Extracted from:

Deploying with JRuby

Deliver Scalable Web Apps Using the JVM

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

Deploying with JRuby

Deliver Scalable Web Apps Using the JVM

Joe Kutner

Edited by Brian P. Hogan

The Facets of Ruby Series

1.2 Introducing Warbler

Warbler is a gem that can create a web application archive (WAR) file from a Rails, Merb, or Rack-based application.

A WAR file is a zip file that follows a few conventions, but we don't have to worry about those conventions because Warbler takes care of them for us. What we do need to know is how to use the Warbler commands to package our application.

The WAR file that Warbler creates will be completely self-contained and ready to be deployed to a Java web server. Warbler bundles JRuby, your web framework, and all of the dependencies needed to adapt a Ruby web application to a Java-based container.

To demonstrate Warbler, let's create the simplest web application we can. First, we'll create a directory called `myapp`. In that directory, we'll create a `config.ru` file and put the following code into it:

Warbler/myapp/config.ru

```
run lambda { |env|
    [200, {'Content-Type' => 'text/html'}, 'Hello, World']
}
```

Next, we need to install the Warbler gem to our JRuby gem path with this command:

```
$ gem install warbler
Successfully installed jruby-jars-1.6.7
Successfully installed jruby-rack-1.1.4
Successfully installed rubyzip-0.9.6.1
Successfully installed warbler-1.3.4
4 gems installed
```

Warbler has two JRuby-specific dependencies. The `jruby-jars` gem includes the core JRuby code and standard library files. This allows other gems to depend on JRuby without freezing to a specific version. The other dependency, the `jruby-rack` gem, is responsible for adapting the Java web server specification to the Rack specification.

Next, let's use the `warble` command to create our archive file. In the same directory as the `config.ru` file we created earlier, we'll run it with the `war` option.

```
$ warble war
```

This will create a `myapp.war` file. In Chapter 3, *Deploying an Archive File*, on page ?, we will discuss all the different ways we can deploy this WAR file. For now, we just want to be able to run it so we can demonstrate how Warbler

Joe asks:
What's in a WAR File?

A WAR file is a special case of Java archive (JAR) file; both are really just zip files. But a WAR file is structured according to a standard that is recognized by all Java web servers. We can take a closer look at this by extracting the WAR file we created in this chapter with any unzipping tool. Inside of it, we find these essential components (among many other things):

```
twitalytics.war
|-- index.html
|-- META-INF/
    `-- MANIFEST.MF
`-- WEB-INF/
    |-- lib/
    `-- web.xml
```

The top-level directory contains all client-accessible content, which is equivalent to the public directory in a Rails application. This is where we'll find all of the HTML files, images, and other static content. The WEB-INF directory contains all the dynamic content for our web application. This includes our Ruby scripts, and the Java libraries need to run a JRuby application. The META-INF directory contains basic metadata about the WAR file, such as who created it and when it was created.

Inside the WEB-INF directory is the web.xml file, which is the most important part of the archive. It contains a description of how the components in the web application are put together at runtime. It's similar to the config/application.rb, config/environment.rb, and config/routes.rb files of a Rails application all put together into a single descriptor. Fortunately, Warbler handles the creation of this file for us based on the settings in our config/warbler.rb file.

A WAR file can be digitally signed, which creates a checksum for each file contained in the archive. This is used by a web server to ensure that no one has tampered with it or that it has not been corrupted in some way. If the checksums do not match, then the server won't load the files.

works. To do this, we'll create an executable WAR file. Let's build the WAR file again by running the same command with the executable option.

```
$ warble executable war
```

This will create a WAR file that is capable of running on its own, without the need for a free-standing Java web server. You probably won't want to use this in production, but it will help us test our archive file. We can run it with this command:

```
$ java -jar myapp.war
```

When the server is started, you'll be able to access the application at http://localhost:8080.

That's all you need to know to get started with Warbler. Now let's make some adjustments to the Twitalytics application. It wasn't built to run on JRuby, so it has some code that's specific to MRI. We're going to fix these parts so they work on our new platform.