Customizations

Custom Development

For a full example of creating a custom module for the application, see the `pac4j-module` project.

As a Spring Boot application built with gradle, there are many opportunities available for custom development against the application. Note that most customizations should be done in separate modules, not directly modifying the source to the base application.

Application Classpath

To get customizations in the application, one must make the changes available on the application classpath. Depending on the deployment method, one has a few options for doing this.

1. Take advantage of the `Properties Launcher` in the executable JAR
2. Put resources in the appropriate `WEB-INF` directory for a WAR deployment

If one is adding Spring configuration, register the configuration classes with the Spring Autoconfiguration service by adding a file called `META-INF/spring.factories` to the JAR file containing the custom java class and referencing any configuration classes:

```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=org.example.SomeConfiguration
```

For more information, see [https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-developing-auto-configuration.html#boot-features-locating-auto-configuration-candidates]

Executable JAR

The executable JAR uses the Properties Launcher provided by Spring Boot ([https://docs.spring.io/spring-boot/docs/current/reference/html/executable-jar.html#executable-jar-property-launcher-features]). The easiest way to add something to the classpath is to create a file named `loader.properties` in the same directory as the jar with the following:

```
loader.path=libs/
```

The noted directory will be added to the classpath, along with any JAR files contained in the directory.

WAR

If deploying a WAR, one would use the standard packaging for providing items to the classpath.

- JAR files should be placed in `WEB-INF/lib`
- all other resources should be placed `WEB-INF/classes`

It is highly recommended that a WAR overlay be used to prevent changing the version fingerprint. Overlay methods exist for both Maven ([https://maven.apache.org/plugins/maven-war-plugin/overlays.html]) and Gradle ([https://github.com/scalding/gradle-waroverlay-plugin]) projects.

Sample Customization

For an example of a customization, refer to the `pac4j-module` in the project source. This module overrides the delivered authentication method to provide a simple SAML authentication method. Along with code changes required, it also shows how one would tie it together using a Docker image.

Use Cases and Strategies

Some ideas and strategies for customizing the application.

TBD