



How to Upgrade

 [Release Notes](#)

[Release Notes 5.0](#) 

Upgrading from one Tapestry version to the next is usually quite easy. Backward compatibility is one of Tapestry's [core principles](#). Even so, sometimes a checklist comes in handy to be sure you consider all of the implications of an upgrade.

Before You Upgrade

1. **Check Java version compatibility:** See the compatibility matrix at [Supported Environments and Versions](#).
2. **Check 3rd Party compatibility:** Find out whether your 3rd party modules are compatible with the new version of Tapestry. Although the Tapestry developers try very hard to maintain backward compatibility across versions, sometimes an older version of a 3rd party module (particularly if it uses internal APIs) won't work with a newly-released version of Tapestry, and in that case you may have to wait until that 3rd party module is updated by its developers.
3. **Find and replace all calls to deprecated APIs.** Those are the places most likely to be broken after the upgrade. Most IDEs make it easy to find all deprecated items. In Eclipse, for example, the "Problems" view will show warnings for the use of deprecated APIs if you set it to show "All Errors/Warnings on Project".
4. **Read the Release Notes:** Each Tapestry version has a [Release Notes](#) document that lists all of the changes, including some that may cause compatibility issues with your current code. You will save yourself a lot of frustration if you carefully read this material before proceeding.

Upgrading

1. **Upgrade one step at a time:** It is usually best to upgrade to each intermediate version of Tapestry rather than skipping ahead multiple versions. Skipping versions (except for minor bug fix releases) makes it harder to find all calls to deprecated APIs (see above).
2. **Update your POM (or download the JARs manually):** If you're using Maven (or Gradle), update the version of the Tapestry dependencies in your pom.xml (or build.gradle) file. Remember to keep all of the Tapestry-supplied modules in sync. For example, don't forget to update the version of Tapestry-hibernate, Tapestry-spring, Tapestry-upload, etc.
3. **Remove old Tapestry JARs:** If you're not using Maven or Gradle (e.g. if you have the Tapestry JARs in your lib directory), be sure you remove older versions of Tapestry JARs (including JARs for any Tapestry-supplied modules).

After You Upgrade

1. **Remove cached JavaScript:** Tapestry's internal JavaScript may change between releases, and your web browser may have cached the older version (especially for Tapestry versions *prior to 5.4*). If you have set a specific [application version](#) in your application's module class (usually AppModule.java), you should increment it to ensure that the URLs to the JavaScript files will have a new version number in their paths. Doing so will cause the browser to download the latest versions from your server. Alternatively, you can just clear your browser's cache (and have all your developers and testers do the same). *This issue is usually not a problem on production servers, since you will likely increment the application version with each new production release. Also, starting in Tapestry 5.4 this is unlikely to be a problem due to file checksums being included in their URLs.*

 [Release Notes](#)

[Release Notes 5.0](#) 