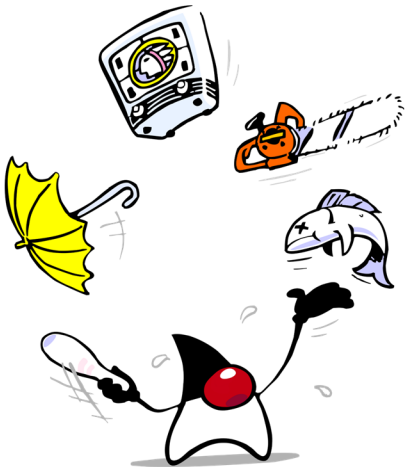
Developing Web Applications

with Play & Scala



Paul E. Sevinç, Dr. sc. ETH Zürich

Developing Web Applications

with Play & Scala

Paul E. Sevinç

This book is available at https://leanpub.com/DevWebApps

This version was published on 2025-09-18

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Cover-Picture Sources **1**

Preface **2**

Prerequisites **3**
Tools of the Trade 4

Teaching Aid **8**
Schedule like it's 200X 8

Getting Started: Hello, Play! 11

Project Layout **12**
Outwith: Überproject 12
Within: So Long, Layered MVC! 12
Monolith, Modulith, and Microservices 12

New **13**
... Repository 13
... Application 13

Dependency Injection **14**
Controllers and Filters 14

Two Become One 1 . . . . . **15**
Dude, where's my API? 15
Prepare for Launch 15

GUI Libraries **16**
Bootstrap 16
React Router 16
Getting htmx and Bootstrap as well as Bootstrap Icons 16

Security as a Forethought **17**
Hardening the backend 17
Hardening the frontend 22

Rinse & Repeat . . . 25

Database Management System . . . **27**
MongoDB . . . 27
Event Sourcing . . . 27
MongoDB . . . 27

Internationalization and Localization . . . **29**
Both Fixadat and SVPofWFTTC . . . 29
Only Fixadat: Backend . . . 29
Only Fixadat: Frontend . . . 29

CI/CD . . . 31

Two Become One 2 . . . **32**
Alternatives . . . 32
Building Images Locally . . . 32
Running Containers Locally . . . 32

Continuous Integration . . . **33**
Test . . . 33
Scan . . . 33

Continuous Delivery . . . **34**
Pull . . . 34
Push . . . 34

Continuous Deployment . . . **35**
MongoDB . . . 35
Docker or Java . . . 35

Features . . . 37

Going Public . . . **38**
index.html . . . 38
Records and Case Classes . . . 38
Primitive Types . . . 38

Interaction Design . . . **39**
Users and Interactions . . . 39
Boilerplate . . . 39

Configuration . . . **40**
Configuration Files . . . 40

Application Secret 40
Session Cookie 40
Logging 40

Template **41**

Boilerplate 42

Privacy **43**

Legalese **44**

Appendix 45

Updating and Upgrading **46**
Updating the Backend 46
Updating the Frontend 46
Upgrading the Backend 46
Upgrading the Frontend 46

Cover-Picture Sources

- https://wiki.openjdk.org/display/duke/Gallery[1]
- https://commons.wikimedia.org/wiki/File:Play_Framework_logo.svg
- https://commons.wikimedia.org/wiki/File:Scala-full-color.svg

[1] https://wiki.openjdk.org/display/duke/Gallery?preview=/http%3A%2F%2Fcr.openjdk.java.net%2F~jeff%2FDuke%2Fpng%2FJuggler.png

Preface

Welcome to *Developing Web Apps*!

This book shows how to develop Web applications with Play & Scala and with the Ports & Adapters pattern as well as DDD's tactical patterns.

The teaching aid that we are going to dissect shows how to implement a Web application's API with Play & Scala and its GUI with React & TypeScript, the focus being on the former and not the latter. Note that the resulting Web application (be it a self-contained system or a monolith) consists of one only deployment unit. This is reflected by there being only one Git/Docker[2] repository.

In order to also show how to implement a GUI without React (or one of its alternatives[3]), a small part of the GUI has been reimplemented twice, once with Play's Twirl in combination with htmx and once with Twirl only. For about a decade, using React & TypeScript for developing GUIs was a no-brainer to me and is still a great choice since both React and TypeScript are state-of-the-art pieces of technology that remain very popular (as the State of JavaScript[4] survey keeps confirming) for many good reasons. But the discovery of htmx[5] was somewhat of a revelation that even allowed for re-discovering Twirl.

MVP

This book is about **building the thing right**, not about **building the right thing**. Therefore, we will not dwell on how to come up with a product, let alone worry about how viable the product is. Check out *Product Management in Practice*[a] if you are interested in product management, too.

[a] https://www.oreilly.com/library/view/product-management-in/9781098119720/

Truth be told, I did have somewhat of a mid-life crisis and was on the verge of switching from Scala to Kotlin and from Play to a framework with Kotlin support (Spring Boot? Quarkus? Ktor?). But I still prefer Play over Spring Boot (which, unlike Quarkus and Ktor, I have experience with) and Scala over both Java and Kotlin.[6]

Enough about me. Let us talk about you for a minute.

[2] Familiarity with Git or Docker is not a prerequisite.
[3] https://2024.stateofjs.com/en-US/libraries/front-end-frameworks/
[4] https://2024.stateofjs.com/
[5] https://www.heise.de/news/JavaScript-htmx-2-0-verabschiedet-sich-vom-Internet-Explorer-9768803.html
[6] Plus, Scala keeps evolving.

Prerequisites

I am not going to teach Play (let alone Scala) to you. Alas, I have read too many books whose authors have bitten off more than they could chew by trying to explain anything & everything themselves. The parts which go to make Play are concisely and well explained in the official, freely accessible docs. There is no point in me reinventing the wheel. Instead, my intention is to show how to apply what you have learned by reading the Play docs (whether lazily[7], while going through the rest of this book, or eagerly[8], before going through the rest of this book) to a production-level, albeit simple, Web app.

In any case, I strongly recommend reading the Play docs (i.e., the *Main concepts for Scala*[9], the *Advanced topics for Scala*[10], and the *Common topics*[11]) "from cover to cover" eventually, both to solidify your Play knowledge and to learn of features you did not know existed but that might actually be of use to you.

In other words:

- You need to understand basic Web technologies such as HTML and HTTP. If that is not the case yet, start to *Learn web development*[12] and go back to the MDN project[13] whenever you are unfamiliar with or unsure about a piece of Web technology.
- You need to know Scala. If you are new to programming, study chapters 1 to 10 as well as 13, 15, and 16 of *Introduction to Programming and Problem Solving Using Scala*[14]. If you are only new to Scala, read *Programming in Scala*[15] or *Scala for the Impatient*[16].
- You need to be familiar with Play. If you are new to Play, take a quick look at its home page[17], skim through the *Getting Started*[18] section, and go through the tutorial[19] in order to gain a first impression of Play.

Depending on which GUI(s) you are interested in, you need to be familiar with React & TypeScript and/or with htmx:

[7] https://en.wikipedia.org/wiki/Lazy_evaluation
[8] https://en.wikipedia.org/wiki/Eager_evaluation
[9] https://www.playframework.com/documentation/latest/ScalaHome
[10] https://www.playframework.com/documentation/latest/ScalaAdvanced
[11] https://www.playframework.com/documentation/latest/Build
[12] https://developer.mozilla.org/en-US/docs/Learn
[13] https://developer.mozilla.org/en-US/
[14] https://www.programmingusingscala.net/home/introduction-to-programming-and-problem-solving-using-scala-2nd-edition
[15] https://www.artima.com/shop/programming_in_scala_5ed
[16] https://horstmann.com/scala/index.html
[17] https://www.playframework.com/
[18] https://www.playframework.com/documentation/latest/Introduction
[19] https://www.playframework.com/documentation/latest/HelloWorldTutorial

- If you are new to React & TypeScript, read *Learn React with TypeScript*[20].
- If you are new to htmx, read *Hypermedia Systems*[21].

It took me quite a while, but thanks to Scala, I finally learned to appreciate functional programming. Even if you were primarily interested in Java or Kotlin (or any other programming language, for that matter), I would strongly recommend studying *Functional and Concurrent Programming*[22] as well as *Functional Programming in Scala*[23]. (If you understand German, you could also study *Funktionale Programmierung in Java und Kotlin*[24].)

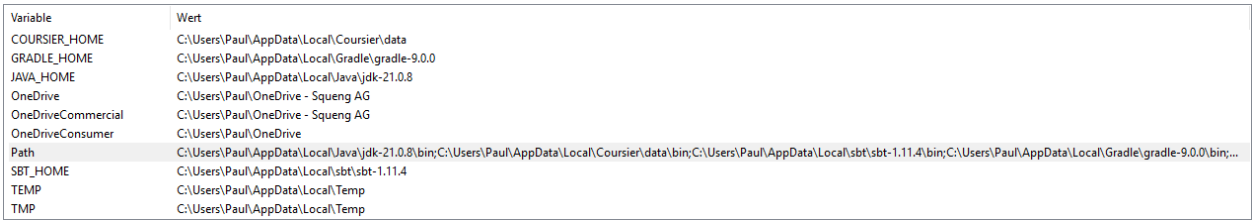
Tools of the Trade

- While Scala can target platforms (namely Scala JavaScript[25] and Scala Native[26]) other than the Java Virtual Machine (JVM), Play apps are JVM-based. So for any Play project, you need to have a Java Development Kit (JDK) installed. There are quite a few to choose from (and there is also GraalVM[27] to consider), including but not limited to
  - the canonical one from Oracle[28]
  - Azul Zulu[29]
  - Amazon Corretto[30]
  - Eclipse Temurin by Adoptium[31]
- To follow along with the examples in this book, you need to have sbt[32] installed. (You could use Gradle[33] or Mill[34] for your Play project, but I am using sbt for mine.)
- Even if you are not interested in the React/TypeScript-based GUI, you need to have Node.js[35] installed (which includes npm[36]) in order to run the teaching aid.
- Unless your app does not need server-side persistence, you need to have a database for development and test purposes (and unless my app really needs a relational DBMS such as PostgreSQL[37] or a graph DBMS such as Neo4j[38], I typically choose a MongoDB database, see the DBMS chapter), but you could run the teaching aid without one.

[20] https://www.packtpub.com/en-ch/product/learn-react-with-typescript-9781836643166
[21] https://hypermedia.systems/
[22] https://www.fcpbook.org/
[23] https://www.manning.com/books/functional-programming-in-scala-second-edition
[24] https://dpunkt.de/produkt/funktionale-programmierung-in-java-und-kotlin/
[25] https://www.scala-js.org/
[26] https://scala-native.org/
[27] https://www.graalvm.org/
[28] https://java.oracle.com/
[29] https://www.azul.com/downloads/
[30] https://aws.amazon.com/corretto/
[31] https://adoptium.net/
[32] https://www.scala-sbt.org/
[33] https://github.com/orgs/playframework/discussions/12338
[34] https://mill-build.org/mill/contrib/playlib.html
[35] https://nodejs.org/
[36] https://www.npmjs.com/
[37] https://www.postgresql.org/
[38] https://neo4j.com/

- You need to have an editor[39] (Visual Studio Code in my case, see below).

Instead of downloading and installing the JDK, sbt, etc. invidiually, you may want to manage them with SDKMAN![40] or Coursier[41].



Variable	Wert
COURSIER_HOME	C:\Users\Paul\AppData\Local\Coursier\data
GRADLE_HOME	C:\Users\Paul\AppData\Local\Gradle\gradle-9.0.0
JAVA_HOME	C:\Users\Paul\AppData\Local\Java\jdk-21.0.8
OneDrive	C:\Users\Paul\OneDrive - Squeng AG
OneDriveCommercial	C:\Users\Paul\OneDrive - Squeng AG
OneDriveConsumer	C:\Users\Paul\OneDrive
Path	C:\Users\Paul\AppData\Local\Java\jdk-21.0.8\bin;C:\Users\Paul\AppData\Local\Coursier\data\bin;C:\Users\Paul\AppData\Local\sbt\sbt-1.11.4\bin;C:\Users\Paul\AppData\Local\Gradle\gradle-9.0.0\bin;...
SBT_HOME	C:\Users\Paul\AppData\Local\sbt\sbt-1.11.4
TEMP	C:\Users\Paul\AppData\Local\Temp
TMP	C:\Users\Paul\AppData\Local\Temp

HOMEs

Make sure that `JAVA_HOME`, `COURSIER_HOME` or `SBT_HOME`, and `Path` are set correctly.

Visual Studio Code

At work, from about 2000 until 2007, TextPad[42] used to be not only my main editor but also my poor man's Java IDE (even though a friend of mine had told me about IntelliJ IDEA[43] when it was brand-new). Somewhere around 2007 or 2008, I kept TextPad as my main editor but switched to Eclipse[44] as my Java (and later also Scala) IDE. Somewhere around 2015 or 2016, I switched to Visual Studio Code[45] (VSC) as both my main editor and my main Java & Scala IDE. I guess I am a sucker for Erich Gamma[46] IDEs.

At home, my main editor used to be BBEdit[47]. I had bought a PowerBook G4 right after Apple's release of OS X even though—only weeks before—a then recent graudate of CMU's HCII[48] and now famous designer had explained to me why the new UI was terrible and doomed to fail. (But then, I had also bought Apple shares the summer before when—you know—Apple was dead and its acquisition of NeXTSTEP doomed to fail; if only I had held on to them instead of selling them with a 100% gain.) Eventually, Larry Page would play around with my Mac and start wondering whether to switch from PCs to Macs …

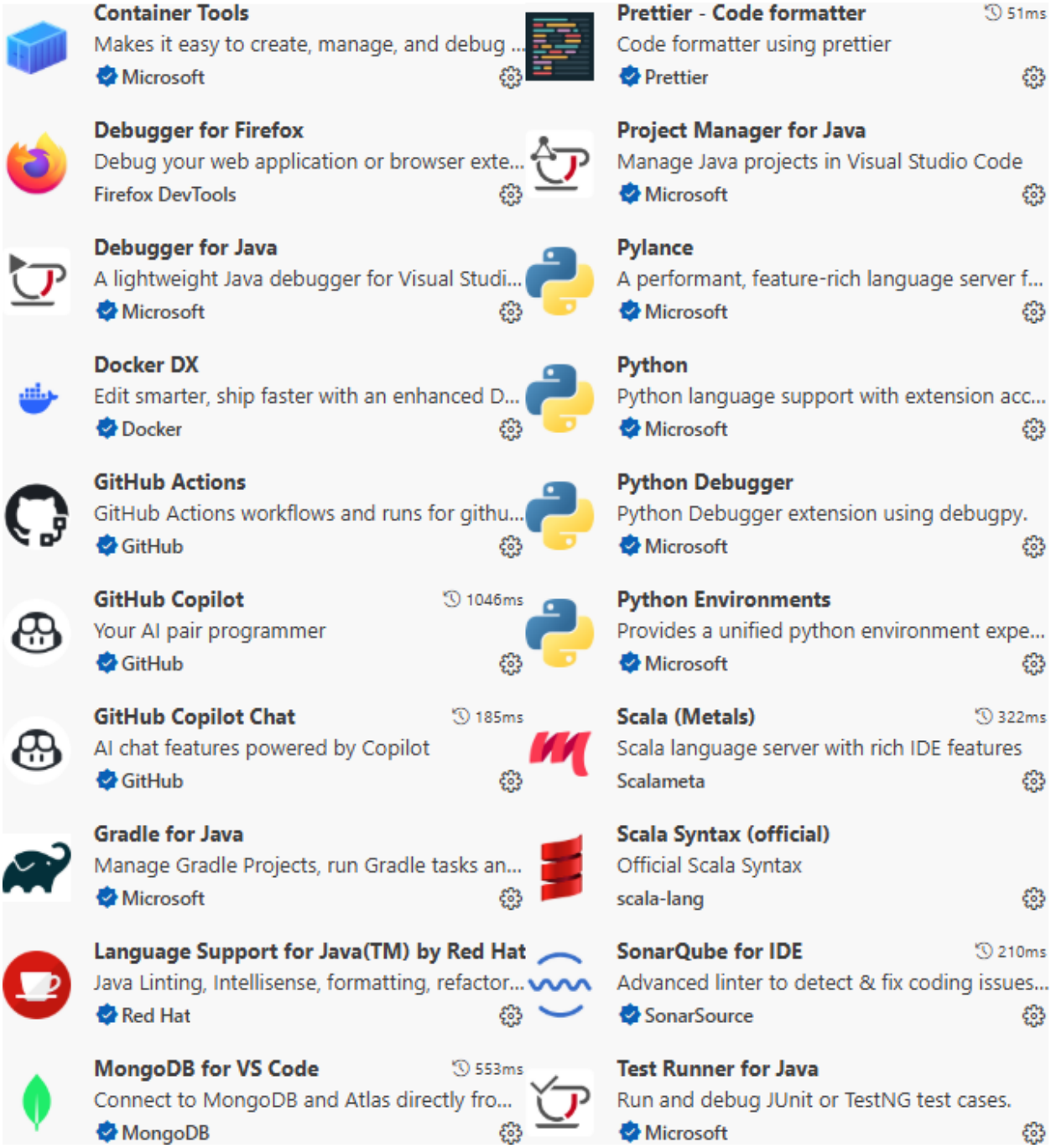
With the following extensions[49], VSC is a great IDE for both Java and Scala:

[39] https://scalameta.org/metals/docs/editors/overview.html
[40] https://sdkman.io/
[41] https://get-coursier.io/
[42] https://www.textpad.com/
[43] https://www.jetbrains.com/idea/
[44] https://eclipseide.org/
[45] https://code.visualstudio.com/
[46] https://github.com/egamma
[47] https://www.barebones.com/products/bbedit/
[48] https://www.hcii.cmu.edu/
[49] https://marketplace.visualstudio.com/VSCode

- Language Support for Java(TM) by Red Hat[50] (this and other extensions are also part of the Extension Pack for Java[51])
- Scala Syntax (offical)[52]
- Scala (Metals)[53]

There are also Java[54] and GraalVM Tools for Java[55] to consider.

[50] https://marketplace.visualstudio.com/items?itemName=redhat.java
[51] https://marketplace.visualstudio.com/items?itemName=vscjava.vscode-java-pack
[52] https://marketplace.visualstudio.com/items?itemName=scala-lang.scala
[53] https://marketplace.visualstudio.com/items?itemName=scalameta.metals
[54] https://marketplace.visualstudio.com/items?itemName=Oracle.oracle-java
[55] https://marketplace.visualstudio.com/items?itemName=oracle-labs-graalvm.graalvm

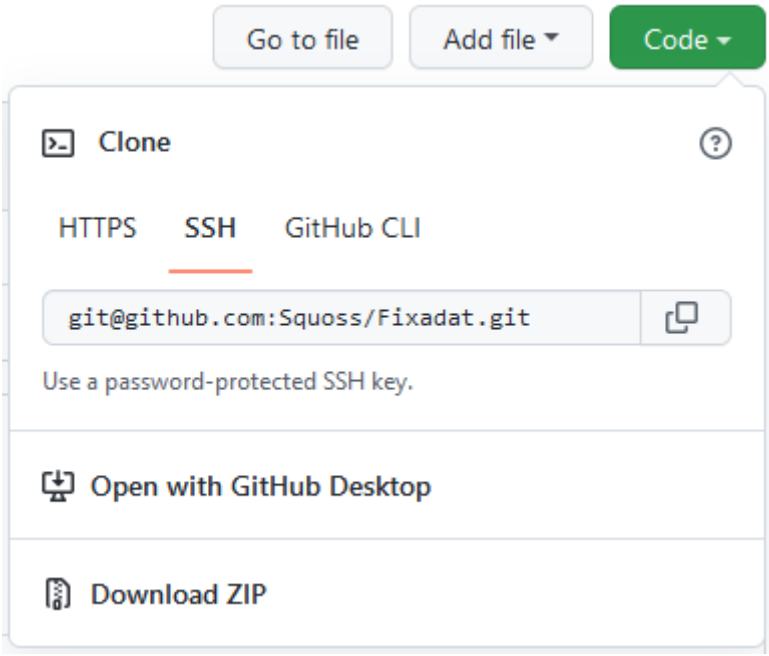


VSC extensions

Teaching Aid

Schedule like it's 200X

The Web app that serves as a teaching aid is Fixadat. Give it a try at https://fixadat.com/ (and feel free to use it for actually fixing dates & times). Once you got an idea of what Fixadat offers its users, head over to GitHub where its source code is published at https://github.com/Squoss/Fixadat and download the project folder as a ZIP file.



Download Fixadat

Extract the ZIP file, which results in a folder called `Fixadat-main`.

With your shell, execute the commands `sbt "run -Dconfig.file=conf/insecureLocalhost.conf"` in `Fixadat-main/beapi` and `npm install` as well as `npm start` in `Fixadat-main/fegui` to run the apps locally; yes, during development, the API and the GUI are two separate apps. We will see how two become one during deployment. With your browser[56], visit http://localhost:9000 and http://localhost:5173.

[56] I am using Firefox with the React Developer Tools extension, by the way.

Access Control

Fixadat requires authorization but does not require authentication. Unless we trust all users to be trustworthy and identify themselves truthfully, how can we achieve one without the other?

> *"The definitions of trust and trustworthy are often confused. The following example illustrates the difference: if an NSA employee is observed in a toilet stall at Baltimore Washington International airport selling key material to a Chinese diplomat, then (assuming his operation was not authorized) we can describe him as 'trusted but not trustworthy'. Hereafter, we'll use the NSA definition that a trusted system or component is one whose failure can break the security policy, while a trustworthy system or component is one that won't fail."*
> Ross Anderson, *Security Engineering*[57]

Capabilities

Even though Fixadat does not have any user accounts, we can still control access to resources by taking advantage of capability URLs[58]. In a capability URL such as `https://doodle.com/inturicogmbh`, the capability `inturicogmbh` must not only serve as an identifier and therefore be unique, but also be virtually unguessable[59].

We could generate the capabilities ourselves by careful, proper use of a cryptographic pseudorandom number generator[60], or we can leave the heavy lifting to Java's java.util.UUID.randomUUID()[61].

Unfortunately, there is an undeniable risk of exposure[62] with capability URLs. Simply moving the capability to the URLs query string would not mitigate the risk[63]. However, we can move it to the fragment identifier[64] and have the frontend pick it up[65] and provide it to the backend via a request header.

While we are at it, we can split the capability into a regular identifier (without security properties) and an access token, separating the two concerns: `https://fixadat.com/events/{EVENT_ID}#{ACCESS_TOKEN}`. This has the added benefit of allowing for revoking and re-issuing access tokens as well as for issuing tokens with differing access right.

If you like capabilities for access control, you may also like Macaroons[66].

[57] https://www.cl.cam.ac.uk/~rja14/book.html
[58] https://www.w3.org/TR/capability-urls/
[59] https://www.w3.org/TR/capability-urls/#capability-url-design
[60] https://en.wikipedia.org/wiki/Cryptographically_secure_pseudorandom_number_generator
[61] https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/util/UUID.html#randomUUID()
[62] https://www.w3.org/TR/capability-urls/#risk-of-exposure
[63] https://owasp.org/www-community/vulnerabilities/Information_exposure_through_query_strings_in_url
[64] https://developer.mozilla.org/en-US/docs/Web/API/Location/hash
[65] https://reactrouter.com/web/api/location
[66] https://www.manning.com/books/api-security-in-action

Identification and Authentication

I would not implement user accounts myself anymore (even though I seem to have got them right with respect to password storage[67] years before many major sites thanks to the first edition of Cryptography Engineering[68]). Instead, I would leave identity management[69] to the pros by falling back on an Identity as a Service (IDaaS) provider such as Zitadel[70], on an off-the-shelf Identity and Access Management (IAM) solution such as Keycloak[71], or on so-called social login[72] (either directly or via Zitadel or Keycloak).

[67] https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
[68] https://www.schneier.com/books/cryptography-engineering
[69] https://link.springer.com/book/10.1007/978-1-4842-8261-8
[70] https://zitadel.com/
[71] https://www.keycloak.org/
[72] https://en.wikipedia.org/wiki/Social_login

Getting Started: Hello, Play!

At this point, I assume that you have skimmed through the *Getting Started*[73] section and have gone through the tutorial[74].

If you have not done so yet, get familiar with Play's philosophy[75] as well.

[73] https://www.playframework.com/documentation/latest/Introduction
[74] https://www.playframework.com/documentation/latest/HelloWorldTutorial
[75] https://www.playframework.com/documentation/latest/Philosophy

Project Layout

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Outwith: Überproject

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Within: So Long, Layered MVC!

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

sbt Subproject

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

ArchUnit tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Monolith, Modulith, and Microservices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

New ...

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

... Repository

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Which came first: The chicken or the egg?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

... Application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Remove Clutter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

The Controllers are Dead, Long Live the Controllers!

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Amend .gitignore

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Dependency Injection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Controllers and Filters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Two Become One 1

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Dude, where's my API?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Prepare for Launch

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

GUI Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Bootstrap

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

React Router

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Getting htmx and Bootstrap as well as Bootstrap Icons

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

htmx

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Bootstrap

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Bootstrap Icons

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Taking Bootstrap and Bootstrap Icons for a Spin

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Security as a Forethought

If you have not done so yet, read *The Configuration API*[76] as well as *Configuration file syntax and features*[77], *Configuring the application secret*[78], and *Configuring the session cookie*[79].

As you certainly know, you cannot develop and deploy any Web app without considering security. And I mean *you* personally. Your organization also needs to seriously consider data protection and privacy[80] as those are a question of your organization's policy (vis-à-vis its customers, its products' users, etc.), some of which is enforced with security mechanisms.

Considering a Web app's security means considering aspects such as identification & authentication, authorization, software security, security software, and many more. This chapter is not about any of these aspects. This chapter is about hardening the app against some common Web threats that are virtually independent of your app and that services such as Mozilla Observatory[81] (bookmark it!) can partially measure.

If you are new to Web application security, peruse the OWASP[82] resources, in particular the OWASP Top Ten[83] and the OWASP Cheat Sheet Series[84].

Hardening a Web app is more of a backend responsibility, but the frontend is not completely off the hook.

Hardening the backend

As we shall see in the configuration chapter, the configuration for a Play app is found in its `application.conf` file (`Fixadat/beapi/conf/application.conf` in our case). Furthermore, Play provides various filters, most of which are security filters. Not all of Play's security mechanisms which allow for hardening a Play app are filters, but all can be configured.

When running Fixadat or SVPofWFTTC, you must see the following output (in any order):

[76] https://www.playframework.com/documentation/latest/ScalaConfig
[77] https://www.playframework.com/documentation/latest/ConfigFile
[78] https://www.playframework.com/documentation/latest/ApplicationSecret
[79] https://www.playframework.com/documentation/latest/SettingsSession
[80] http://williamstallings.com/Privacy/
[81] https://observatory.mozilla.org/
[82] https://owasp.org/
[83] https://owasp.org/www-project-top-ten/
[84] https://cheatsheetseries.owasp.org/

```
play.filters.csrf.CSRFFilter
play.filters.headers.SecurityHeadersFilter
play.filters.hosts.AllowedHostsFilter
play.filters.csp.CSPFilter
play.filters.https.RedirectHttpsFilter
```

This is necessary but not sufficient for Play to be configured securely.

The first three filters listed above are enabled by default; to make it explicit, include the key-value pair `play.http.filters = play.api.http.EnabledFilters`. The last two filters listed above must be enabled explicitly by adding the key-value pairs `play.filters.enabled += play.filters.csp.CSPFilter`, and `play.filters.enabled += play.filters.https.RedirectHttpsFilter`. Note that even in Fixadat's case, the CORS filter[85] is not enabled[86]; Fixadat is a self-contained system[87] whose API is not meant for third-party clients (yet).

> # Testing
>
> If the (default) filters interfere with (unit) tests, refer to
>
> - Testing Default Filters[a]
> - Testing CSRF[b]
> - Testing with CSRFFilter[c]
> - Testing[d]
> - Testing with AllowedHostsFilter[e]
>
> [a] https://www.playframework.com/documentation/latest/Filters#Testing-Default-Filters
> [b] https://www.playframework.com/documentation/latest/ScalaCsrf#Testing-CSRF
> [c] https://www.playframework.com/documentation/latest/Filters#Testing-with-CSRFFilter
> [d] https://www.playframework.com/documentation/latest/AllowedHostsFilter#Testing
> [e] https://www.playframework.com/documentation/latest/Filters#Testing-with-AllowedHostsFilter

Application Secret

Play requires an application secret, which defaults to `changeme`, which in turn would not be accepted in production as it would be insecure. In production, we are going to set it via an environment variable[88]. We are going to define the environment variable (named `APPLICATION_SECRET`) in the next part. Right now, we only need to add the key-value pair `play.http.secret.key = ${?APPLICATION_SECRET}` so that Play looks for it.

[85] https://www.playframework.com/documentation/latest/CorsFilter
[86] If your app needs to allow for Cross Origin Resource Sharing, heed OWASP's advice.
[87] https://scs-architecture.org/
[88] https://www.playframework.com/documentation/latest/ApplicationSecret#Environment-variables

Session Cookie

Unless you have really really good reasons (Do you really?) not to, you should harden all your app's cookies by setting `Secure`[89], `SameSite=Strict`[90], and `HttpOnly`[91]. You can configure Play to do so for the session cookie by adding the key-value pairs `play.http.session.secure = true`, `play.http.session.sameSite = "strict"`, and `play.http.session.httpOnly = true`.

Cross-Site Request Forgery (CSRF)

In order to prevent[92] CSRF attacks[93], a CSRF token must be included with certain HTTP requests. Remember that *"By default, Play will require a CSRF check when **all** of the following **are true**:"*

- The request method is not GET, HEAD or OPTIONS.
- The request has one or more Cookie or Authorization headers.
- The CORS filter is not configured to trust the request's origin.

The first requirement implies that an API better be RESTful[94]; more specifically, GET, HEAD, and OPTIONS requests must not have any side effects. The second requirement can be re-configured to protect all requests: add the key-value pair `play.filters.csrf.header.protectHeaders = null`. The third requirement can be re-configured to NOT trust CORS requests[95]: add the key-value pair `play.filters.csrf.bypassCorsTrustedOrigins = false`.

Since *"CSRF tokens should not be transmitted using cookies"*[96], we are going to use a custom request header[97]. In production, the backend is going to store the CSRF token in the DOM[98]. Therefore, the frontend must provide a placeholder in `Fixadat/fegui/public/index.html` and set the custom header[99] when it makes certain API calls. In order to test the replacement of the placeholder during development, add the line `<meta name="csrf-token" content="REPLACE_CSRF_TOKEN" />` to `Fixadat/beapi/public/index.html`'s `<head>` section. In order to actually replace the placeholder, overwrite the implementation of the `index()` method in `Fixadat/beapi/app/controllers/HomeController.scala` with the following one (and add `import play.filters.csrf.CSRF`):

[89] https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#secure-attribute
[90] https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#samesite-attribute
[91] https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#httponly-attribute
[92] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
[93] https://owasp.org/www-community/attacks/csrf
[94] https://dpunkt.de/produkt/rest-und-http-2/
[95] https://www.playframework.com/documentation/latest/ScalaCsrf#Trusting-CORS-requests
[96] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#synchronizer-token-pattern
[97] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#use-of-custom-request-headers
[98] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#storing-the-csrf-token-value-in-the-dom
[99] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#overriding-defaults-to-set-custom-header

```
  def index() = Action { implicit request: Request[AnyContent] =>
    val token =
      CSRF.getToken // // https://www.playframework.com/documentation/latest/ScalaCs\
rf#Getting-the-current-token
    Ok(string.replace("REPLACE_CSRF_TOKEN", token.get.value))
      .as("text/html")
  }
```

In production, Fixadat's frontend and backend will agree on what the value of the CSRF token is. During development, they will not. Instead, the backend can be configured to skip the CSRF check[100] when the value of the `Csrf-Token` header is `REPLACE_CSRF_TOKEN`. For obvious reasons, we do not want to add the key-value pair `play.filters.csrf.header.bypassHeaders.Csrf-Token = "REPLACE_CSRF_TOKEN"` to `application.conf`. Instead, add the following configuration as `insecureLocalhost.conf` (the inverse of a production configuration file[101], so to speak) to `Fixadat/beapi/conf` and start the backend locally with `[Fixadat] $ run -Dconfig.file=conf/insecureLocalhost.conf` from now on:

```
include "application"

play.filters.csrf.header.bypassHeaders.Csrf-Token = "REPLACE_CSRF_TOKEN"
```

Security Headers

Play supports various headers[102] to enhance security[103]. In two cases, we can be even stricter than Play's defaults[104] by adding the key-value pairs `play.filters.headers.permittedCrossDomainPolicies = "none"` and `play.filters.headers.referrerPolicy = "strict-origin-when-cross-origin"`. Futhermore, let us make it explicit that we do not allow for action-specific overrides[105] by adding the key-value pair `play.filters.headers.allowActionSpecificHeaders = false`.

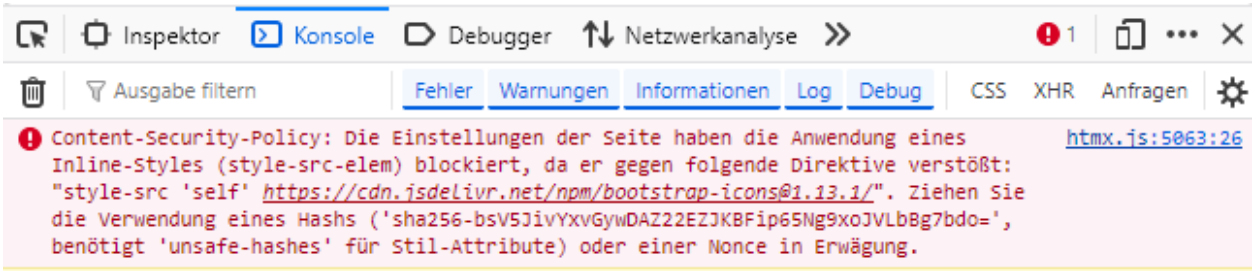
Content Security Policy (CSP)

Play features a dedicated CSP[106] filter. As we want to have a tight basic content-security policy[107] (namely, `Content-Security-Policy: default-src 'none'; script-src 'self'; connect-src 'self'; img-src 'self'; style-src 'self';`), add the following key-value pairs:

[100] https://www.playframework.com/documentation/latest/ScalaCsrf#Plays-CSRF-protection
[101] https://www.playframework.com/documentation/latest/ApplicationSecret#Production-configuration-file
[102] https://owasp.org/www-project-secure-headers/
[103] https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#http-headers-to-enhance-security
[104] https://www.playframework.com/documentation/latest/SecurityHeaders#Configuring-the-security-headers
[105] https://www.playframework.com/documentation/latest/SecurityHeaders#Action-specific-overrides
[106] https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
[107] https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html#basic-csp-policy

```
play.filters.csp.directives.default-src = "'none'"
play.filters.csp.directives.connect-src = "'self'"
play.filters.csp.directives.font-src = "'self'"
play.filters.csp.directives.img-src = "'self'"
play.filters.csp.directives.manifest-src = "'self'"
play.filters.csp.directives.script-src = "'self'"
play.filters.csp.directives.style-src = "'self'"
```

Furthermore, we want click-jacking to be prevented[108] by adding the key-value pair `play.filters.csp.directives.frame-ancestors = "'none'"`.



Hash as a Service

Such a tight CSP might be too tight in practice. Resist the temptation to simply allow `unsafe-eval`, `unsafe-inline`, etc. (their prefix is `unsafe` for a reason). Instead, loosen the policy in a controlled fashion and maybe even refactor bits and pieces of your app in order to avoid loosing it further; in SVPofWFTTC's case, for example, the `onchange="switchBootstrapTheme()"` attribute had to be removed from the `input` tag in `main.scala.html`and the line `document.getElementById('bsThemeSwitch').onchange = switchBootstrapTheme;` added to `main.js`.

Allowed Hosts

As you know[109], Play allows for limiting the hosts that can make requests by allow-listing those. We would like fixadat.com (including any and all sub-domains) and localhost to be allow-listed, which is why we add the key-value par `play.filters.hosts.allowed = [".fixadat.com", "localhost", ${?PAAS_DOMAIN}]`. The last entry in the array value allows for substituting the domain of the PaaS provider via an environment variable.

In the last part, we are going to allow-list `"fixadat.cleverapps.io"`. If we would like to allow-list further domains of our PaaS provider Clever Cloud, fixadat-test.cleverapps.io for instance, we have to allow-list them explicitly and must not allow-list cleverapps.io in general!

[108] https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html#preventing-clickjacking
[109] from https://www.playframework.com/documentation/latest/ScalaContentNegotiation#Language

Redirecting HTTP to HTTPS

In production, enforcing HTTP over TLS or simply HTTPS is a must. As you know[110], Fixadat further instructs browsers to switch to HTTPS even if the user entered only "http://" by taking advantage of Strict Transport Security[111]: `play.filters.https.strictTransportSecurity = "max-age=31536000; includeSubDomains"`

If you would like to already enforce HTTPS during development[112], add the key-value pair `play.filters.https.redirectEnabled = true`. If you would like to, you could configure a custom TLS certificate for the Play project[113] or even for your browser[114], but the latter would be akin to playing with fire, so beware!

Note that in production, determining whether a request was sent over TLS requires configuring trusted proxies[115], which we are going to do in the last part. That's because the TLS connection terminates at the edge and is not handled by Play[116] itself.

In the last part, we are also configuring our PaaS provider Clever Cloud to enforce HTTPS.

Force HTTPS

Hardening the frontend

Content Security Policy (CSP)

Even though the content security policy[117] (CSP) is configured at and served by the backend, there is something we have to do for it (or rather because of it) in the frontend. Since we want a CSP[118] that does **not** allow `'unsafe-inline'`[119] (let alone `'unsafe-eval'`), React needs to be configured not to generate any inline scripts. As you know[120], you need to add a file called `.env` to `Fixadat/fegui` and add the line `INLINE_RUNTIME_CHUNK=false` to it.

[110] from https://www.playframework.com/documentation/latest/RedirectHttpsFilter
[111] https://www.playframework.com/documentation/latest/RedirectHttpsFilter#Strict-Transport-Security
[112] https://www.playframework.com/documentation/latest/RedirectHttpsFilter#Enabling-the-HTTPS-filter
[113] (https://www.playframework.com/documentation/latest/ConfiguringHttps#SSL-Certificates-from-a-keystore)
[114] https://github.com/FiloSottile/mkcert
[115] https://www.playframework.com/documentation/latest/HTTPServer#Configuring-trusted-proxies
[116] https://www.playframework.com/documentation/latest/ConfiguringHttps#Production-usage-of-HTTPS
[117] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy
[118] https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
[119] https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/script-src
[120] from https://create-react-app.dev/docs/production-build/, https://create-react-app.dev/docs/adding-custom-environment-variables/, and https://create-react-app.dev/docs/advanced-configuration/

Cross-Site Request Forgery (CSRF)

In order to prevent[121] CSRF attacks[122], a CSRF token must be included with certain HTTP requests. Since *"CSRF tokens should not be transmitted using cookies"*[123], we are going to use a custom request header[124]. In production, the backend is going to store the CSRF token in the DOM[125]. Therefore, the frontend must provide a placeholder in `Fixadat/fegui/public/index.html` and set the custom header[126] when it makes certain API calls. For the former, add the line `<meta name="csrf-token" content="REPLACE_CSRF_TOKEN" />` to `Fixadat/fegui/public/index.html`'s `<head>` section. For the latter, add the following helper functions as `fetchJson.ts` (which is a mash-up between https://www.carlrippon.com/fetch-with-async-await-and-typescript/ and https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#xmlhttprequest-native-javascript) to `Fixadat/fegui/src`:

```
// https://www.carlrippon.com/fetch-with-async-await-and-typescript/

interface HttpResponse<T> extends Response {
  parsedBody?: T;
}
async function fetchJson<T>(request: Request): Promise<HttpResponse<T>> {
  // https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Preve\
ntion_Cheat_Sheet.html#xmlhttprequest-native-javascript
  if (!/^(GET|HEAD|OPTIONS)$/.test(request.method)) {
    const csrf_token = document.querySelector("meta[name='csrf-token']")!.getAttribu\
te("content");
    request.headers.append("Csrf-Token", csrf_token!);
  }

  const response: HttpResponse<T> = await fetch(request);
  try {
    response.parsedBody = await response.json();
  } catch (ex) { }

  if (!response.ok) {
    throw new Error(response.statusText);
  }
  return response;
```

[121] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
[122] https://owasp.org/www-community/attacks/csrf
[123] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#synchronizer-token-pattern
[124] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#use-of-custom-request-headers
[125] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#storing-the-csrf-token-value-in-the-dom
[126] https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#overriding-defaults-to-set-custom-header

```
}

export async function get<T>(path: string, accessToken: string, args: RequestInit = \
{ method: "get", mode: "same-origin", credentials: "same-origin", cache: "no-store",\
 redirect: "error", headers: { "X-Access-Token": accessToken } }): Promise<HttpRespo\
nse<T>> {
  return await fetchJson<T>(new Request(path, args));
}

export async function patch<T>(
  path: string,
  accessToken: string,
  body: any,
  args: RequestInit = { method: "PATCH", body: JSON.stringify(body), mode: "same-ori\
gin", credentials: "same-origin", cache: "no-store", redirect: "error", headers: { "\
Content-Type": "application/json", "X-Access-Token": accessToken } },
): Promise<HttpResponse<T>> {
  return await fetchJson<T>(new Request(path, args));
}

export async function post<T>(
  path: string,
  accessToken = "",
  body = {},
  args: RequestInit = { method: "POST", body: JSON.stringify(body), mode: "same-orig\
in", credentials: "same-origin", cache: "no-store", redirect: "error", headers: { "C\
ontent-Type": "application/json", "X-Access-Token": accessToken } },
): Promise<HttpResponse<T>> {
  return await fetchJson<T>(new Request(path, args));
}

export async function put<T>(
  path: string,
  accessToken: string,
  body: any,
  args: RequestInit = { method: "PUT", body: JSON.stringify(body), mode: "same-origi\
n", credentials: "same-origin", cache: "no-store", redirect: "error", headers: { "Co\
ntent-Type": "application/json", "X-Access-Token": accessToken } },
): Promise<HttpResponse<T>> {
  return await fetchJson<T>(new Request(path, args));
}
```

Cross-Site Scripting (XSS)

In order to prevent[127] Stored or Reflected XSS attacks[128] and to prevent[129] DOM-based attacks[130], the GUI's content must be properly escaped. Luckily, React takes care of escaping[131]. If you really really have to circumvent React escaping[132] some content (Do you really?), do yourself a favor and at least avoid any user-generated content as well as content consumed from third parties (e.g., via their APIs)!

What about TLS?

In production, enforcing HTTP over TLS or simply HTTPS is a must. If you already enable or even enforce HTTPS in the backend during development, you may want to do so in the frontend as well. Simply add the line `HTTPS=true` to `Fixadat/fegui/.env`. If you would like to, you could configure a custom TLS certificate for your frontend project[a] or even for your browser[b], but the latter would be akin to playing with fire, so beware!

[a] (https://create-react-app.dev/docs/using-https-in-development/#custom-ssl-certificate)
[b] https://github.com/FiloSottile/mkcert

Rinse & Repeat

For as long as your project is deployed, you will not be done with security; you need to strive for continuous security[133]. With respect to this chapter, you need to answer at least two questions on a regular basis (e.g., at the end of a Sprint[134] before you deliver the Increment[135] if you happen to employ Scrum with Essence[136] (or without)):

1. Do I need to update Play[137] or another dependency[138] because of a newly discovered security vulnerability?
2. Do I need to update React[139] or another package[140] because of a newly discovered security vulnerability?

[127] https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
[128] https://owasp.org/www-community/attacks/xss/
[129] https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html
[130] https://owasp.org/www-community/attacks/DOM_Based_XSS
[131] https://reactjs.org/docs/introducing-jsx.html#jsx-prevents-injection-attacks
[132] https://reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml
[133] https://www.manning.com/books/securing-devops
[134] https://www.scrumguides.org/scrum-guide.html#the-sprint
[135] https://www.scrumguides.org/scrum-guide.html#increment
[136] https://www.scruminc.com/better-scrum-with-essence/
[137] https://www.playframework.com/documentation/latest/Migration28
[138] https://github.com/albuch/sbt-dependency-check
[139] https://create-react-app.dev/docs/updating-to-new-releases
[140] https://docs.npmjs.com/cli/v6/commands/npm-audit

3. Are the steps taken above still necessary[141] and sufficient?

As for 1 and 2, you could automate some parts within your repository[142] and/or pipeline[143] and/or …

[141] https://infosec.mozilla.org/guidelines/web_security
[142] https://docs.github.com/en/free-pro-team@latest/github/managing-security-vulnerabilities
[143] https://support.snyk.io/hc/en-us/sections/360001152577-CI-CD-integrations

Database Management System

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

MongoDB

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Event Sourcing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

MongoDB

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Driver

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Connection helper

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Dependency Injection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Internationalization and Localization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Both Fixadat and SVPofWFTTC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Supported Locales

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Switching the Locale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Only Fixadat: Backend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Serving the Localizations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Only Fixadat: Frontend

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Using a Localization Context

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Fetching and Providing the Localizations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Using the Localizations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Switching the Locale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

CI/CD

Like the previous part and unlike the next part, this entire part is pretty much independent of the domain.

Two Become One 2

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Alternatives

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Building Images Locally

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Running Containers Locally

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Continuous Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Test

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Scan

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Continuous Delivery

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Pull

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Push

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Cleaning up & out

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Continuous Deployment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

MongoDB

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Docker or Java

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Pull

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Push

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Cleaning up & out

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Clever Cloud

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

GitHub

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Domains

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Features

Going Public

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

index.html

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Records and Case Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Primitive Types

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Unsigned Integers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Interaction Design

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Users and Interactions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Hosts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Guests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Boilerplate

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Configuration Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Application Secret

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Session Cookie

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Logging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Template

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Boilerplate

Privacy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Legalese

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/DevWebApps.

Appendix

Updating and Upgrading

At the very least for security reasons, you need to update and eventually upgrade your projects on a regular basis.

Updating the Backend

- update `FROM hseeberger/scala-sbt:JAVA_SBT_SCALA as play` in `Fixadat/Dockerfile`
- follow the migration instructions[144] if need be
- update `scalaVersion` in `Fixadat/beapi/build.sbt` as well as in `Fixadat/beapi/hexagon/build.sbt`
- update `sbt.version` in `Fixadat/beapi/project/build.properties` as well as in `Fixadat/beapi/hexagon/project/build.properties`
- update `addSbtPlugin("com.typesafe.play" % "sbt-plugin" % "X.Y.Z")` in `Fixadat/beapi/project/plugins.sbt`
- update third-party dependencies in both `Fixadat/beapi/build.sbt` and `Fixadat/beapi/hexagon/build.sbt`

Updating the Frontend

- follow the migration instructions[145] if need be
- with your shell, execute the command `npm update` in `Fixadat/fegui`
- with your shell, execute the command `npm audit fix` in `Fixadat/fegui`

Upgrading the Backend

- upgrade `FROM openjdk:JAVA` as well as `FROM hseeberger/scala-sbt:JAVA_SBT_SCALA as play` in `Fixadat/Dockerfile`
- follow the migration instructions[146]
- to be continued

Upgrading the Frontend

- upgrade `FROM node:JS as react` in `Fixadat/Dockerfile`
- follow the migration instructions[147]
- to be continued

[144] https://www.playframework.com/documentation/latest/Migration28
[145] https://github.com/facebook/create-react-app/blob/main/CHANGELOG.md
[146] https://www.playframework.com/documentation/latest/Migration28
[147] https://github.com/facebook/create-react-app/blob/main/CHANGELOG.md