

TracNav

- [JPFWiki](#) - Welcome Page

Introduction...

Installing JPF...

User Guide...

Developer Guide

- [Design](#)
- [Choice Generator](#)
- [Partial Order Reduction](#)
- [Attributes](#)
- [Listener](#)

MJI...

- [Bytecode Factory](#)
- [Logging](#)
- [Report](#)
- [Embedded](#)
- [JPF tests](#)
- [JPF project layout](#)
- [Create a JPF project](#)
- [Coding Conventions](#)
- [Hosting update site](#)

Projects...

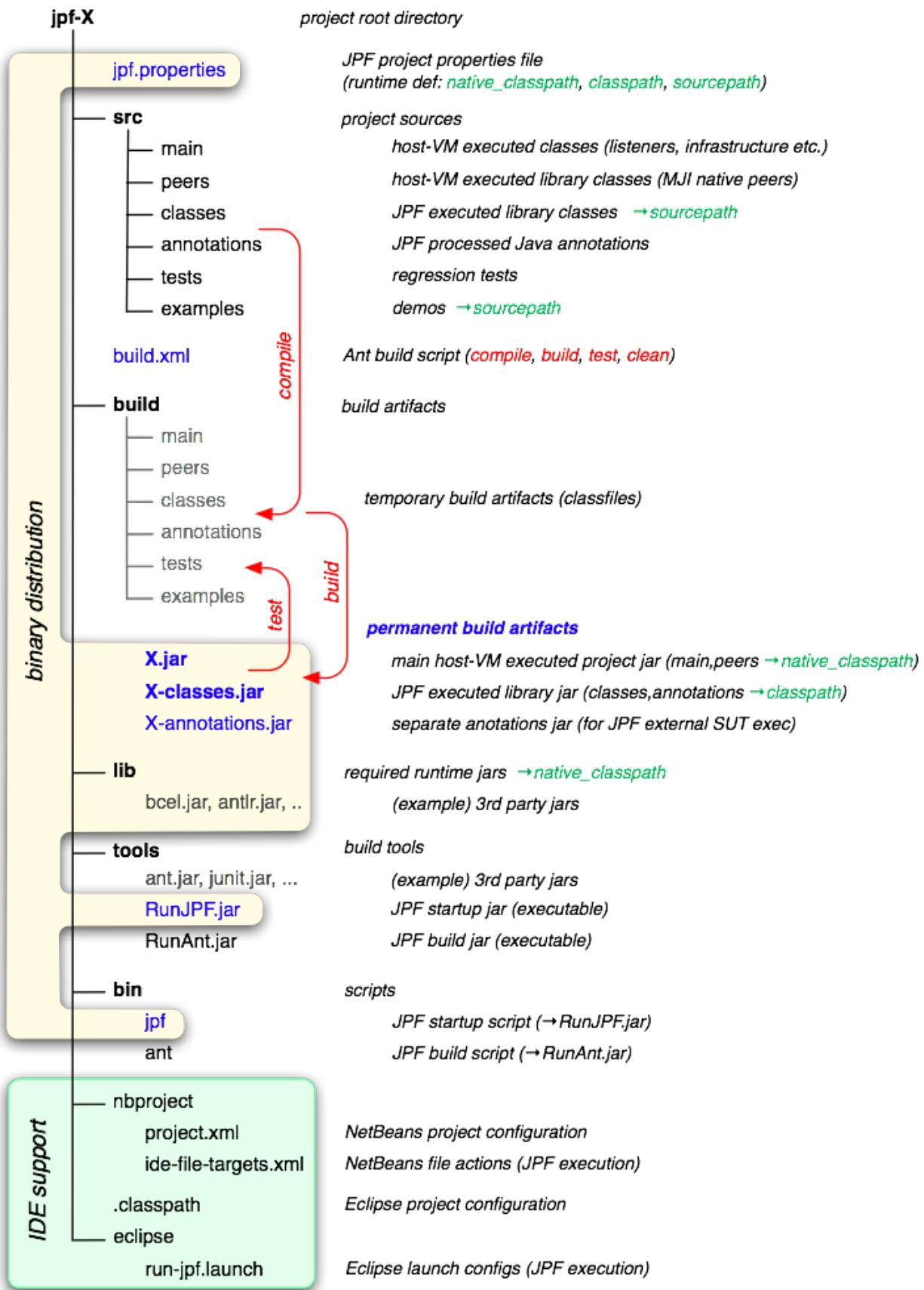
- [Summer Projects](#)
- [External Projects](#)
- [Change\(B\)log](#)

About...

- [Events](#)
- [Presentations](#)
- [Papers](#)
- [FAQ](#)
- [History?](#)
- [Support](#)
- [People?](#)
- [Playground](#)
- [Table of Context](#)

JPF Runtime Modules

JPF is partitioned into separate projects that all follow the same directory layout and build process. Projects can be distributed as source- or binary-distributions. Binary distributions are just slices through the directory tree of a source distribution that preserve the permanent build artifact, i.e. both distribution forms are runtime-compatible.



main artifacts are the *.jar files created and stored in the build directory. We can divide this into classes that are executed by the host-VM (i.e. have to be in JPFs native_classpath setting), and classes that are executed by JPF itself (i.e. have to be in JPFs classpath setting). The first category

includes [listeners](#) and [native peers](#), the second one model classes (compiled from `src/classes`) and annotations, i.e. code that is used by the system under test.

The build process is [Ant](#) based, which means every source distribution comes with a `build.xml` script that implements the basic build targets `clean`, `compile`, `build` and `test`.

As a general principle, we try to include all required 3rd party runtime and build libraries.

As an exception to this rule, the `compile` Ant target uses the standard 'javac' command, which is not included in the JPF distribution and therefore requires a full JDK installation. `test` generally executes a JUnit based regression test suite.

Both JUnit and Ant libraries are included in the jpf-core distribution, which also contains the minimal RunAnt.jar executable jar which can be distributed with other JPF projects to use the jpf-core provided 3rd party build tools.

The `lib` directory contains 3rd party libraries that are required at runtime of the project (like `bcel.jar` in jpf-core).

`tools` contains programs and libraries that are used by the build process (like `ant.jar` and `junit.jar` in jpf-core).

For convenience reasons, JPF projects come with corresponding NetBeans and Eclipse configurations, i.e. can be directly opened within these IDEs.