Java

The Maven package `com.weblitzpdf:weblitzpdf` is Weblitz for Java: HTML, a file or a web address to a PDF, an Excel, Word or PowerPoint file, or a picture, printed as Chrome prints them, with no browser to install, start or drive. The same package renders RDLC and RDL reports and Markdown. Its options are Playwright's `Page.PdfOptions` under the same names.

Install

The Java packages are in Weblitz's own Maven repository, `https://maven.weblitzpdf.com`, not in Maven Central. Add the repository and the dependency:

```
<repositories>
    <repository>
        <id>weblitzpdf</id>
        <url>https://maven.weblitzpdf.com</url>
    </repository>
</repositories>

<dependencies>
    <dependency>
        <groupId>com.weblitzpdf</groupId>
        <artifactId>weblitzpdf</artifactId>
        <version>2.0.0-rc.2</version>
    </dependency>
</dependencies>
```

Version 2.0.0-rc.2, a prerelease of 2.0, the version of every package. Java 17 or later, on Linux with glibc (x64, Arm64), macOS (Arm64, x64) and Windows (x64, Arm64), with no other dependency. The engine runs in a worker process that comes as a jar of its own for each platform, `com.weblitzpdf:weblitzpdf-worker:2.0.0-rc.2:<platform>` (linux-x64, linux-arm64, osx-arm64, osx-x64, win-x64, win-arm64): it holds the engine and the .NET runtime it runs on, so there is nothing else to install and no .NET on the machine is needed. Its first use copies it into the user's cache, which takes a second or two once.

Maven adds the worker of the machine it builds on by itself. A build for another machine names it, and so does Gradle:

```
<!-- Maven: an application built on a Mac for a Linux server -->
<dependency>
    <groupId>com.weblitzpdf</groupId>
    <artifactId>weblitzpdf-worker</artifactId>
    <version>2.0.0-rc.2</version>
    <classifier>linux-x64</classifier>
    <scope>runtime</scope>
</dependency>

// Gradle (Kotlin)
repositories { maven { url = uri("https://maven.weblitzpdf.com") } }
dependencies {
    implementation("com.weblitzpdf:weblitzpdf:2.0.0-rc.2")
    runtimeOnly("com.weblitzpdf:weblitzpdf-worker:2.0.0-rc.2:linux-x64")
}
```

Without the worker a call fails with the code `workerMissing`, naming the jar to add. Alpine and other musl systems are not supported yet; on a slim Linux image, add fontconfig and ICU (`libfontconfig1` and `libicu72`, the ICU package's number following the system's release).

Your first PDFs

```
import com.weblitzpdf.*;
import java.nio.file.Files;
import java.nio.file.Path;

// HTML text
Result invoice = Weblitz.renderHtml("<h1>Invoice 42</h1><p>...</p>",
        new HtmlOptions().setFormat("A4").setPrintBackground(true));
invoice.save(Path.of("invoice.pdf"));

// A web page, after its scripts ran
Weblitz.renderUrl("https://example.com/report",
        new HtmlOptions().setMargin(new
Margin().setTop("18mm").setRight("18mm").setBottom("20mm").setLeft("20mm")))
       .save(Path.of("report.pdf"));

// A file on disk: its CSS, pictures, fonts and scripts are read from its folder
Result letters =
Weblitz.renderHtml(Files.readString(Path.of("templates/letters.html")),
        new HtmlOptions().setBaseUrl(Path.of("templates").toUri().toString()));

// What could not be drawn or loaded is said, never thrown
for (Note note : invoice.notes()) System.out.println(note.severity() + " " +
note.code() + ": " + note.message());
```

A render answers a `Result` (`file()`, `contentType()`, `extension()`, `pages()`, `notes()`, `elapsedMs()`, `trial()`, and `save(path)`) or throws a `WeblitzException`, whose `code()` says why. The options are .NET's options under Java's names: `setFormat`, `setMargin`, `setLandscape`, `setPrintBackground`, `setPageRanges`, `setHeaderTemplate`, `setBaseUrl`, `setOutput`, `setScripts`, `setNetwork`, `setLocale`, `setTimeZone`, `setNow`...

Excel, Word, PowerPoint and pictures

```
Weblitz.renderUrl("https://example.com/prices", new HtmlOptions().setOutput("xlsx"))
// or "docx", "pptx"
       .save(Path.of("prices.xlsx"));
Weblitz.renderUrl("https://example.com", new HtmlOptions().setOutput("png")
        .setViewport(new Viewport(1280,
720).setDeviceScaleFactor(2))).save(Path.of("example.png"));
Result pages = Weblitz.renderHtml(html, new
HtmlOptions().setOutput("png").setCapture("pages").setFormat("A4"));   // a ZIP
```

`setOutput("png")` (or `"jpeg"`, `"webp"`, `"tiff"`) draws the page instead of printing it: pictures of a page.

What a page loads

A page loads nothing from the network by default but, for `renderUrl`, its own host. `setNetwork` opens more, with headers and cookies for pages behind a login, each going only to the hosts it names:

```
Weblitz.renderUrl("https://app.example.com/reports/42", new
HtmlOptions().setNetwork(new Network()
        .setAllowedHosts("app.example.com", "cdn.example.com")
        .addHeader("Authorization", "Bearer " + token, "app.example.com")));
```

Reports and Markdown

The same package renders SSRS2's reports: `renderReport`, `checkReport`, `markdownTo` and `importReport`. A dataset's rows are Java's own values: records, beans, maps, lists and streams. `net.ssrs2:ssrs2` (`net.ssrs2:ssrs2:2.0.0-rc.2`) has the same functions for projects that only render reports; it runs on `com.weblitzpdf:weblitzpdf`.

```
import net.ssrs2.Ssrs2;   // or Weblitz's functions of the same names

record Item(String description, BigDecimal price, int qty) {}

Ssrs2.renderReport(Path.of("Reports/Invoice.rdlc"), new ReportOptions()
        .setFormat("PDF")               // or WORDOPENXML, EXCELOPENXML, PPTX, IMAGE,
PNG, HTML5, CSV, XML
        .setDataset("Items", List.of(new Item("Widget", new BigDecimal("104.99"), 1)))
        .setParameter("InvoiceNumber", "42"))
     .save(Path.of("invoice.pdf"));

CheckResult check = Ssrs2.checkReport(Path.of("Reports/Invoice.rdlc"));   // its
problems, without rendering it
```

Workers, futures and timeouts

Calls go to a pool of worker processes kept warm, started on first use; any thread may make them. `Weblitz.configure(...)` sets its options; `new Engine(...)` is a pool of your own, which `close()` ends. Every method has an `...Async` twin answering a `CompletableFuture`, and cancelling it cancels the render. Times are in milliseconds.

```
Weblitz.configure(new EngineOptions().setWorkers(2).setTimeout(30_000));

try (Engine engine = new Engine(new EngineOptions().setWorkers(4))) {
    Result result = engine.renderHtml(html, new HtmlOptions().setTimeout(5_000));
    CompletableFuture<Result> later = engine.renderHtmlAsync(html, new HtmlOptions());
}
```

On a server

A server that prints pages other people send should start its workers with SSRS2's server settings: a page then loads nothing but a URL's own host, private addresses and `file:` base URLs are refused, report code is always restricted, and a render stops at 2,000 pages or 120 seconds. `SSRS2_WORKER_SETTINGS` changes them, as JSON (the Node.js page shows it).

```
Weblitz.configure(new EngineOptions().setEnv("SSRS2_WORKER_POLICY", "service"));
```

Licence

The SSRS2 licence covers Weblitz and SSRS2's reports. Give the key as `setLicense(...)` on the engine's options or as the environment's `SSRS2_LICENSE`. Without one, PDF and image pages carry a trial watermark and a result's `trial()` is true. Buy a licence