Configuring Dependencies for Modular Applications

There are two ways of configuring dependencies for modular applications. (Modular applications are ones that include dynamically loadable modules, see Modular applications overview in Flex documentation.)

One way is to list the main classes for runtime-loaded modules (RLMs) in the build configuration for the main application (a.k.a. shell). This way doesn't require creating build configurations for the modules.

The other way is to create build configurations for each of the RLMs and then, in the build configuration for the main application, specify the dependencies on these build configurations.

These two approaches along with their advantages and drawbacks are discussed below.

- Specifying the dependencies by listing the main RLM classes
- Specifying the dependencies by listing the RLM build configurations

Specifying the dependencies by listing the main RLM classes

If the source code of the main application and its runtime-loaded modules are in the same IntelliJ IDEA module, you can just list the main RLM classes in the build configuration for the main application. In addition, if you want to optimize a module against the main application (this considerably reduces the size of compiled module file), you can do that by turning the corresponding option on in the UI.

Note that this way of specifying the dependencies is available for Web and desktop applications but unavailable for mobile applications.

Advantages:

- You don't need to create build configurations for your modules (RLMs).
- To use the `load-externs` and `link-report` compiler options, you don't need to specify them manually, a check box is provided in the UI to turn module optimization on or off.

Drawbacks:

- The same set of compiler options is used for the application and the modules.
- The source code of the main application and the modules must be in the same IntelliJ IDEA module.

Here are the main steps of the procedure to be used:

1. Open the build configuration settings for the main application. For instructions, see Managing build configurations and their settings and Using shortcuts to open build configuration settings.
2. On the **General** tab, to the right of the **Runtime-loaded modules** field, click  . (Alternatively, click the field and press `Shift+Enter`.)
3. In the Runtime-Loaded Modules dialog that opens, click + (`Alt+Insert`).
4. In the **Choose Main Class of Runtime-Loaded Module** dialog that opens, select the main class of the corresponding RLM and click **OK**.
5. If you want the module SWF file size to be optimized, select the **Optimize** check box.
6. In a similar way, add dependencies on other RLMs.
7. Click **OK** in the **Runtime-Loaded Modules** dialog.
8. Click **OK** in the **Project Structure** dialog.

Specifying the dependencies by listing the RLM build configurations

When using this way of specifying the dependencies, you should have build configurations for each of the RLMs. Besides, to optimize the file size of the RLMs, you should specify the `load-externs` and `link-report` compiler options manually (in the corresponding build configurations).

Advantages:

- The main application and the modules can be compiled using different sets of compilation options. (Every build configuration has its own set of compiler options.)
- The source code of the main application and the RLMs may be in the same IntelliJ IDEA module or in different modules (within the same project).

Drawbacks:

- You have to create a build configuration for each of the RLMs.
- To use the `load-externs` and `link-report` compiler options, you should specify them manually.

Here are the main steps of the procedure to be used:

1. Open the build configuration settings for the main application. For instructions, see Managing build configurations and their settings and Using shortcuts to open build configuration settings.
2. On the **Dependencies** tab, click + (`Alt+Insert`), and select **Build Configuration**.
3. In the **Add Dependency** dialog that opens, select all the necessary build configurations for the RLMs and click **OK**.
4. If you need module optimization:
    1. On the **Compiler Options** tab, in the **Additional compiler options** field, add `-link-report=<path_to_report_file>`, for example, `-link-report=c:/temp/link-report.xml`.
    2. Add `-load-externs=<path_to_report_file>` (e.g. `-load-externs=c:/temp/link-report.xml`) to the **Additional compiler options** field in all the corresponding RLM build configurations.
5. Click **OK** in the **Project Structure** dialog.

See Also

Procedures:

- Working with Build Configurations
- Building ActionScript and Flex Applications

Reference:

- Build Configuration Page for a Flash Module

Web Resources:

- Developer Community