- **Example:** a Support Agent can manage tickets; an Admin can edit deployment settings.
- Never rely *only* on hiding a button — server-side logic should also protect the action.

```
bool canReviewDeployment =
    currentUser.Role == "Admin" || currentUser.Role == "Team Lead";

btnReviewPackage.Enabled = canReviewDeployment;
lblPermission.Text = canReviewDeployment
    ? "Deployment review available."
    : "Review is restricted.";
```

2. Logging and troubleshooting checklist

Logging should record what happened without exposing secrets. A good troubleshooting log is short, readable and useful after something goes wrong.

- Log the start and finish of important workflows.
- Log validation failures and deployment-review failures.
- Log exceptions server-side with enough detail for a developer.
- Don't log passwords, license keys or private user data.
- Add timestamps so the order of events is clear.

3. Deployment targets

Target	Junior-level checks
IIS	Publish the app folder, review <code>Web.config</code> , check the hosting bundle if needed, confirm the app pool, test the URL.
Kestrel / self-host	Confirm environment variables, ports, reverse proxy, HTTPS and JSON config files.
Cloud	Confirm app settings, secrets, static files, logging and the deployment slot / staging setup.

4. Required release checklist

- `Web.config` reviewed and debug mode set correctly.
- Wisej license key handled securely.
- Default/startup window, theme and URL settings checked.
- Themes and static resources included.
- Logging destination configured.
- Authentication and authorization plan reviewed.
- Sensitive files are not publicly downloadable.
- Release build tested locally before deployment.
- Deployment target requirements checked.



You're ready for the lab

Next: watch the deployment-review walkthrough, pass the knowledge check, then build a release checklist screen — required and optional checks, deployment notes, a troubleshooting log, and a package-status message that changes when the required checks are complete.