報道各位

2020 年 4 月 16 日
株式会社インプスR&D
https://nextpublishing.jp/

Kotlin でのバックエンド開発をもっと詳しく！

『もっと実践！サーバーサイド Kotlin』発行

技術の泉シリーズ、4 月の新刊

インプレスグループで電子出版事業を手がける株式会社インプスR&D は、『もっと実践！サーバーサイド Kotlin』（著者:FORTE）を発行いたします。

最新の知見を発信する『技術の泉シリーズ』は、「技術書典」や「技術書同人誌博覧会」をはじめとした各種即売会や、勉強会・LT 会などで頒布された技術同人誌を底本とした商業書籍を刊行し、技術同人誌の普及と発展に貢献することを目指します。

『もっと実践！サーバーサイドKotlin』
https://nextpublishing.jp/isbn/9784844378631

著者:FORTE
小売希望価格:電子書籍版　1600 円（税別）／印刷書籍版　2000 円（税別）
電子書籍版フォーマット:EPUB3／Kindle Format8
印刷書籍版仕様:B5 判／カラー／本文 110 ページ
ISBN:978-4-8443-7863-1
発行:インプレス R&D

＜＜発行主旨・内容紹介＞＞

本書は「入門!実践!サーバーサイド Kotlin」の内容を発展させた続編です。前書で作成した簡単な掲示板アプリにより発展的な機能を追加しながら、バックエンド開発についてさらに学びを深めることが出来ます。

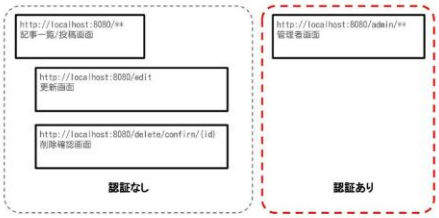
（本書は、次世代出版メソッド「NextPublishing」を使用し、出版されています。）

ユーザー登録と認証についての解説を掲載

たびに毎回パスワードが変わってしまいますし、いちいちログからパスワードを拾ってこなければなりません。ユーザー名もuser固定ですし、パスワードは長すぎて覚えられないでしょう。もっと自分で扱いやすいパスワードにしたいと思ってしまいます。

つまりまったく実用的ではないわけです。そこでSpring Securityの設定を変更して実用的な仕組みにしてみましょう。認証のイメージとしては管理者しかアクセスできない管理者画面を作成し、そのページに対する認証と認可を設定してみます。

図1.3: 管理者画面のイメージ



1.4.1 認証情報として任意のユーザー名とパスワードを設定する

まずは認証情報として任意のユーザー名とパスワードの設定を行っていきます。まずは「/src/main/kotlin/com/example/app/bbs/」の下に「config」というパッケージを作成し、新規ファイル「BbsAdminWebSecurityConfig.kt」を作成します。中身は次のとおりです。なお、import文は紙面の都合で途中で改行しています。実装時は改行しないようしてください。

リスト1.1: 管理者画面認証情報設定

```
 1: package com.example.app.bbs.config
 2:
 3: import org.springframework.beans.factory.annotation.Autowired
 4: import org.springframework.context.annotation.Bean
 5: import org.springframework.context.annotation.Configuration
 6: import org.springframework.security.config.annotation.authentication.
 7: builders.AuthenticationManagerBuilder
 8: import org.springframework.security.config.annotation.web.configuration.
 9: EnableWebSecurity
10: import org.springframework.security.config.annotation.web.configuration.
11: WebSecurityConfigurerAdapter
12: import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder
13: import org.springframework.security.crypto.password.PasswordEncoder
14:
15:
16: @Configuration
17: @EnableWebSecurity
18: class BbsAdminWebSecurityConfig : WebSecurityConfigurerAdapter() {
19:
20:     @Autowired
21:     lateinit var passwordEncoder: PasswordEncoder
22:
23:     @Bean
24:     fun passwordEncoder(): PasswordEncoder {
25:         return BCryptPasswordEncoder()
26:     }
27:
28:     @Override
29:     override fun configure(auth: AuthenticationManagerBuilder) {
30:
31:         auth.inMemoryAuthentication()
32:                 .withUser("admin")
33:                 // 次は解説用に平文で記載しているので実際にやってはダメ！
34:                 .password(passwordEncoder.encode("root"))
35:                 .authorities("ROLE_ADMIN")
36:     }
37: }
```

ここで新しく登場した構文について解説していきます。

継承とクラス名のあとの括弧

今回は「BbsAdminWebSecurityConfig : WebSecurityConfigurerAdapter()」となっていますが、この継承元についている括弧はコンストラクタを呼びだす、という指定になります。KotlinではJavaと違って明示的にコンストラクタを定義して継承元のコンストラクタを呼び出すような書き方をしなくとも、継承元の宣言時にコンストラクタを一緒に呼び出すことが可能となっています。

override句

メソッド宣言の前にある「override」句は明示的にオーバーライドすることを宣言します。Kotlinはデフォルトでメソッドのオーバーライドは禁止されています。そのため、オーバーライドしたい

アプリケーションの公開を実践してみる

第2章 デプロイ

次にデプロイについて解説していきます。これまでは自分の開発環境でしか動かせませんでしたが、デプロイすることで外部に公開することができます。システムやサービスとして公開することはもちろん、一時的に公開して仲間内で触ってもらい、フィードバックを得ることなども可能になります。

ただし、外部公開するということはさまざまなリスクを伴いますので、それを踏まえて慎重な検討、作業を行うようにしましょう。

2.1 herokuによるデプロイ

今回はheroku（へろく）というサービスを用いてデプロイします。herokuはアプリやサービスのインフラ（WebやDBサーバー）のクラウドサービスです。今まではオンプレ（物理的なサーバーを用いるインフラ形態）にしろクラウドインフラ（AWSなど）にしろサーバーの構築、設定作業が必要でしたが、今回のような簡単なサンプルアプリを公開するにしては必要な作業や学ばなければならないことが多いため、割りに合わない作業となっていました。herokuはこういった構築、設定作業を自動化、クラウドサービス化することで省力化しアプリやサービスをすぐに公開できるようにしたサービスになります。

2.2 デプロイのリスクと対策

最初に外部公開することで発生するリスクとその対策について解説しておきます。

2.2.1 リスク

まずシステムの脆弱性を突かれることによる情報漏えいや、乗っ取りが考えられます。これにより個人情報を扱うサービスであれば個人情報の漏洩が発生したり、サーバーを踏み台にされて別のサービスやサイトに攻撃をかけてしまうことなどが考えられます。

さらにそういった自体を引き起こすと評判の低下に繋がり、特にフリーランスのように次の仕事に評判が直接影響してくる状態であれば致命傷となりかねません。もちろんサラリーマンだから大丈夫、というわけでもありません。

こういったリスクがあることを理解した上でデプロイしている、という認識を持ちましょう。

2.2.2 対策

では、そのリスクの対策について解説していきますが、まず完璧な対策はないということを意識しておきましょう。どこまでいっても人はミスしますし、もし悪意を持ったユーザーからの攻撃対象にさらされれば日進月歩で新しくなる攻撃手法に完璧に対応するのは非常に難しいです。

そのうえで行える対策としては、やはり知識が大事となります。たとえば本、インターネットを用いてサービスを公開する際の注意点を調べたり、周囲のITエンジニアに公開したい旨を伝えてセキュリティについて相談してみましょう。さらには脆弱性診断ツールのようなセキュリティ関連ツールの使用も検討もします。

またもしものときを想定して、すぐにサービスを止める、止められるようにしておきましょう。特に個人情報に代表されるデータの漏洩は非常に大きな問題となるので、注意し過ぎぐらいがちょうどいいかもしれません。なにかあったときに調査できるようにログなども出力しておく必要があります。こういったセキュリティの観点での相談や確認は、有料で依頼するのもよいかもしれません。セキュリティエンジニアやセキュリティ監査を行う会社も存在しています。

なお、セキュリティについては今後も勉強を続けてこのシリーズで取り扱っていく予定です。最終的には個人サービスとして公開した上で、その運用経験なども含めて公開していけたらと考えています。少し骨すような形になってしまいましたが、なにかあってからでは遅いのがセキュリティです。常に意識するくらいがちょうどよいでしょう。

2.3 デプロイの準備

それではデプロイについて方法を解説していきます。

2.3.1 認証情報の環境変数化

これまでデータベースの接続情報（ID、パスワード）はapplication.propertiesで管理してきました。しかし、このファイルをGithubなどで公開してしまうと外部からデータベースにアクセスできてしまう可能性が飛躍的に高まります。Githubに公開しないという手段もありますが、herokuを用いる今回の仕組みではそういうわけにも行きませんし、今回は一般的に用いられているOSの環境変数にIDパスワードを登録し、それを読み出す形式で対応します。

「application.properties」のH2 Databaseをコメントアウトし、PostgreSQLを追加します。

application.propertiesにPostgreSQLの設定を追加

```
# H2 Database
#spring.jpa.hibernate.ddl-auto=update
#spring.datasource.driver-class-name=org.h2.Driver
#spring.datasource.url=jdbc:h2:./h2db/db_bbs
#spring.datasource.username=username
#spring.datasource.password=password

～省略～

# PostgreSQL(heroku)
spring.jpa.hibernate.ddl-auto=update
```

herokuを使った公開方法を紹介

brewでインストール

```
brew tap heroku/brew && brew install heroku
```

Macでインストールが成功したかは次のコマンドでバージョンが表示されればOKです。

バージョン確認

```
heroku --version
```

図2.7: バージョン確認

```
FORTE-MacBook-Pro:server_side_kotlin_2 fortegp05$ heroku --version
heroku/7.35.1 darwin-x64 node-v12.13.0
FORTE-MacBook-Pro:server_side_kotlin_2 fortegp05$
```

2.3.5 heroku cliでログインする

heroku cliがインストールできたら、さきほど作成したユーザーにログインしてみます。次のコマンドを実行しましょう。

heroku cliでherokuへのログイン

```
heroku login
```

するとブラウザーにてログインするように促されるのでq以外の任意のキー（EnterでOK）を押します。立ち上がったブラウザーでLoginボタンをクリックするとログイン完了です。次のようにコンソールに「Logging in... done」が表示されていればOKです。

図2.8: ログイン完了

```
FORTE-MacBook-Pro:server_side_kotlin_2 fortegp05$ heroku login
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/
Logging in... done
Logged in as fortegp05@gmail.com
FORTE-MacBook-Pro:server_side_kotlin_2 fortegp05$
```

なお、この手順はWindowsもMacも違いはありません。また、ユーザー登録した直後などブラウザーでログインしている状態であればLoginボタンをクリックするだけで済みますが、未ログインであれば認証情報を入力する必要があります。

2.3.6 gitのインストール

herokuはgitの利用が前提となっています。紙面の都合上、本書では解説しませんが「git mac」や「git windows」などでググればふんだんに解説記事がでてきます。また本書のような技術同人誌、商用の技術書が存在しています。ぜひそちらをご覧になってください。個人的には湊川あいさんの「マンガで分かるGit」シリーズがわかりやすくてお勧めです。

2.3.7 Githubの準備

gitをインストールしたらGithubにリポジトリーを作成し、masterブランチにpushします。こちらも紙面の都合により解説はしませんが、ググれば解説記事が出てくると思います。

2.3.8 psqlをインストールする

herokuではデータベースとしてPostgreSQLを使用します。単純にアプリからPostgreSQLを利用するだけなら特に何も準備はいらないのですが、今回は最初に管理者ユーザーを登録してあげる必要があるため、PostgreSQLを操作できるツールが必要になります。そのツールがpsqlになります。

2.3.9 psqlをWindowsにインストールする

psqlをWindowsにインストールするにはPosrgeSQLをインストールする必要があります。公式サイトからダウンロードしましょう。

https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

図2.9: インストーラーのダウンロード

PostgreSQL Database Download

今回はバージョン12で解説しますので、12.2のWindows 64bit版をダウンロードします。ダウンロードしたインストーラーを起動します。

＜＜目次＞＞

＜＜著者紹介＞＞

FORTE

Web アプリケーションのバックエンドエンジニアですが、いろいろやっています。Twitter、ブログ、Podcast 配信、数多くの趣味と楽しく活動中。今回はプログラミング言語とフレームワークについて、自分が勉強したことをまとめた技術同人誌の執筆に挑戦してみました。

＜＜販売ストア＞＞

電子書籍:

Amazon Kindle ストア、楽天 kobo イーブックストア、Apple Books、紀伊國屋書店 Kinoppy、Google Play Store、honto 電子書籍ストア、Sony Reader Store、BookLive!、BOOK☆WALKER

印刷書籍:

Amazon.co.jp、三省堂書店オンデマンド、honto ネットストア、楽天ブックス

※　各ストアでの販売は準備が整いしだい開始されます。

※　全国の一般書店からもご注文いただけます。

【インプレス R&D】 https://nextpublishing.jp/

株式会社インプレス R&D（本社：東京都千代田区、代表取締役社長：井芹昌信）は、デジタルファーストの次世代型電子出版プラットフォーム「NextPublishing」を運営する企業です。また自らも、NextPublishing を使った「インターネット白書」の出版など IT 関連メディア事業を展開しています。

※NextPublishing は、インプレス R&D が開発した電子出版プラットフォーム（またはメソッド）の名称です。電子書籍と印刷書籍の同時制作、プリント・オンデマンド（POD）による品切れ解消などの伝統的出版の課題を解決しています。これにより、伝統的出版では経済的に困難な多品種少部数の出版を可能にし、優秀な個人や組織が持つ多様な知の流通を目指しています。

【インプレスグループ】 https://www.impressholdings.com/

株式会社インプレスホールディングス（本社：東京都千代田区、代表取締役：唐島夏生、証券コード：東証 1 部 9479）を持株会社とするメディアグループ。「IT」「音楽」「デザイン」「山岳・自然」「旅・鉄道」「学術・理工学」を主要テーマに専門性の高いメデイア&サービスおよびソリューション事業を展開しています。さらに、コンテンツビジネスのプラットフォーム開発・運営も手がけています。

【お問い合わせ先】

株式会社インプレス R&D NextPublishing センター
TEL 03-6837-4820
電子メール: np-info@impress.co.jp