|  |  |
| --- | --- |
|  | |
| **Java 9 - Creating Custom runtime Image with jlink**  [Last Updated: Oct 12, 2018] | Top of Form   |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | |  |   Bottom of Form  [Previous Page](https://www.logicbig.com/tutorials/core-java-tutorial/java-9-changes/platform-logging.html) |
| Java 9 introduced jlink command-line tool which assembles and optimizes the specified modules and their dependencies into a custom runtime image.  In other words it assembles a Java application and its dependent modules (instead of all modules which come with default JDK) into a custom JRE. The resultant JRE can execute the provided application without the need of a preinstalled JRE.  jlink also does a lot of optimizations that are otherwise difficult at compile time or costly at run-time.  **Example**  Let's create a very simple modular application. We are using JDK 11 on Windows 10.  **jlink-example/src/com/logicbig/example/Test.java**  package com.logicbig.example;  import java.util.logging.Logger;  public class Test {  private static final Logger LOGGER = Logger.getLogger(Test.class.getName());  public static void main(String[] args) {  LOGGER.info("Running test application..");  }  }  **jlink-example/src/module-info.java**  module test.example {  requires java.logging;  }  **Compiling and running**  D:\jlink-example>tree /a /f Folder PATH listing for volume Data Volume serial number is 68F9-EDFA D:. \---src  | module-info.java  |   \---com  \---logicbig  \---example  Test.java  D:\jlink-example>javac -d out src\module-info.java  D:\jlink-example>javac -d out --module-path out src\com\logicbig\example\Test.java  After compilation:  D:\jlink-example>tree /a /f Folder PATH listing for volume Data Volume serial number is 68F9-EDFA D:. +---out | | module-info.class | |  | \---com | \---logicbig | \---example | Test.class |  \---src  | module-info.java  |   \---com  \---logicbig  \---example  Test.java  Let's run to see whether our application is working as expected:  D:\jlink-example>java --module-path out --module test.example/com.logicbig.example.Test Oct 11, 2018 6:15:16 PM com.logicbig.example.Test main INFO: Running test application..  **The dependent modules**  Let's use jdeps command to see the dependent modules:  D:\jlink-example>jdeps --module-path out -s --module test.example test.example -> java.base test.example -> java.logging  **Creating runtime image with jlink**  General jlink command:  jlink --module-path <module-path> --add-modules <comma-separated-module-names> --output <target-directory>  We just need to include our main modules for <comma-separated-module-names>, all other modules the main modules depend on will automatically be included.  Let's continue with our example application and create runtime image for it:  D:\jlink-example>jlink --module-path D:\java\jdk-11\jmods;out --add-modules test.example --output build  In above command we also included D:\java\jdk-11\jmods for --module-path, so that the required JDK modules (in our example: java.base and java.logging) can be resolved and included in the output image.  The generated image in 'build' folder:  D:\jlink-example\build>tree /a /f Folder PATH listing for volume Data Volume serial number is 68F9-EDFA D:. | release |  +---bin | | api-ms-win-core-console-l1-1-0.dll | | api-ms-win-core-datetime-l1-1-0.dll | | api-ms-win-core-debug-l1-1-0.dll | | api-ms-win-core-errorhandling-l1-1-0.dll | | api-ms-win-core-file-l1-1-0.dll | | api-ms-win-core-file-l1-2-0.dll | | api-ms-win-core-file-l2-1-0.dll | | api-ms-win-core-handle-l1-1-0.dll | | api-ms-win-core-heap-l1-1-0.dll | | api-ms-win-core-interlocked-l1-1-0.dll | | api-ms-win-core-libraryloader-l1-1-0.dll | | api-ms-win-core-localization-l1-2-0.dll | | api-ms-win-core-memory-l1-1-0.dll | | api-ms-win-core-namedpipe-l1-1-0.dll | | api-ms-win-core-processenvironment-l1-1-0.dll | | api-ms-win-core-processthreads-l1-1-0.dll | | api-ms-win-core-processthreads-l1-1-1.dll | | api-ms-win-core-profile-l1-1-0.dll | | api-ms-win-core-rtlsupport-l1-1-0.dll | | api-ms-win-core-string-l1-1-0.dll | | api-ms-win-core-synch-l1-1-0.dll | | api-ms-win-core-synch-l1-2-0.dll | | api-ms-win-core-sysinfo-l1-1-0.dll | | api-ms-win-core-timezone-l1-1-0.dll | | api-ms-win-core-util-l1-1-0.dll | | api-ms-win-crt-conio-l1-1-0.dll | | api-ms-win-crt-convert-l1-1-0.dll | | api-ms-win-crt-environment-l1-1-0.dll | | api-ms-win-crt-filesystem-l1-1-0.dll | | api-ms-win-crt-heap-l1-1-0.dll | | api-ms-win-crt-locale-l1-1-0.dll | | api-ms-win-crt-math-l1-1-0.dll | | api-ms-win-crt-multibyte-l1-1-0.dll | | api-ms-win-crt-private-l1-1-0.dll | | api-ms-win-crt-process-l1-1-0.dll | | api-ms-win-crt-runtime-l1-1-0.dll | | api-ms-win-crt-stdio-l1-1-0.dll | | api-ms-win-crt-string-l1-1-0.dll | | api-ms-win-crt-time-l1-1-0.dll | | api-ms-win-crt-utility-l1-1-0.dll | | java.dll | | java.exe | | javaw.exe | | jimage.dll | | jli.dll | | keytool.exe | | msvcp140.dll | | net.dll | | nio.dll | | ucrtbase.dll | | vcruntime140.dll | | verify.dll | | zip.dll | |  | \---server | jvm.dll |  +---conf | | logging.properties | | net.properties | |  | \---security | | java.policy | | java.security | |  | \---policy | | README.txt | |  | +---limited | | default\_local.policy | | default\_US\_export.policy | | exempt\_local.policy | |  | \---unlimited | default\_local.policy | default\_US\_export.policy |  +---include | | classfile\_constants.h | | jni.h | | jvmti.h | | jvmticmlr.h | |  | \---win32 | jni\_md.h |  +---legal | +---java.base | | aes.md | | asm.md | | c-libutl.md | | cldr.md | | COPYRIGHT | | icu.md | | LICENSE | | public\_suffix.md | | unicode.md | | zlib.md | |  | \---java.logging | COPYRIGHT | LICENSE |  \---lib  | classlist  | jrt-fs.jar  | jvm.cfg  | jvm.lib  | modules  | tzdb.dat  | tzmappings  |   +---security  | blacklisted.certs  | cacerts  | default.policy  | public\_suffix\_list.dat  |   \---server  Xusage.txt  The size of the image (the 'build' folder) is about 40MB as compared to 277MB of the whole JDK 11.  **Running application with generated image**  We can run our main class from bin folder as follows:  D:\jlink-example\build\bin>java.exe --module test.example/com.logicbig.example.Test Oct 11, 2018 6:15:31 PM com.logicbig.example.Test main INFO: Running test application..  In above command 'java.exe' is the one which is included in bin folder, not the one on the classpath.  **Using launcher option**  --launcher command=moduleName/mainClass option of jlink can be used to create a 'launcher script' to run our application:  Let's use this option for our above example:  D:\jlink-example>rmdir /S /Q build  D:\jlink-example>jlink --launcher test-app=test.example/com.logicbig.example.Test --module-path D:\java\jdk-11\jmods;out --add-modules test.example --output build  It created two launcher scripts test-app (for linux) and test-app.bat (for windows).  The content of test-app.bat  @echo off  set JLINK\_VM\_OPTIONS=  set DIR=%~dp0  "%DIR%\java" %JLINK\_VM\_OPTIONS% -m test.example/com.logicbig.example.Test %\*  Let's run the script:  D:\jlink-example\build\bin>test-app.bat Oct 11, 2018 6:15:41 PM com.logicbig.example.Test main INFO: Running test application..  **Example Project**  Dependencies and Technologies Used:   * JDK 9.0.1 | [Java 9 for Programmers (Deitel Developer)](https://aax-us-east.amazon-adsystem.com/x/c/QuLzpMWPDi-v7FjspI-qpuMAAAF7CgesGAEAAAFKAYT-SnY/https:/www.amazon.com/dp/0134777565/ref=sm_n_se_dkp_IN_pr_sea_0_0?adId=0134777565&creativeASIN=0134777565&linkId=ec98fbe3322cb32dce01764a9c07c107&tag=logicbig-20&linkCode=w42&ref-refURL=https%3A%2F%2Fwww.logicbig.com%2Ftutorials%2Fcore-java-tutorial%2Fjava-9-changes%2Fjlink.html&slotNum=0&imprToken=6d88edc3a25653d91e7f1b1f0f616226&adType=smart&adMode=search&adFormat=grid&impressionTimestamp=1627960880436)  **$44.31**~~$59.99~~   (61)    [Java 9 Modularity: Patterns and Practices for Develo…](https://aax-us-east.amazon-adsystem.com/x/c/QuLzpMWPDi-v7FjspI-qpuMAAAF7CgesGAEAAAFKAYT-SnY/https:/www.amazon.com/dp/1491954167/ref=sm_n_se_dkp_IN_pr_sea_0_1?adId=1491954167&creativeASIN=1491954167&linkId=ec98fbe3322cb32dce01764a9c07c107&tag=logicbig-20&linkCode=w42&ref-refURL=https%3A%2F%2Fwww.logicbig.com%2Ftutorials%2Fcore-java-tutorial%2Fjava-9-changes%2Fjlink.html&slotNum=0&imprToken=6d88edc3a25653d91e7f1b1f0f616226&adType=smart&adMode=search&adFormat=grid&impressionTimestamp=1627960880440)  **$31.45**~~$39.99~~   (25)    [Mastering Java 9: Write reactive, modular, co…](https://aax-us-east.amazon-adsystem.com/x/c/QuLzpMWPDi-v7FjspI-qpuMAAAF7CgesGAEAAAFKAYT-SnY/https:/www.amazon.com/dp/1786468735/ref=sm_n_se_dkp_IN_pr_sea_1_0?adId=1786468735&creativeASIN=1786468735&linkId=ec98fbe3322cb32dce01764a9c07c107&tag=logicbig-20&linkCode=w42&ref-refURL=https%3A%2F%2Fwww.logicbig.com%2Ftutorials%2Fcore-java-tutorial%2Fjava-9-changes%2Fjlink.html&slotNum=0&imprToken=6d88edc3a25653d91e7f1b1f0f616226&adType=smart&adMode=search&adFormat=grid&impressionTimestamp=1627960880443)  **$44.99**   (3)    [Modern Java in Action: Lambdas, streams, funct…](https://aax-us-east.amazon-adsystem.com/x/c/QuLzpMWPDi-v7FjspI-qpuMAAAF7CgesGAEAAAFKAYT-SnY/https:/www.amazon.com/dp/1617293563/ref=sm_n_se_dkp_IN_pr_sea_1_1?adId=1617293563&creativeASIN=1617293563&linkId=ec98fbe3322cb32dce01764a9c07c107&tag=logicbig-20&linkCode=w42&ref-refURL=https%3A%2F%2Fwww.logicbig.com%2Ftutorials%2Fcore-java-tutorial%2Fjava-9-changes%2Fjlink.html&slotNum=0&imprToken=6d88edc3a25653d91e7f1b1f0f616226&adType=smart&adMode=search&adFormat=grid&impressionTimestamp=1627960880445)  **$42.37**~~$54.99~~   (99)  [**Ads by Amazon**](https://aax-us-east.amazon-adsystem.com/x/c/QuLzpMWPDi-v7FjspI-qpuMAAAF7CgesGAEAAAFKAYT-SnY/https:/affiliate-program.amazon.com/home/ads/ref=sm_n_se_dkp_IN_logo?adId=logo&creativeASIN=logo&linkId=ec98fbe3322cb32dce01764a9c07c107&tag=logicbig-20&linkCode=w42&ref-refURL=https%3A%2F%2Fwww.logicbig.com%2Ftutorials%2Fcore-java-tutorial%2Fjava-9-changes%2Fjlink.html&slotNum=0&imprToken=6d88edc3a25653d91e7f1b1f0f616226&adType=smart&adMode=search&adFormat=grid&impressionTimestamp=1627960880454&ac-ms-src=nsa-ads&cid=nsa-ads)  **Core Java Tutorials**   * [Java 16 Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-16-changes.html) * [Java 15 Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-15-changes.html) * [Java 14 Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-14-changes.html) * [Java 13 Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-13-changes.html) * [Java 12 Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-12-changes.html) * [Java 11 Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-11-changes.html) * [Java 10 Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-10-changes.html) * [Java 9 Module System](https://www.logicbig.com/tutorials/core-java-tutorial/modules.html) * [Java 9 Misc Features](https://www.logicbig.com/tutorials/core-java-tutorial/java-9-changes.html) * [Java 9 JShell](https://www.logicbig.com/tutorials/core-java-tutorial/jshell.html)   **Recent Tutorials**   * [Spring - The Use Of PropertyEditors With @Value Annotation](https://www.logicbig.com/tutorials/spring-framework/spring-core/use-of-property-editors-with-value-annotation.html) * [Spring - The Use Of PropertyEditors With XML Configuration](https://www.logicbig.com/tutorials/spring-framework/spring-core/property-editors-used-with-xml-config.html) * [Spring - Use of PropertyEditors via BeanWrapper](https://www.logicbig.com/tutorials/spring-framework/spring-core/property-editors-used-by-stand-alone-bean-wrapper.html) * [Spring - Using BeanUtils for resolving string based method signatures](https://www.logicbig.com/tutorials/spring-framework/spring-core/bean-utils-resolving-method-signature-as-string-representation.html) * [Spring - Copying properties using BeanUtils](https://www.logicbig.com/tutorials/spring-framework/spring-core/bean-utils-copy-properties.html) * [Spring - Obtaining BeanInfo And PropertyDescriptors](https://www.logicbig.com/tutorials/spring-framework/spring-core/obtaining-bean-info-and-property-descriptors.html) * [Spring - Directly setting fields via DirectFieldAccess](https://www.logicbig.com/tutorials/spring-framework/spring-core/direct-field-access.html) * [Spring - BeanWrapper, setting nested beans](https://www.logicbig.com/tutorials/spring-framework/spring-core/using-bean-wrapper-to-set-nested-beans.html) * [Spring - Using BeanWrapper](https://www.logicbig.com/tutorials/spring-framework/spring-core/using-bean-wrapper.html) * [Spring - Transforming events](https://www.logicbig.com/tutorials/spring-framework/spring-core/spring-events-transforming-to-other-events.html) * [Spring - Conditional Event Handling via @EventListener annotation](https://www.logicbig.com/tutorials/spring-framework/spring-core/conditional-event-handling-via-event-listener-annotation.html) * [Spring - Using @EventListener annotation with multiple event types](https://www.logicbig.com/tutorials/spring-framework/spring-core/event-listener-annotation-with-multiple-events.html) * [Spring - Publishing and Consuming Custom Events](https://www.logicbig.com/tutorials/spring-framework/spring-core/creating-and-publishing-custom-events.html) * [Java - Introduction to ResourceBundle](https://www.logicbig.com/tutorials/core-java-tutorial/internationalization/resource-bundle-example.html) * [Spring - Injecting Resource using @Value annotation](https://www.logicbig.com/tutorials/spring-framework/spring-core/injecting-resources.html) * [Spring - Injecting ResourceLoader](https://www.logicbig.com/tutorials/spring-framework/spring-core/injecting-resource-loader.html) * [Spring - Resource Loading](https://www.logicbig.com/tutorials/spring-framework/spring-core/resource-loading.html) * [Spring - Using Spring Expression Language with @Value Annotation](https://www.logicbig.com/tutorials/spring-framework/spring-core/value-annotation-with-expression-language.html) * [Spring - Using @Value Annotation](https://www.logicbig.com/tutorials/spring-framework/spring-core/using-value-annotation.html) * [Spring - Injecting Environment to access properties in beans](https://www.logicbig.com/tutorials/spring-framework/spring-core/injecting-environment.html) * [Spring - Adding New Property Source to Environment](https://www.logicbig.com/tutorials/spring-framework/spring-core/adding-new-property-source.html) * [Spring - Adding user properties by using @PropertySource](https://www.logicbig.com/tutorials/spring-framework/spring-core/accessing-environmental-properties-by-using-property-source.html) * [Spring - Accessing Environment Properties](https://www.logicbig.com/tutorials/spring-framework/spring-core/spring-env-properties.html) * [Spring - Profiles](https://www.logicbig.com/tutorials/spring-framework/spring-core/spring-profiles.html) * [Spring - Successfully injecting circular dependencies using @Lazy Annotation](https://www.logicbig.com/tutorials/spring-framework/spring-core/avoid-circular-dependencies-exception-with-lazy-annotation.html) * [Spring - Successfully injecting circular dependencies using setter injection](https://www.logicbig.com/tutorials/spring-framework/spring-core/avoid-circular-dependencies-with-setter.html) * [Java 16 - Records Features, Quick Walk-through](https://www.logicbig.com/tutorials/core-java-tutorial/java-16-changes/java-records-features-quick-walk-through.html) * [Java 16 - Introduction to Records](https://www.logicbig.com/tutorials/core-java-tutorial/java-16-changes/intro-to-java-records.html) * [Spring - Injecting beans into Arrays/Collections, Using @Qualifiers And Specifying the Ordering](https://www.logicbig.com/tutorials/spring-framework/spring-core/using-qualifiers-and-specifying-the-ordering.html) * [Injecting Collections - Injecting Beans Into Arrays And Collections, ordering with Ordered Interface](https://www.logicbig.com/tutorials/spring-framework/spring-core/injecting-beans-into-arrays-and-collections-with-ordered-interface.html) * [Spring - Injecting beans Into Arrays and Lists, ordering with @Ordered annotation](https://www.logicbig.com/tutorials/spring-framework/spring-core/injecting-beans-into-arrays-and-collections-with-ordered-annotation.html) * [Spring - Injecting beans into Arrays and Collections, selecting elements with @Qualifier annotation](https://www.logicbig.com/tutorials/spring-framework/spring-core/injecting-arrays-and-collection-selecting-elements-with-qualifier-annotation.html) * [Spring - Injecting multiple Beans Into Arrays and Collections](https://www.logicbig.com/tutorials/spring-framework/spring-core/injecting-multiple-beans-into-arrays-and-collections.html) * [Spring - Arrays and Collections As Beans](https://www.logicbig.com/tutorials/spring-framework/spring-core/arrays-and-collections-as-beans.html) * [Spring - Using @ComponentScan#excludeFilters to exclude classes from scanning based on annotations](https://www.logicbig.com/tutorials/spring-framework/spring-core/component-scan-exclude-filter-with-filter-type-annotation.html) * [Spring - Using @ComponentScan#includeFilters to scan non component classes based on annotations](https://www.logicbig.com/tutorials/spring-framework/spring-core/component-scan-include-filter-with-filter-type-annotation.html) * [Spring - Implementing ApplicationContextAware Interface](https://www.logicbig.com/tutorials/spring-framework/spring-core/implementing-application-context-aware-interface.html) * [Spring - Using excludeFilters attribute of @ComponentScan to exclude component classes](https://www.logicbig.com/tutorials/spring-framework/spring-core/component-scan-using-exclude-filter-for-component-classes.html) * [Spring - Using @ComponentScan to scan non component classes via includeFilters attribute](https://www.logicbig.com/tutorials/spring-framework/spring-core/component-scan-include-filter-for-non-component-classes.html) * [Spring - Using Filters To Customize Scanning with @ComponentScan](https://www.logicbig.com/tutorials/spring-framework/spring-core/component-scan-using-filters-to-customize-scanning.html) * [Spring - Using basePackageClasses Attribute of @ComponentScan](https://www.logicbig.com/tutorials/spring-framework/spring-core/component-scan-base-package-classes-attribute.html) * [Spring - Specifying packages to be scanned with basePackages attribute of @ComponentScan](https://www.logicbig.com/tutorials/spring-framework/spring-core/component-scan-base-packages-attribute.html) * [JUnit - How to test user command line Input in Java?](https://www.logicbig.com/how-to/junit/java-test-user-command-line-input.html) * [Spring - Session based Prototype Bean Example](https://www.logicbig.com/tutorials/spring-framework/spring-core/spring-session-based-prototype-bean-example.html) * [Spring - Prototype Bean Example](https://www.logicbig.com/tutorials/spring-framework/spring-core/spring-prototype-bean-example.html) * [Spring - Singleton Bean Example](https://www.logicbig.com/tutorials/spring-framework/spring-core/spring-singleton-bean-example.html) * [Spring - Receiving lifecycle callbacks by implementing InitializingBean and DisposableBean](https://www.logicbig.com/tutorials/spring-framework/spring-core/lifecycle-callbacks-by-implementing-spring-interfaces.html) * [Spring - Receiving lifecycle callbacks by using 'initMethod' and 'destroyMethod' of @Bean annotation](https://www.logicbig.com/tutorials/spring-framework/spring-core/lifecycle-callbacks-using-bean-annotation.html) |
| |  |  | | --- | --- | | ui-buttonui-button Java 9 - jlink example | [Select All](javascript:selectAllCode(1);) [Download](https://www.logicbig.com/tutorials/core-java-tutorial/java-9-changes/jlink/jlink-example.zip) |  * jlink-example   + src     - com       * logicbig         + example   **Test.java**   * + - **module-info.java**   package com.logicbig.example;  import java.util.logging.Logger;  public class Test {  private static final Logger LOGGER = Logger.getLogger(Test.class.getName());  public static void main(String[] args) {  LOGGER.info("Running test application..");  }  } | |