Building server-side web applications with htmx and Spring

Devoxx Belgium 2024 – 2024-10-07

Thomas Schühly

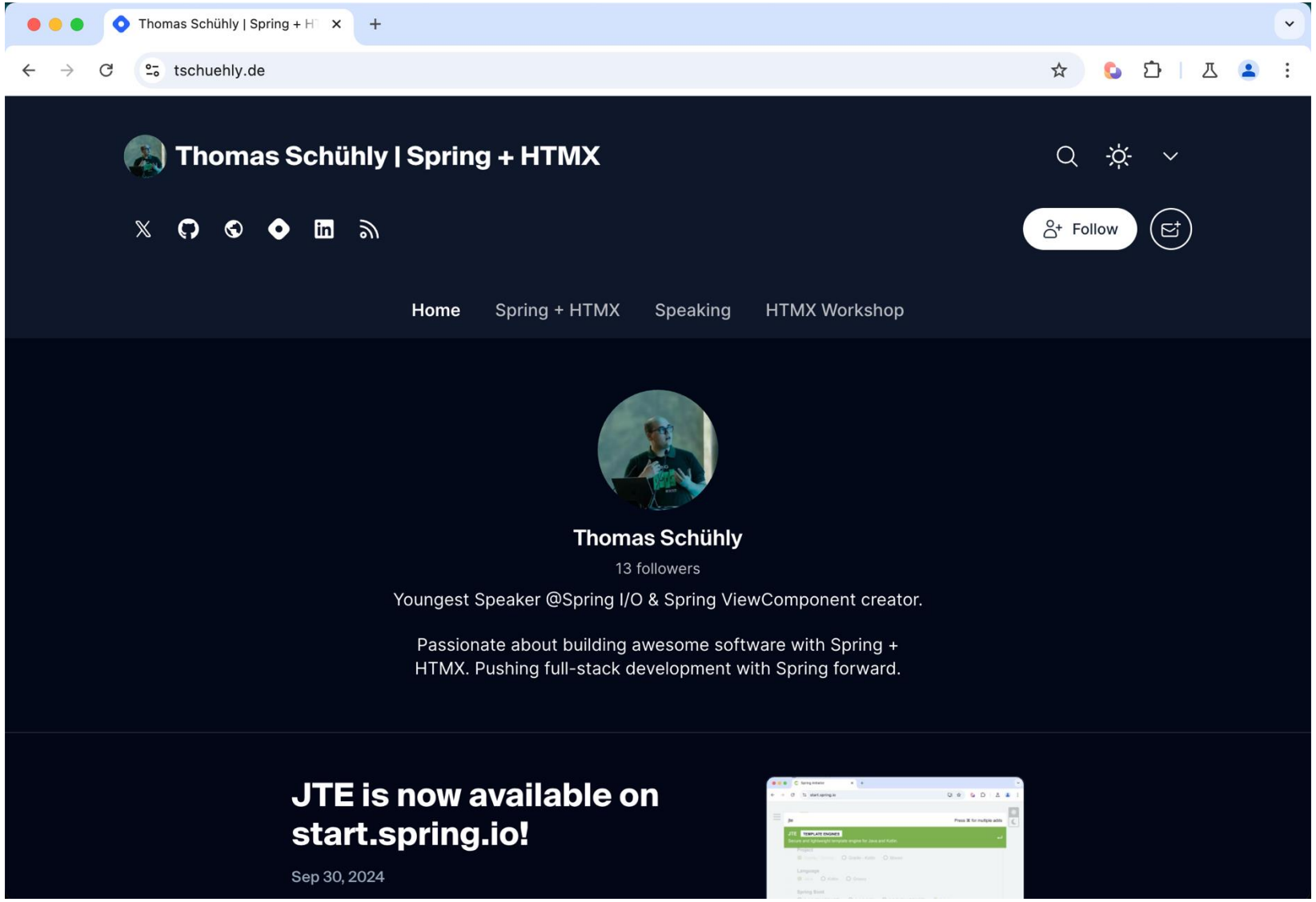
SETUP:

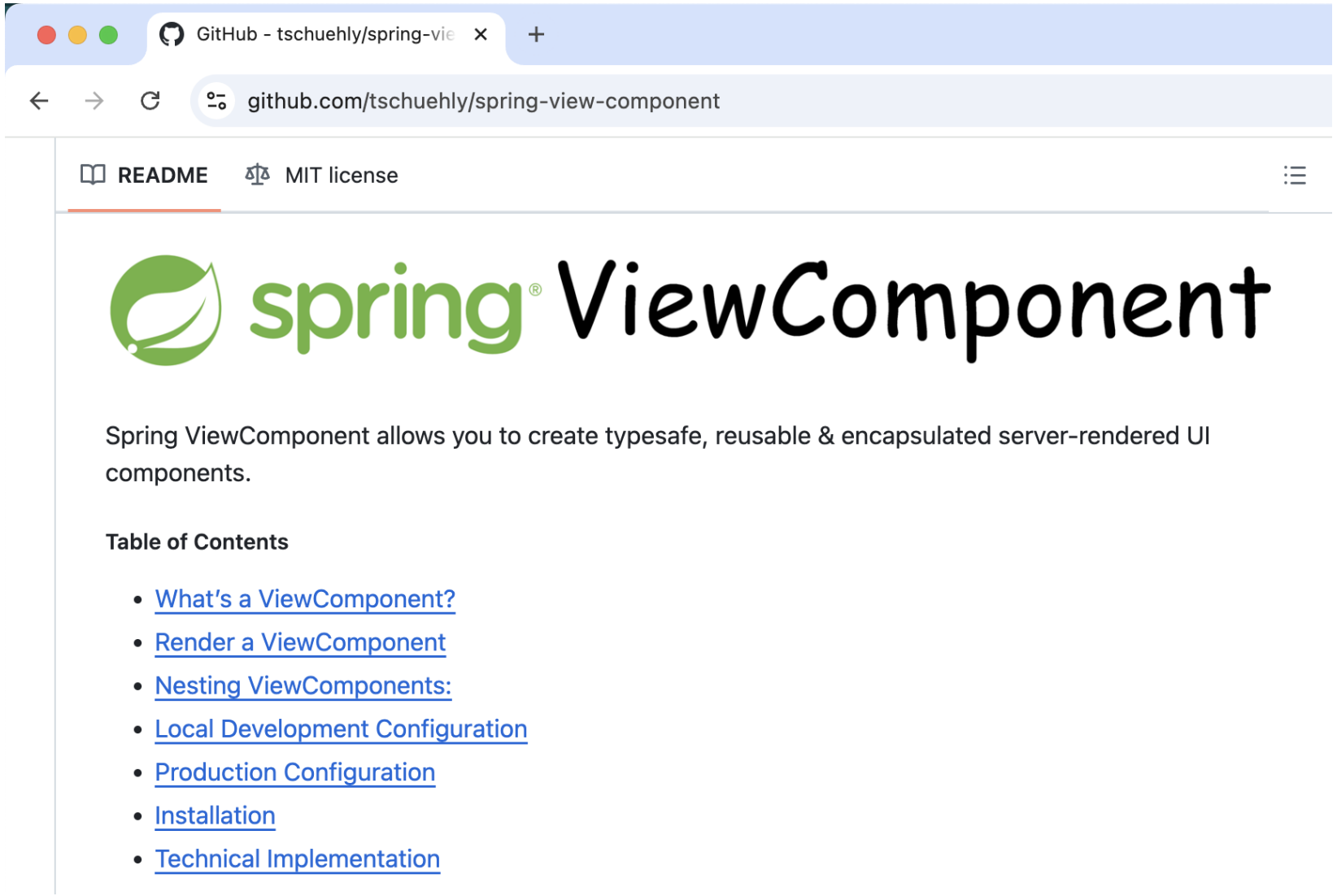
htmx.tschuehly.de

Thomas Schühly

- Spring Boot + HTMX = ❤️
- Speaker: @Spring IO, @JavaZone, @JUG
- Software Engineer @alanda.io

My blog

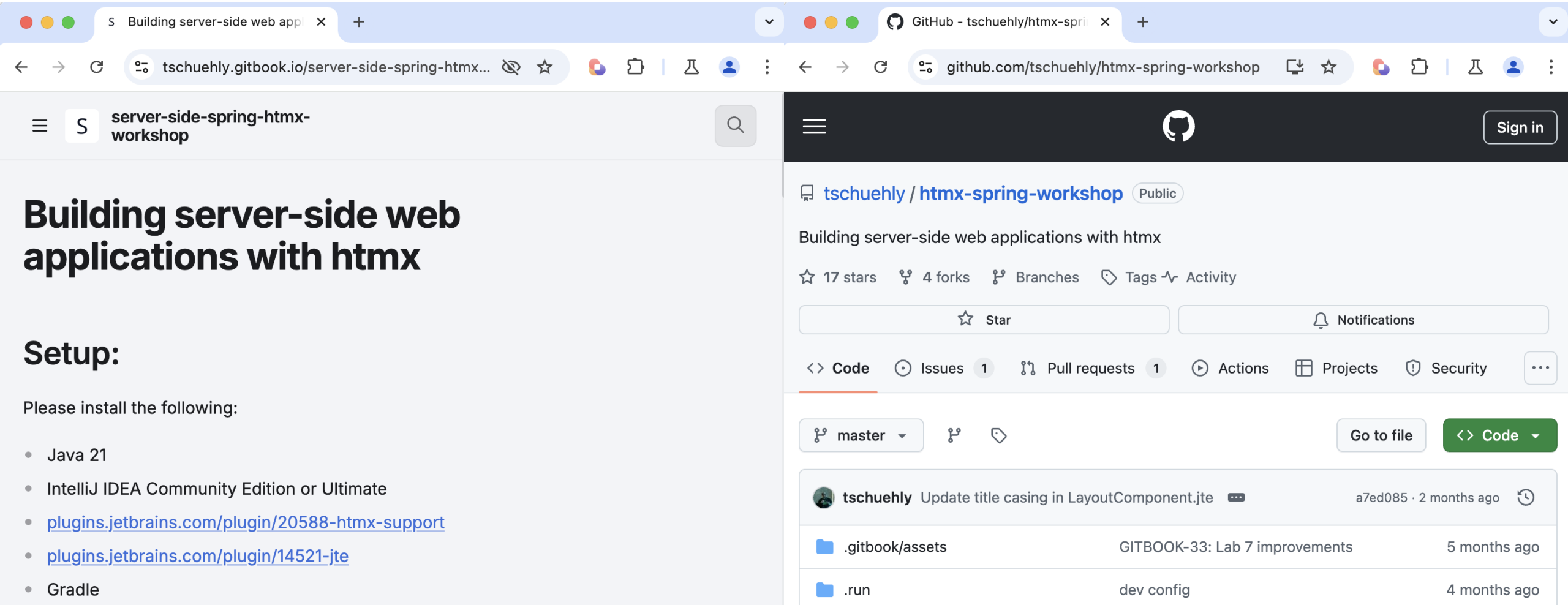

Thomas Schühly | Spring + HTMX
tschuehly.de
Thomas Schühly | Spring + HTMX
Follow
Home
Spring + HTMX
Speaking
HTMX Workshop
Thomas Schühly
13 followers
Youngest Speaker @Spring I/O & Spring ViewComponent creator.
Passionate about building awesome software with Spring + HTMX. Pushing full-stack development with Spring forward.
JTE is now available on start.spring.io!
Sep 30, 2024



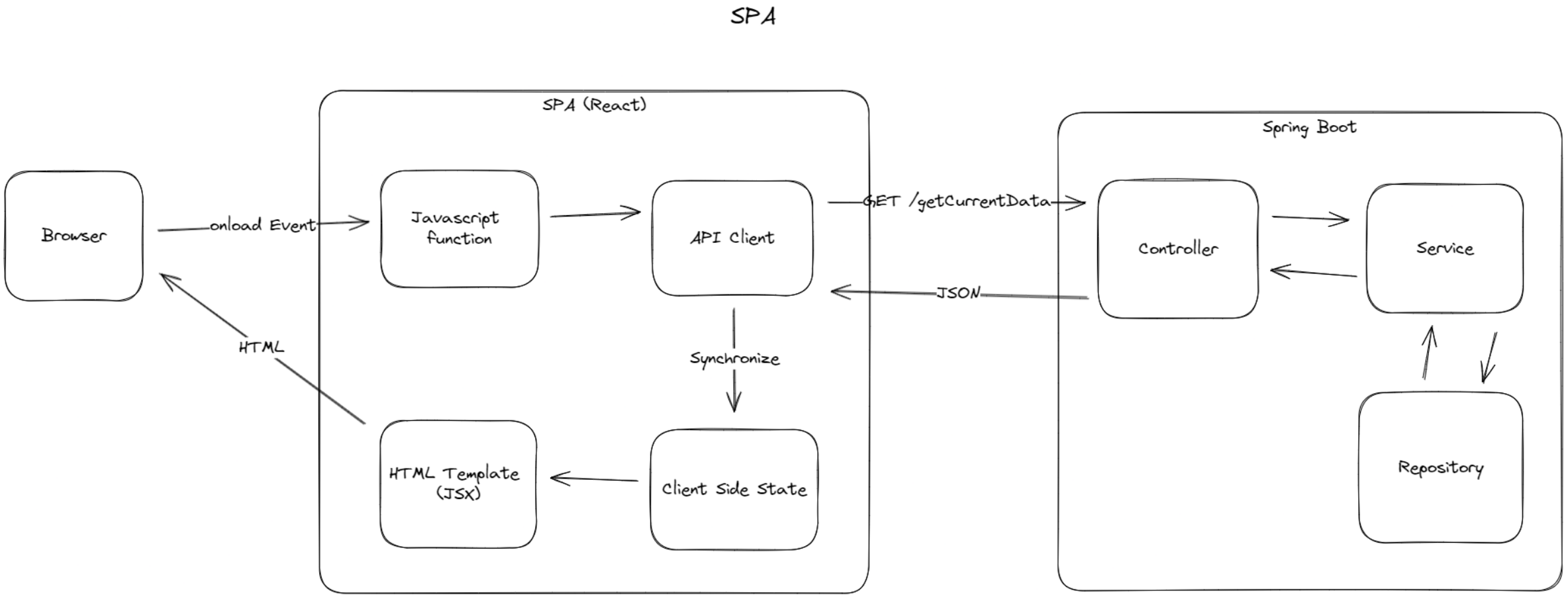
Star it on GitHub!

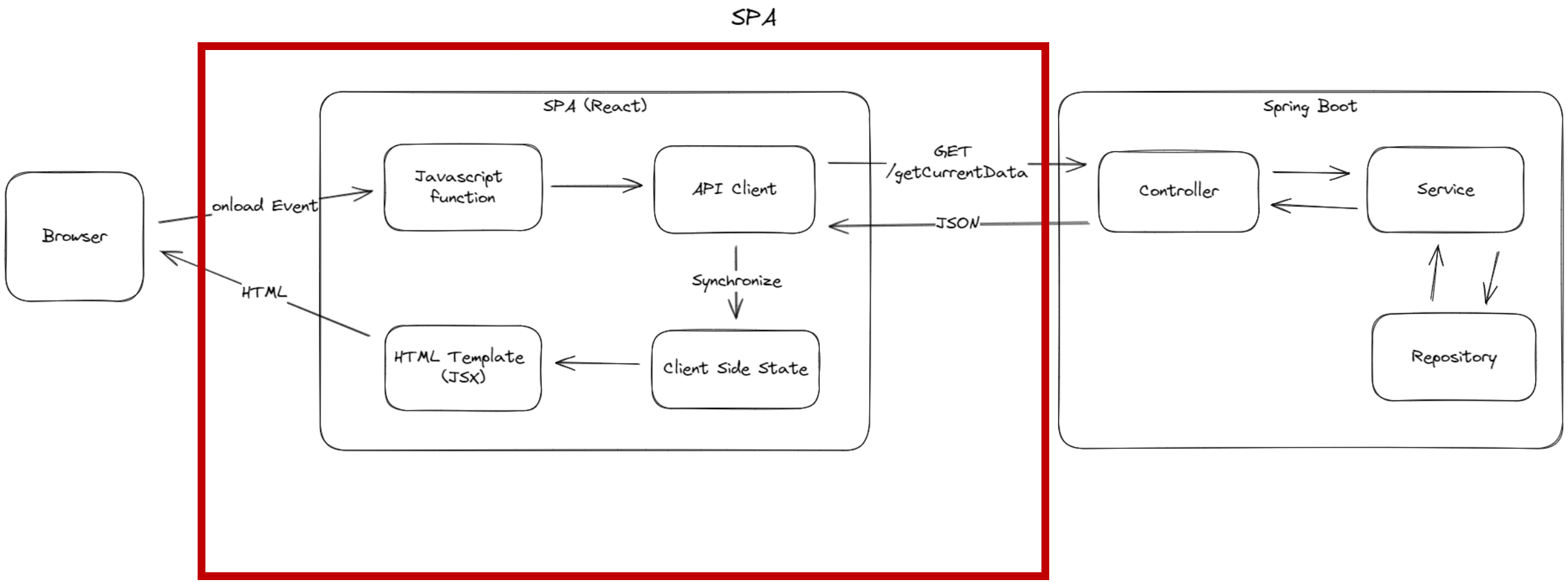
Setup Time!

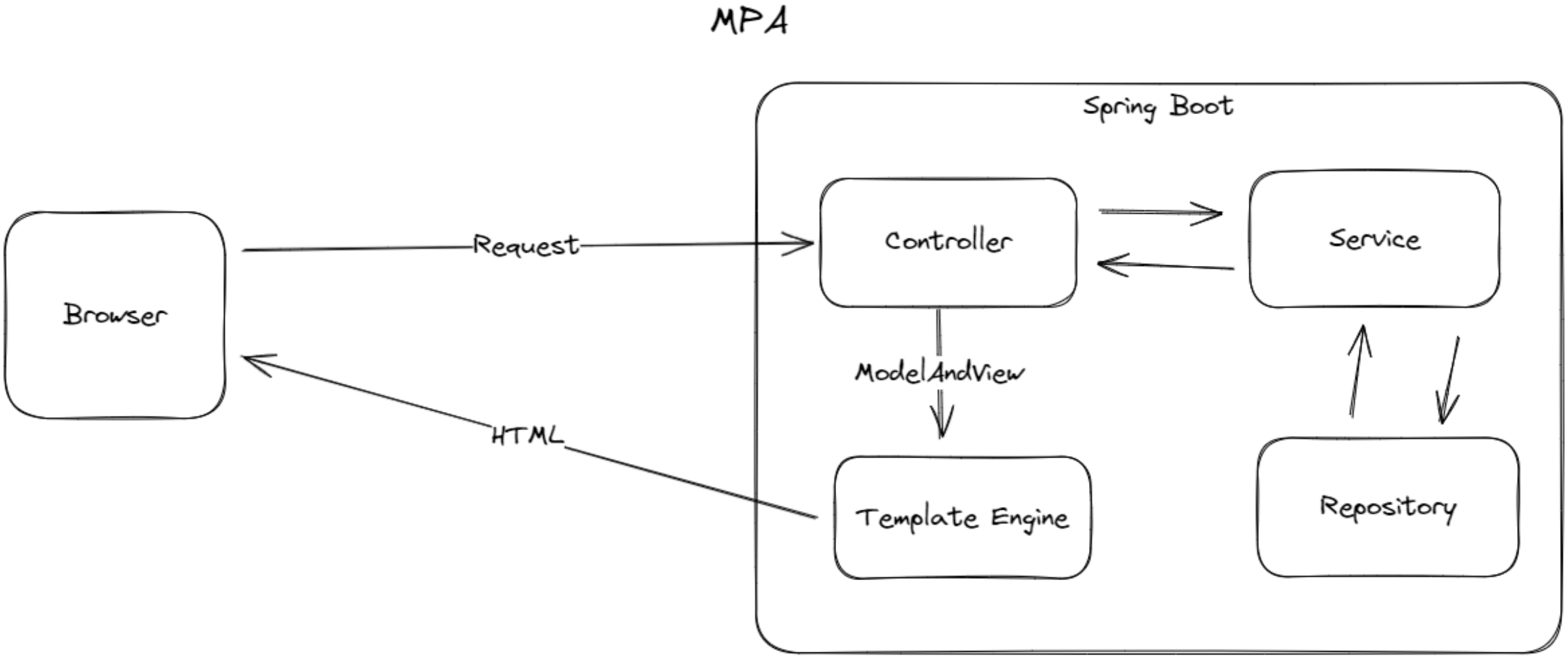
htmx.tschuehly.de

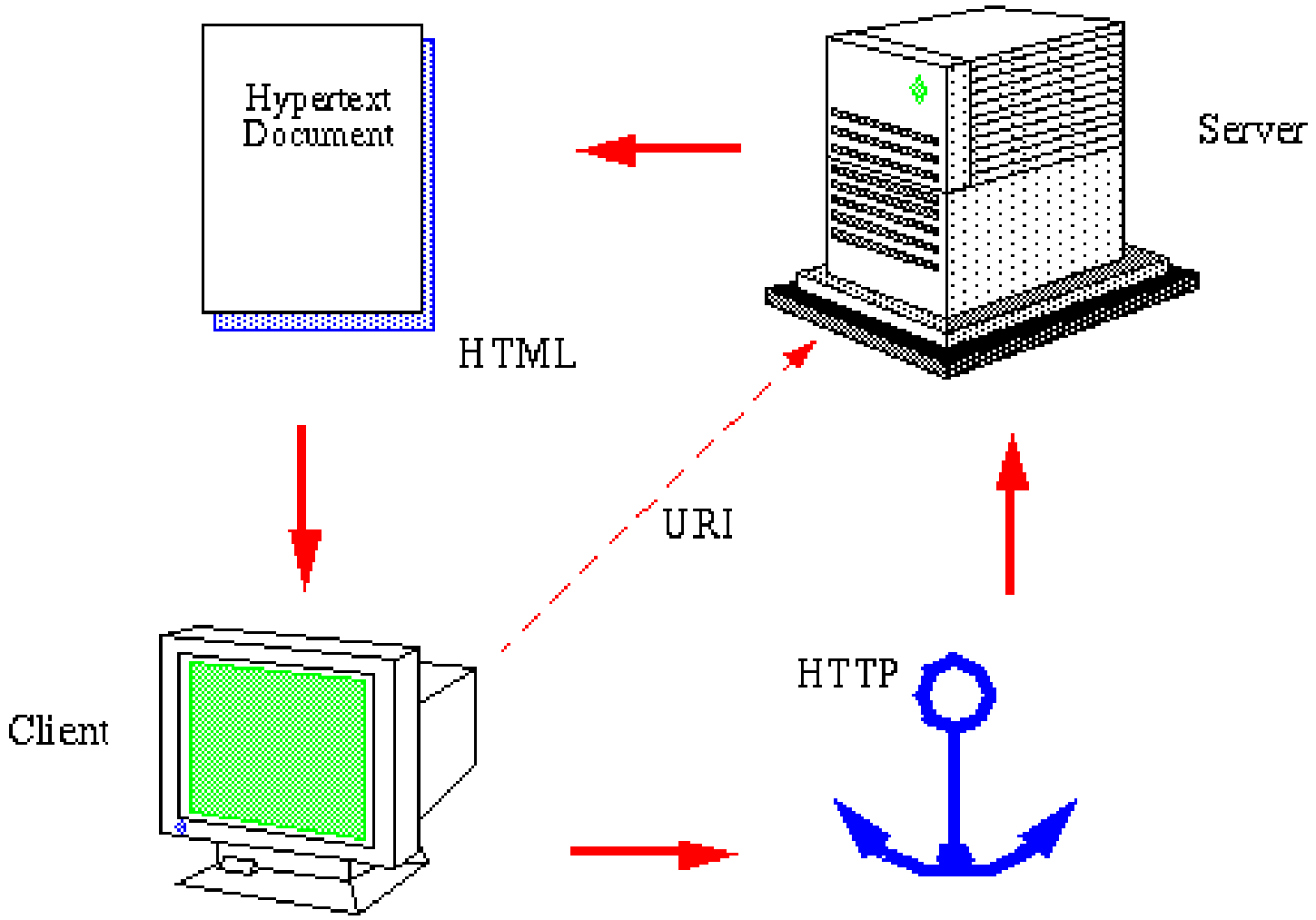


Why?


SPA
SPA (React)
Spring Boot
Browser
onload Event
Javascript function
API Client
GET /getCurrentData
Controller
Service
JSON
HTML
Synchronize
HTML Template (JSX)
Client Side State
Repository

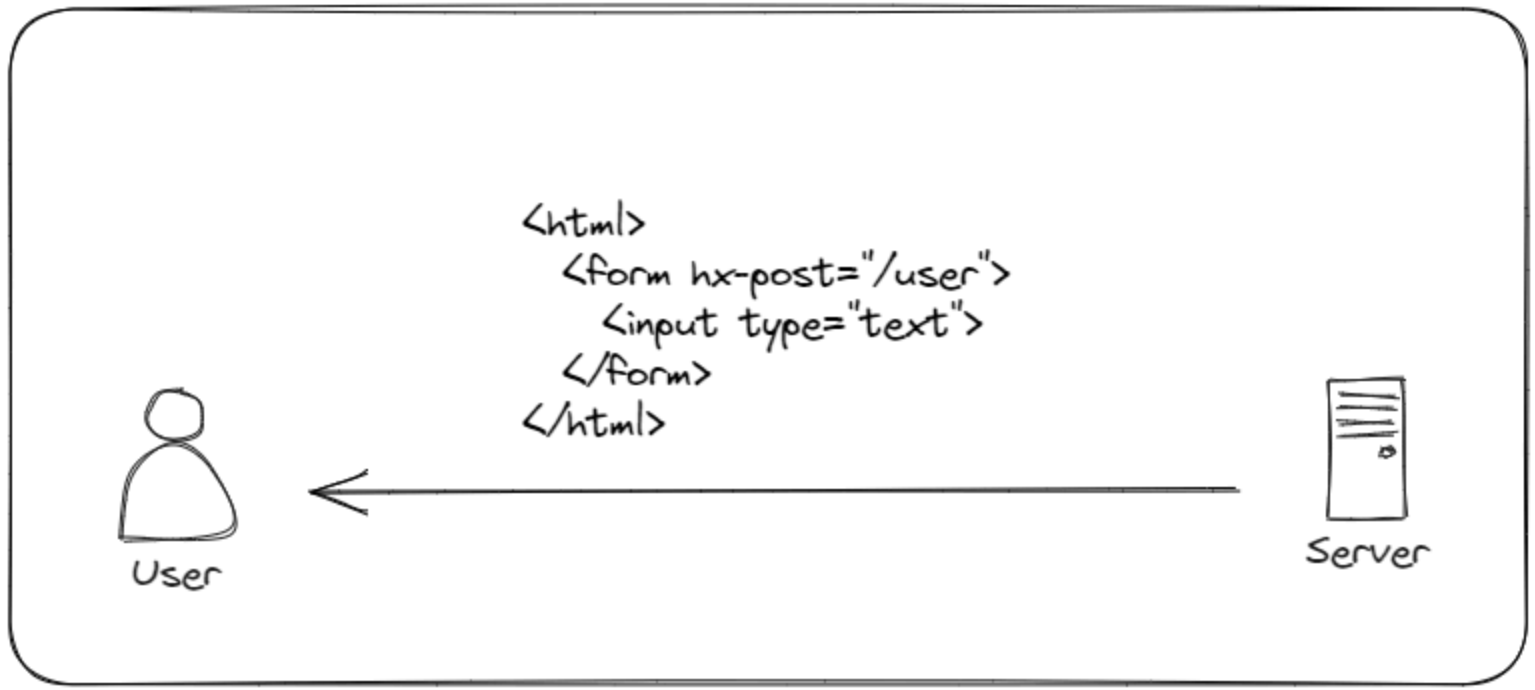

SPA
SPA (React)
Spring Boot
Browser
onload Event
HTML
Javascript function
API Client
GET /getCurrentData
JSON
Synchronize
HTML Template (JSX)
Client Side State
Controller
Service
Repository

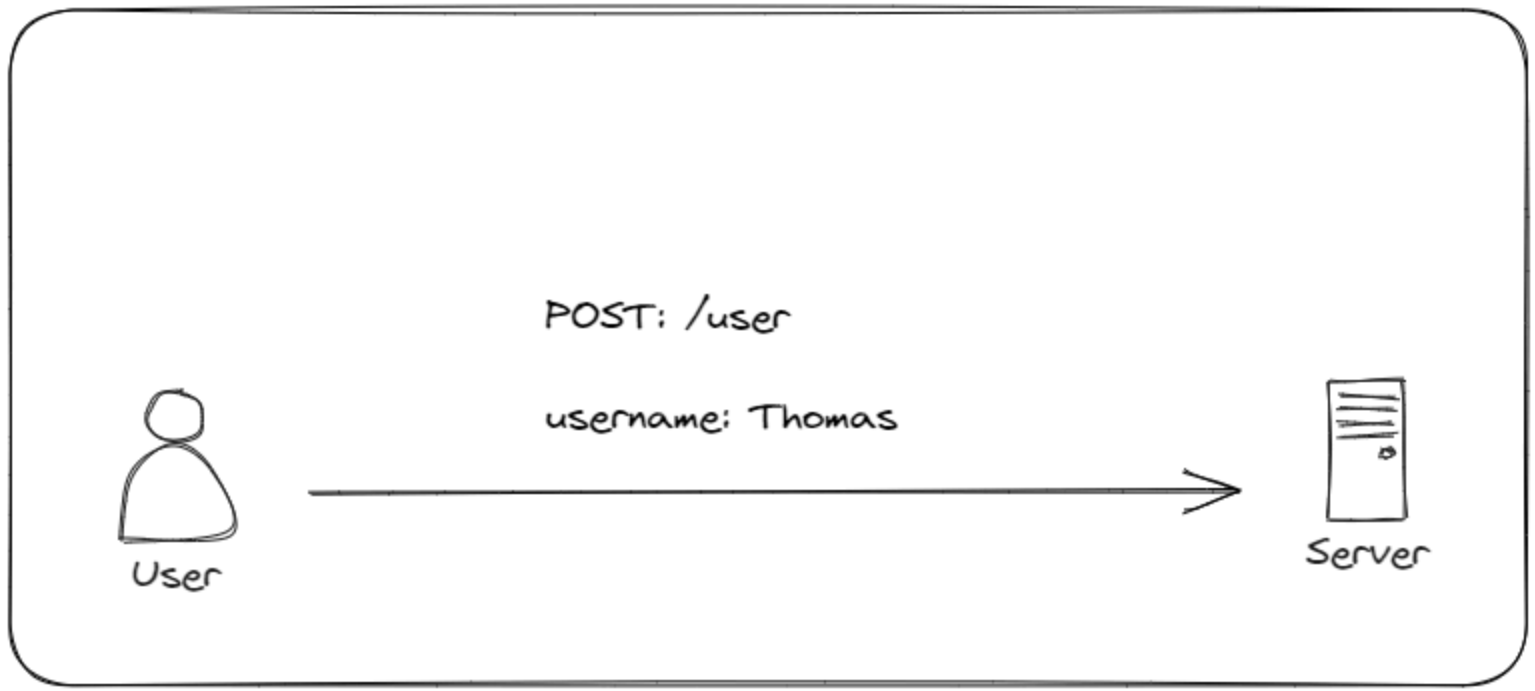

MPA
Spring Boot
Browser
Request
Controller
Service
ModelAndView
HTML
Template Engine
Repository

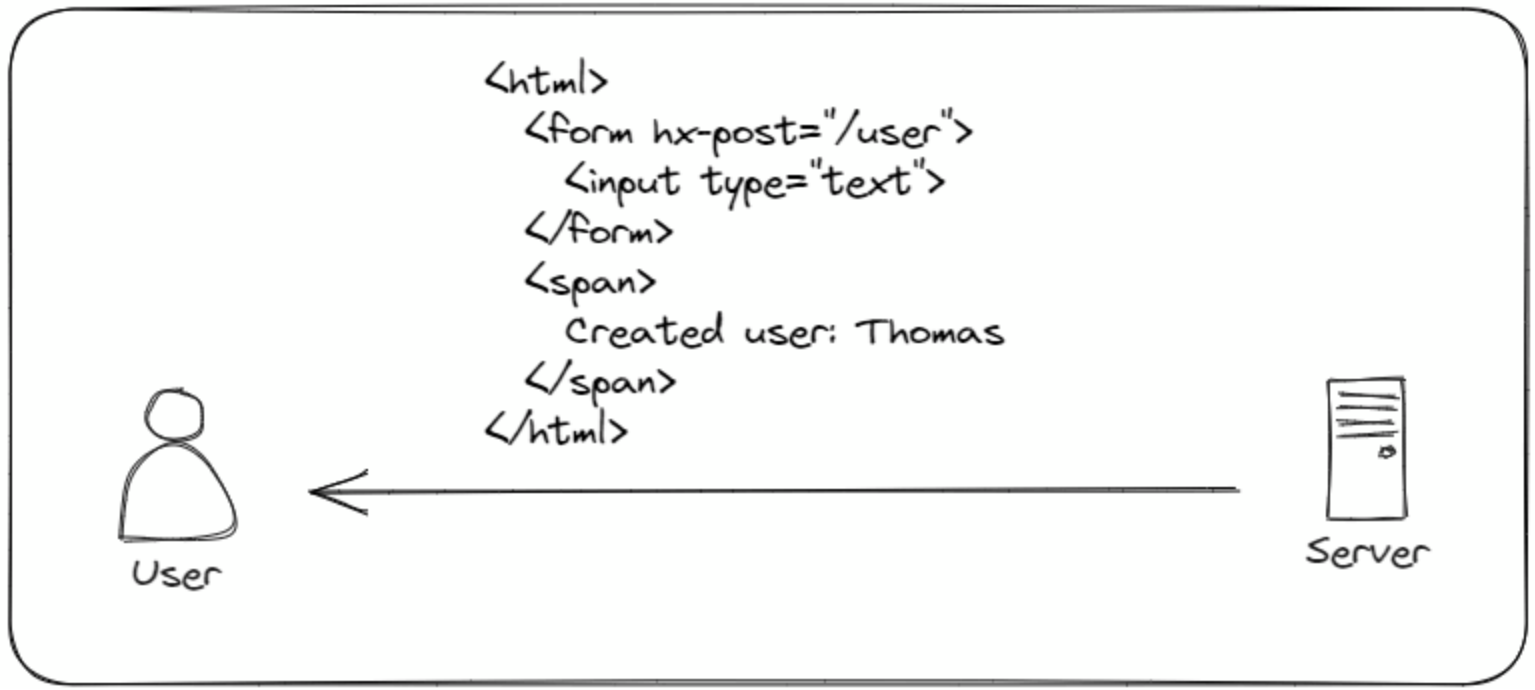


https://htmx.org/essays/hateoas/


GET /userform
User
Server


<html>
<form hx-post="/user">
<input type="text">
</form>
</html>
User
Server


POST: /user
username: Thomas
User
Server


<html>
<form hx-post="/user">
<input type="text">
</form>

Created user: Thomas

</html>
User
Server


htmx
high power tools for HTML

```
<a href="/blog">Blog</a>
```


<button hx-post="/clicked"
hx-trigger="click"
hx-target="#parent-div"
hx-swap="outerHTML" >
Click Me!
</button>

haiku

javascript fatigue:
longing for a hypertext
already in hand

Benefits

- Hypermedia allows your application API to be much more aggressively refactored and optimized
- Hypermedia is often significantly less complex than an SPA approach would be for many problems

https://htmx.org/essays/when-to-use-hypermedia/

Our Speakers

Talk

David Guillot

From React to htmx on a real-world SaaS product: we did it, and it's awesome!

2022 djangocon.eu porto

SEPTEMBER 21ST-25TH

htmx.org/essays/a-real-world-react-to-htmx-port/

Transitional Applications

- Adopting Hypermedia is not an either/or decision
- Use the right tool for the right use case

https://htmx.org/essays/when-to-use-hypermedia/

PhotoQuest

Not Secure photoquest.local:8080/onboarding/karten

PhotoQuest

Vorne | Hinten

Generate

1 **Gestalte deine Karten**

Diese Karten erhalten deine Gäste auf der Hochzeit

Hochzeitsdatum

13.10.2024

Erster Partner

Name eingeben | Weiblich

Zweiter Partner

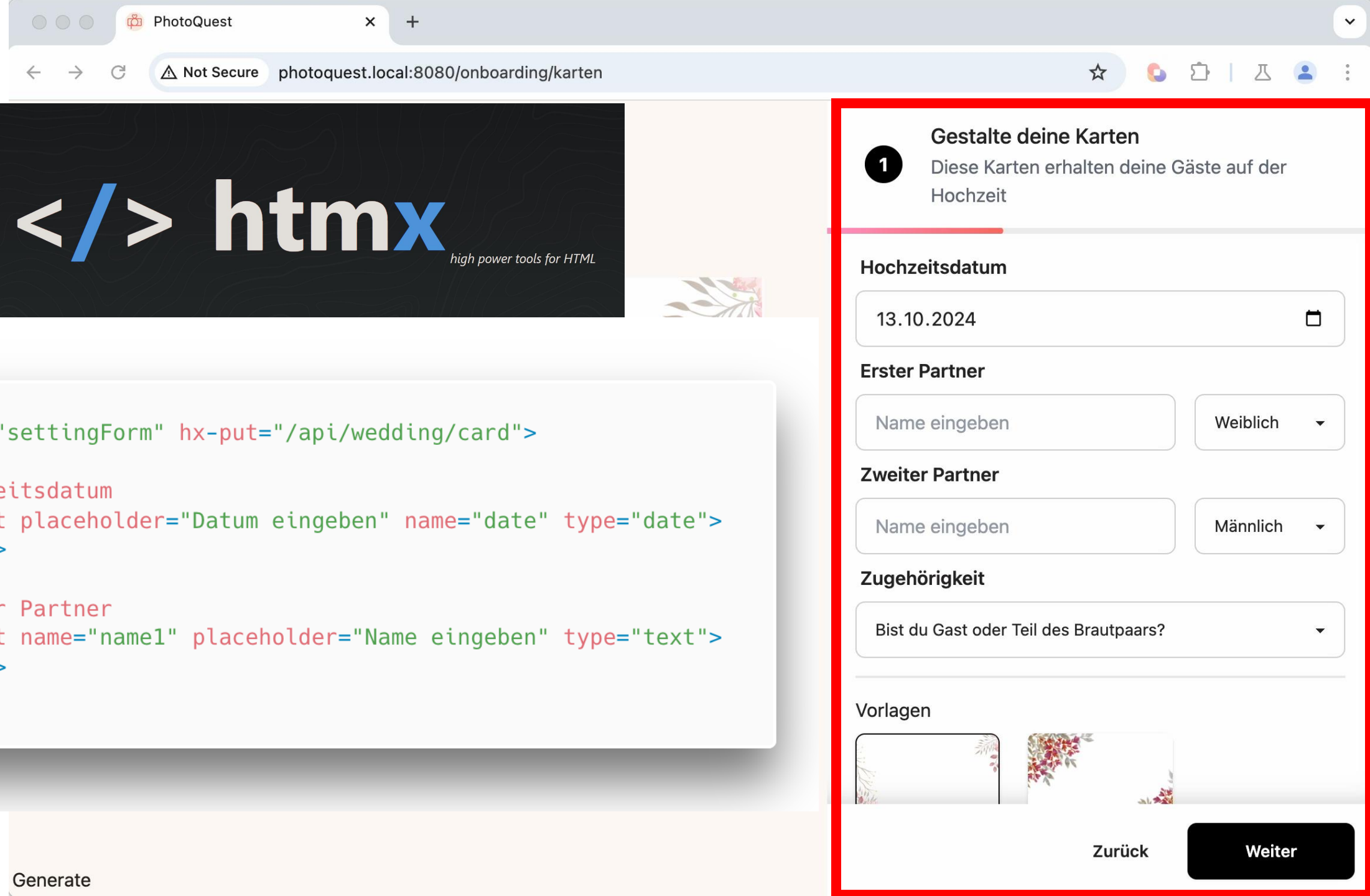
Name eingeben | Männlich

Zugehörigkeit

Bist du Gast oder Teil des Brautpaars?

Vorlagen

Zurück | Weiter




```
<form id="settingForm" hx-put="/api/wedding/card">
  <label>
    Hochzeitsdatum
    <input placeholder="Datum eingeben" name="date" type="date">
  </label>
  <label>
    Erster Partner
    <input name="name1" placeholder="Name eingeben" type="text">
  </label>
</form>
```

Generate

PhotoQuest

Not Secure photoquest.local:8080/onboarding/karten

PhotoQuest

Vorne Hinten

Generate

1 Gestalte deine Karten

Diese Karten erhalten deine Gäste auf der

pdfme: Free and Open source PDF generation library!

A powerful PDF generation library fully written in TypeScript, featuring a React-based UI template editor for seamless PDF creation.

Zurück Weiter

PhotoQuest

Not Secure photoquest.local:8080/onboarding/karten

PhotoQuest

Vorne Hinten

Generate

1 Gestalte deine Karten

Diese Karten erhalten deine Gäste auf der

pdfme: Free and Open source PDF generation library!

A powerful PDF generation library fully written in TypeScript, featuring a React-based UI template editor for seamless PDF creation.

Zurück Weiter

PhotoQuest

Not Secure photoquest.local:8080/onboarding/karten

PhotoQuest

Vorne Hinten

1 **Gestalte deine Karten**

Diese Karten erhalten deine Gäste auf der Hochzeit

Hochzeitsdatum

13.10.2024

Erster Partner

Name eingeben | Weiblich

Zweiter Partner

Name eingeben | Männlich

Zugehörigkeit

Bist du Gast oder Teil des Brautpaars?

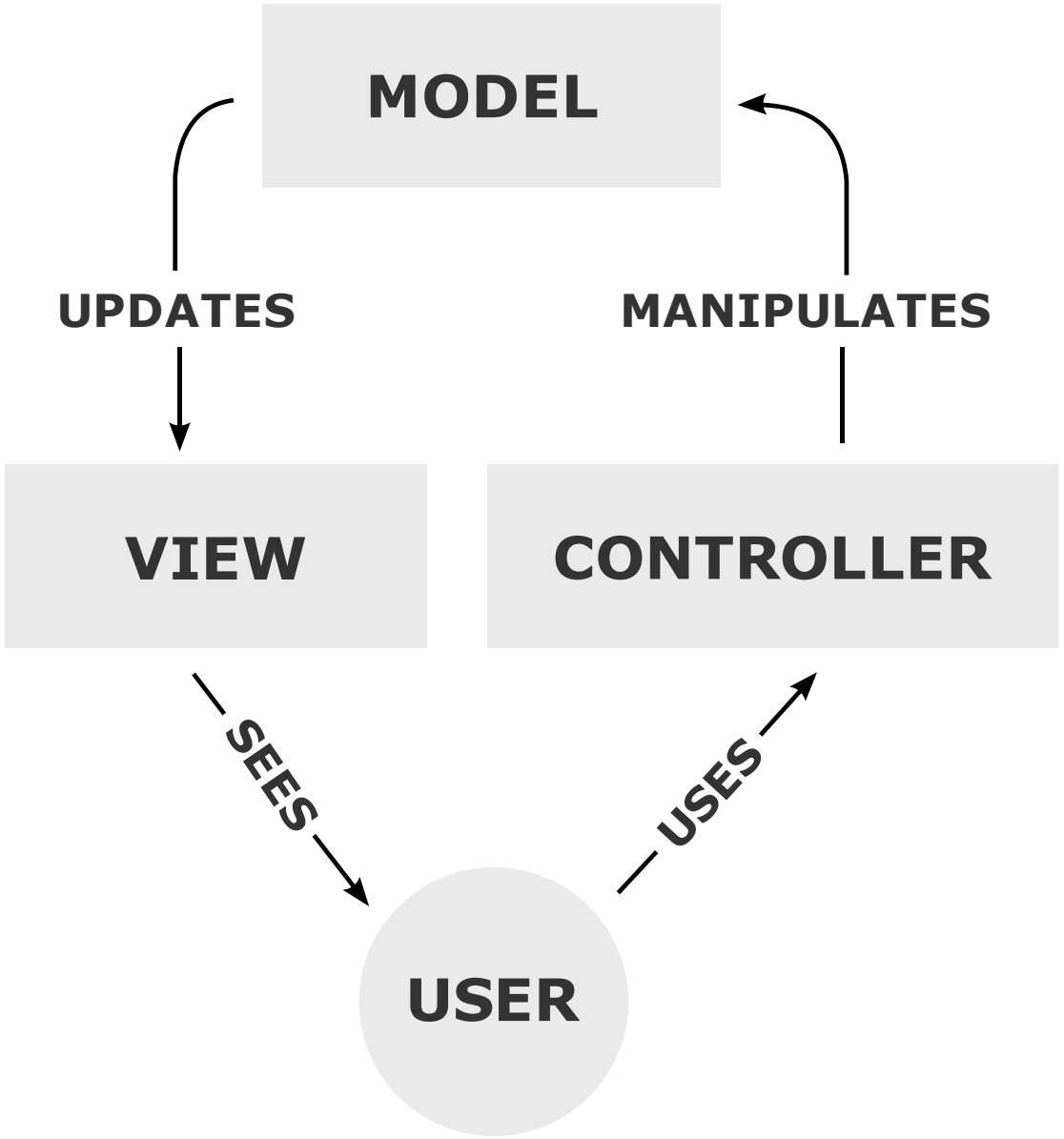
Vorlagen

Zurück Weiter

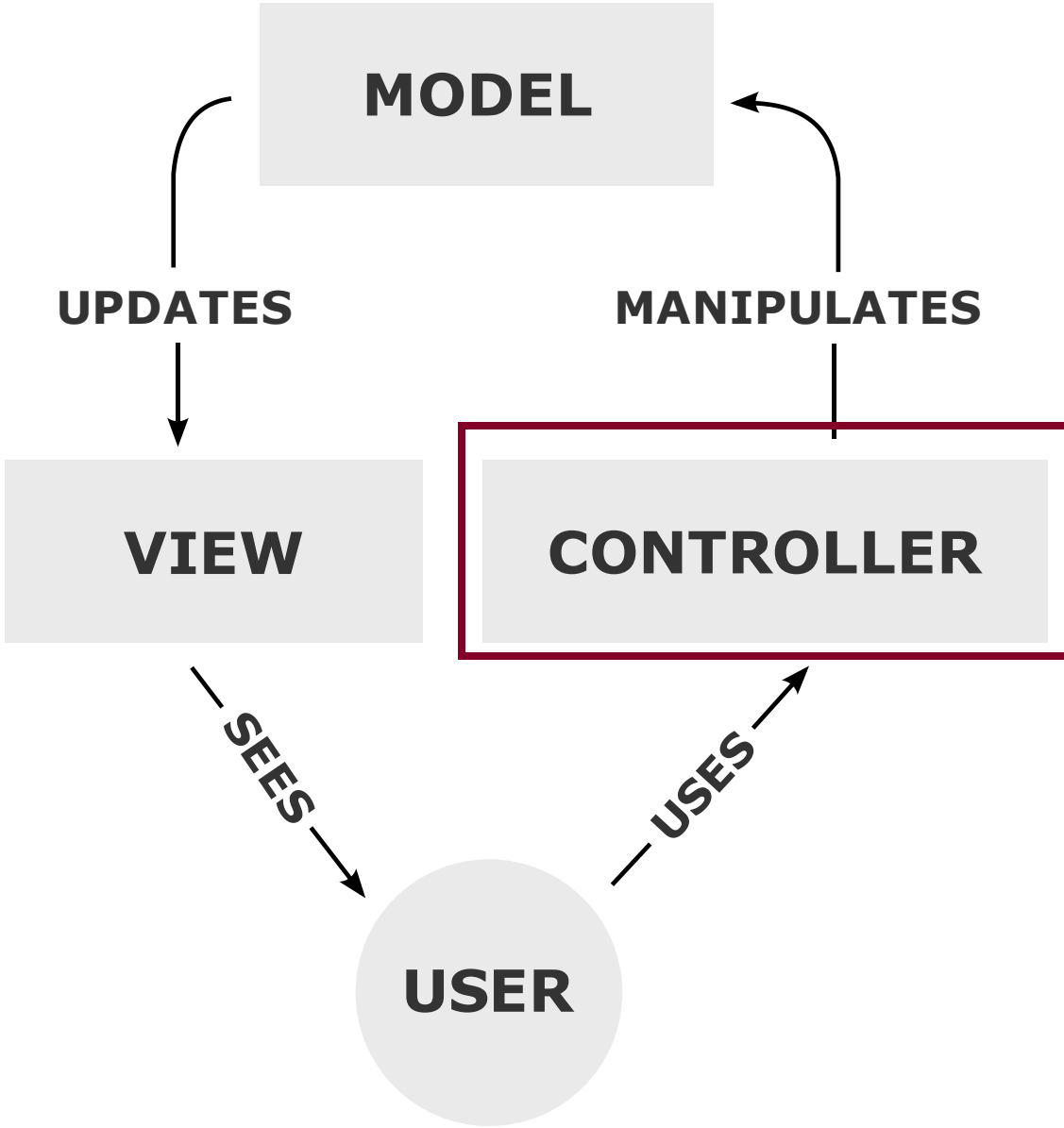
Generate


spring

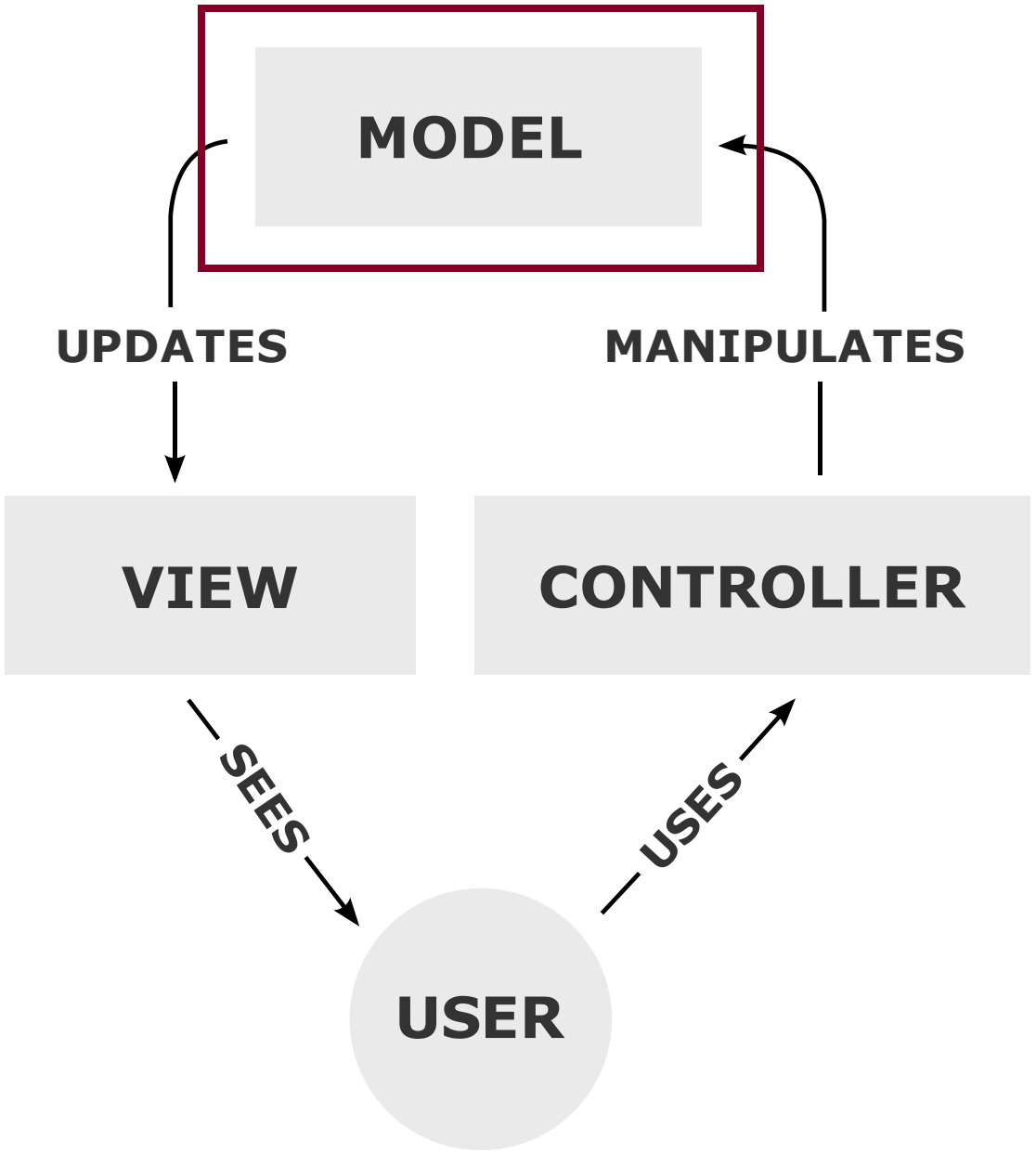

```
@Controller
class GreetingController {

    @GetMapping("/greeting")
    fun home(model: Model): String{

        model.addAttribute("name", "SpringIO")

        return "greeting"
    }
}
```



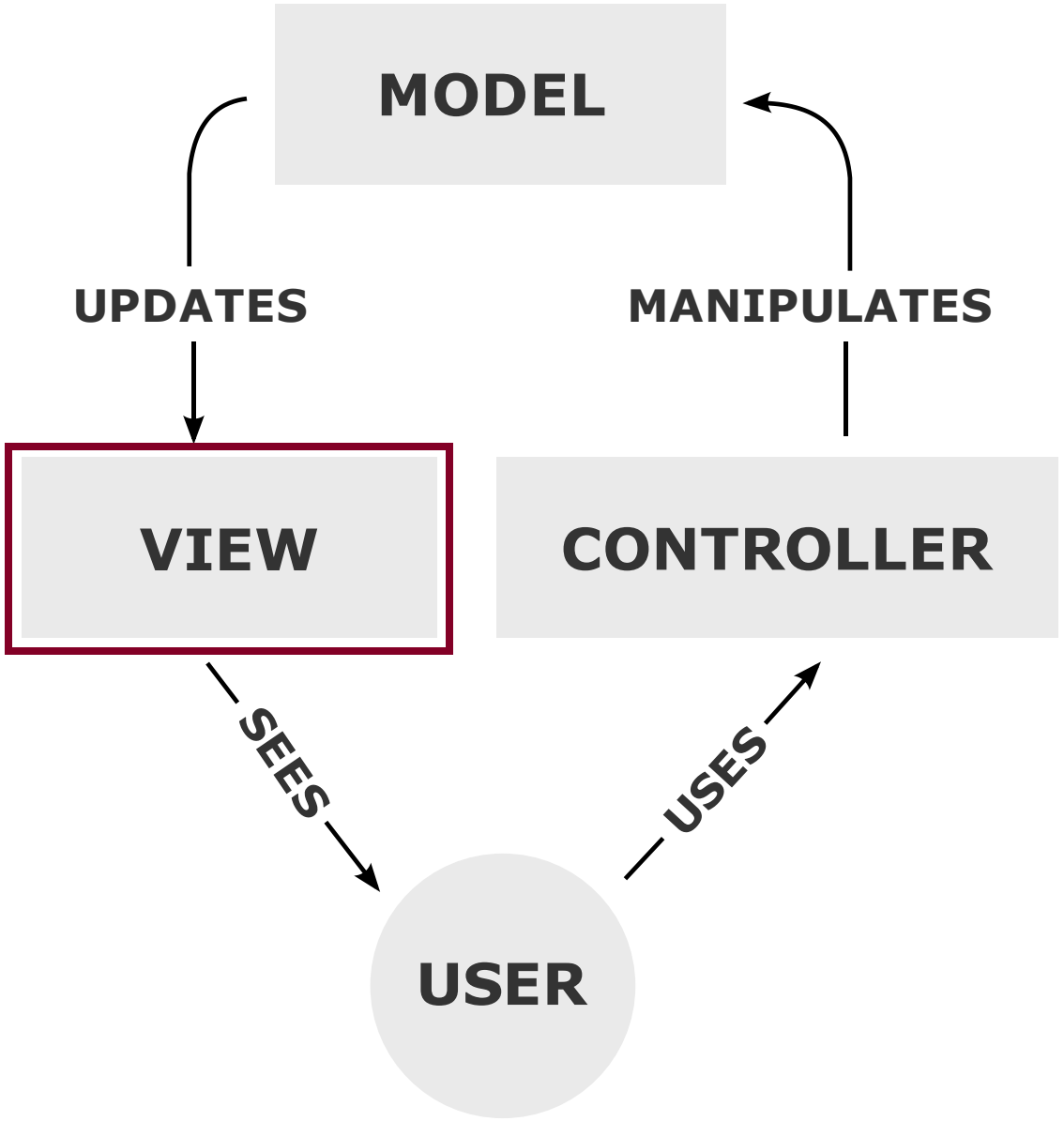

```java
@Controller
class GreetingController {

    @GetMapping("/greeting")
    fun home(model: Model): String{

        model.addAttribute("name", "SpringIO")

        return "greeting"
    }
}
```




```kotlin
@Controller
class GreetingController {

    @GetMapping("/greeting")
    fun home(model: Model): String{

        model.addAttribute("name", "SpringIO")

        return "greeting"
    }
}
```




```kotlin
@Controller
class GreetingController {

    @GetMapping("/greeting")
    fun home(model: Model): String{

        model.addAttribute("name", "SpringIO")

        return "greeting"
    }
}
```

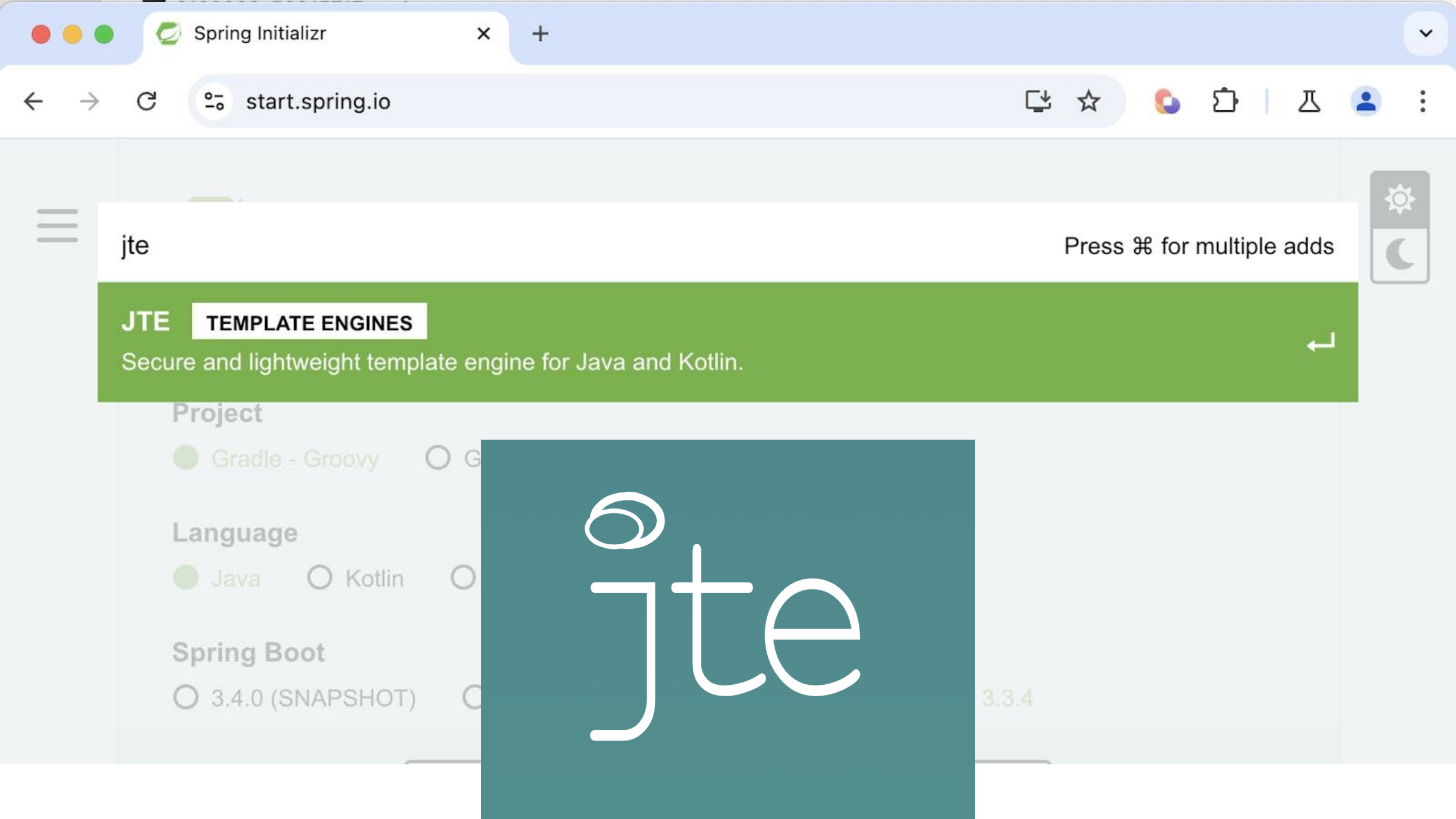



jstachio


Thymeleaf


<#FREEMARKER>


jte


Spring Initializr
start.spring.io
jte
Press ⌘ for multiple adds
JTE
TEMPLATE ENGINES
Secure and lightweight template engine for Java and Kotlin.
Project
Gradle - Groovy
Language
Java
Kotlin
Spring Boot
3.4.0 (SNAPSHOT)
3.3.4
jte

```
@import org.example.Page
@param Page page

<head>
        <meta name="description"
              content="${page.getDescription()}">
    <title>${page.getTitle()}</title>
</head>
```

```
@import org.example.Page
@param Page page

<head>
        <meta name="description"
              content="${page.getDescription()}">
    <title>${page.getTitle()}</title>
</head>
```

```
@import org.example.Page
@param Page page

<head>
    @if(page.getDescription() != null)
        <meta name="description" content="${page.getDescription()}">
    @endif
    <title>${page.getTitle()}</title>
</head>
<body>
    <h1>${page.getTitle()}</h1>
    <p>Welcome to my example page!</p>
</body>
```

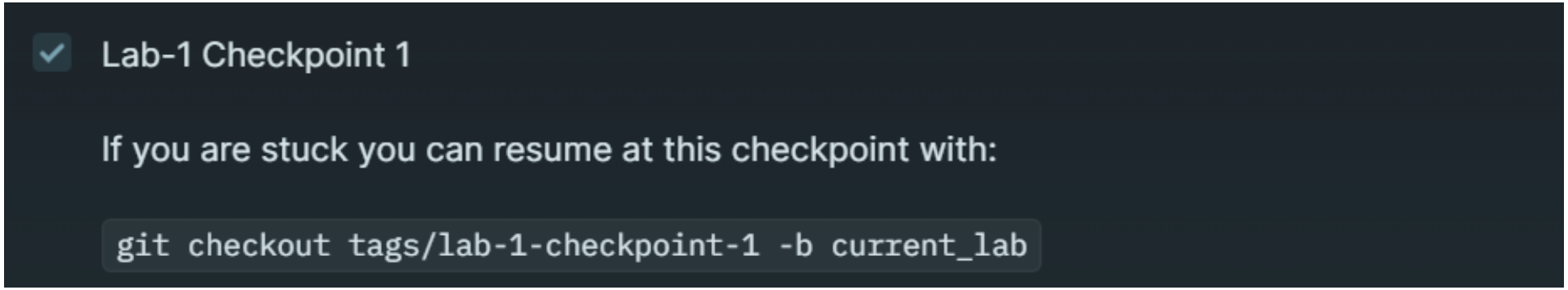
UserManagement.jte

```
@import de.tschuehly.easy.spring.auth.user.EasyUser
@param java.util.List<EasyUser> easyUserList

@for(var user: easyUserList)
    ${user.username}
@endfor
```

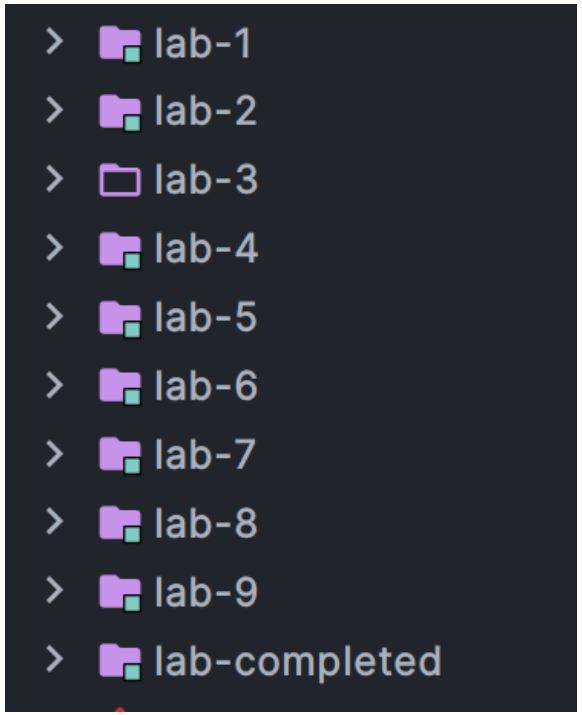
What are we going to build today?

Checkpoints

When you follow the workshop instructions there are checkpoints in the lab that you can start over at!



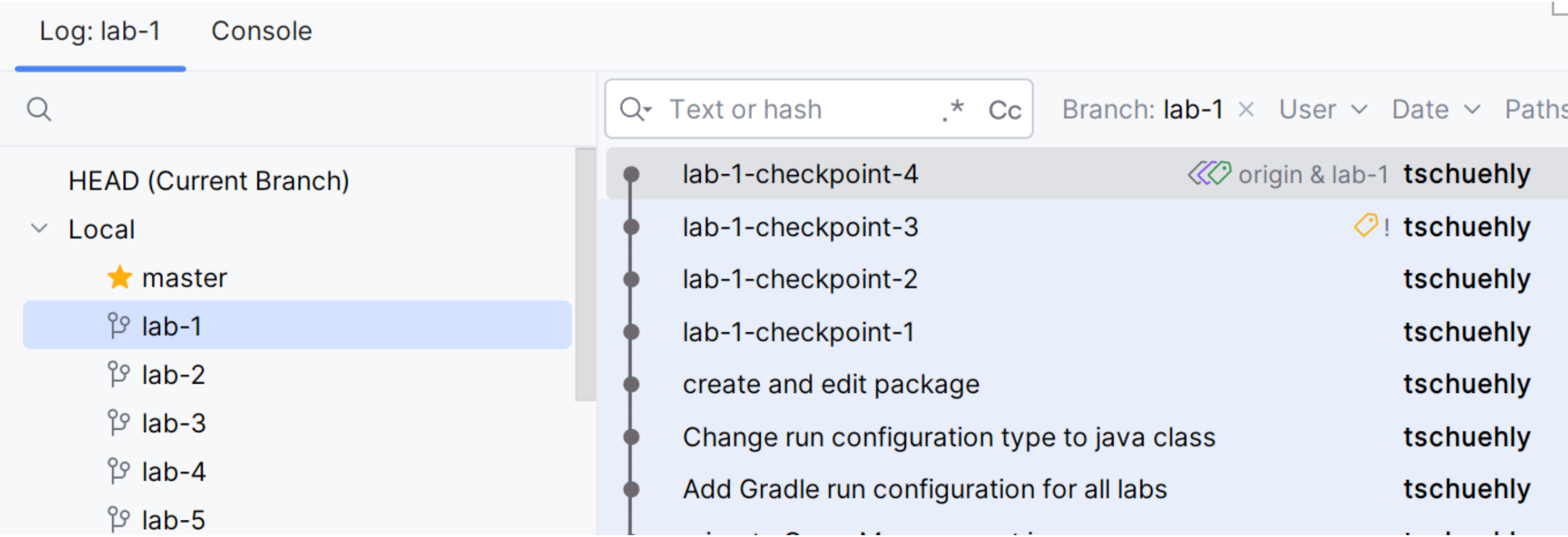
Repository structure:



For each lab there is a folder in the repo.
This is always the starting point of the lab.

For each lab there is a corresponding branch,
this is the state of the completed lab.

For each checkpoint there is a commit that
shows the changes



Lab 1: Server-side rendering with Spring Boot and JTE

htmx attributes:

- hx-get
- hx-target
- hx-post
- hx-on

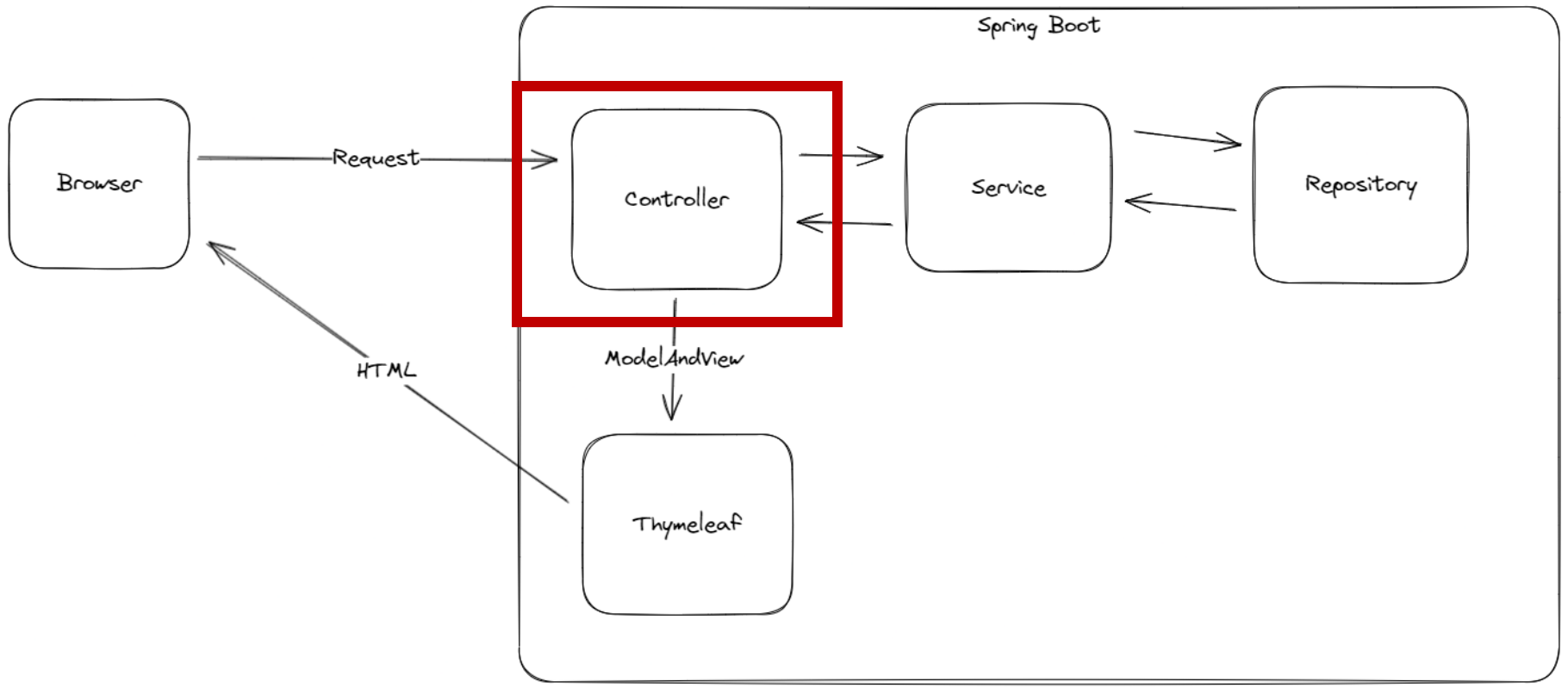
htmx HTTP response headers:

- HX-Retarget
- HX-Reswap
- HX-Trigger

Progress Tracker + Q&A

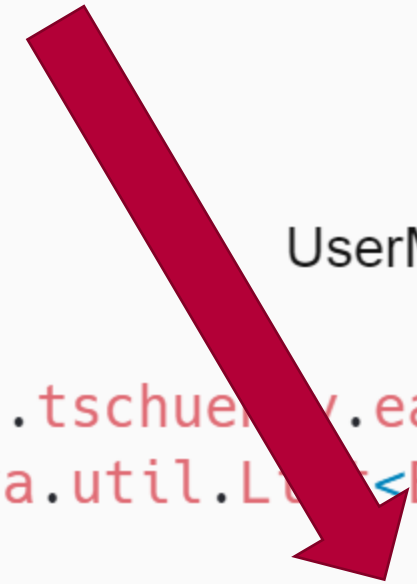

spring

MPA


Spring Boot
Browser
Request
Controller
Service
Repository
HTML
ModelAndView
Thymeleaf

UserController.java

```java
@Controller
public class UserController {
  @GetMapping("/")
  public String index(Model model) {
    model.addAttribute("easyUserList", userService.findAll());
    return "UserManagement";
  }
}
```

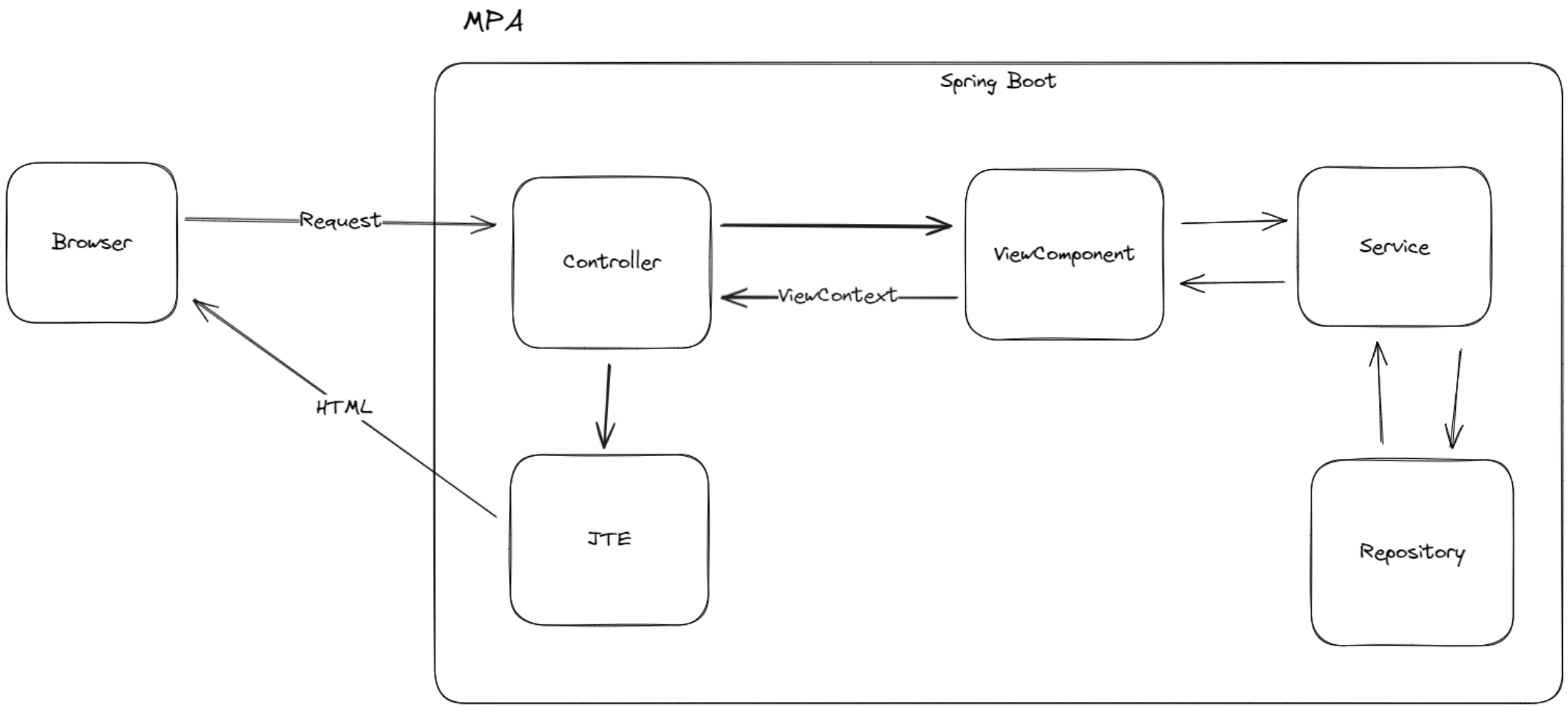


UserManagement.jte

```
@import de.tschuehly.easy.spring.auth.user.EasyUser
@param java.util.List<EasyUser> easyUserList

@for(var user: easyUserList)
    ${user.username}
@endfor
```

ViewComponent

github.com/tschuehly/spring-view-component


MPA
Spring Boot
Browser
Request
Controller
ViewComponent
Service
ViewContext
HTML
JTE
Repository

UserManagement.java

```java
@ViewComponent
public class UserManagement {
  private final UserService userService;

  public UserManagement(UserService userService) {
    this.userService = userService;
  }

  public record UserManagementContext(java.util.List<EasyUser> userTable)
        implements ViewContext{}

  public ViewContext render(){
    return new UserManagementContext(userService.findAll());
  }
}
```

UserManagement.java

```java
@ViewComponent
public class UserManagement {
  private final UserService userService;

  public UserManagement(UserService userService) {
    this.userService = userService;
  }

  public record UserManagementContext(java.util.List<EasyUser> userTable)
        implements ViewContext{}

  public ViewContext render(){
    return new UserManagementContext(userService.findAll());
  }
}
```

UserManagement.java

```java
@ViewComponent
public class UserManagement {

  public record UserManagementContext(java.util.List<EasyUser> userTable)
        implements ViewContext{}

  public ViewContext render(){
    return new UserManagementContext(userService.findAll());
  }
}
```

UserManagement.jte

```
@import de.tschuehly.easy.spring.auth.user.management.UserManagement.UserManagementContext
@param UserManagementContext userManagementContext
@for(var user: userManagementContext.easyUserList())
    ${user.username}
@endfor
```

UserController.java

```java
@Controller
public class UserController {

  private final UserManagement userManagement;

  public UserController(UserManagement userManagement) {
    this.userManagement = userManagement;
  }

  @GetMapping("/")
  public ViewContext index() {
    return userManagement.render();
  }
}
```

UserController.java

```java
@Controller
public class UserController {

  private final UserManagement userManagement;

  public UserController(UserManagement userManagement) {
    this.userManagement = userManagement;
  }

  @GetMapping("/")
  public ViewContext index() {
    return userManagement.render();
  }
}
```

LayoutViewComponent.java

```java
@ViewComponent
public class LayoutViewComponent {

  public ViewContext render(ViewContext nestedViewComponent) {
    return new LayoutView(nestedViewComponent);
  }

  public record LayoutView(ViewContext nestedViewComponent) implements ViewContext {}
}
```

LayoutViewComponent.jte

```
@param de.tschuehly.example.jte.web.layout.LayoutViewComponent.LayoutView layoutView
<nav></nav>
${layoutView.nestedViewComponent()}
<footer></footer>
```

Lab 2: Using Spring ViewComponent

Lab 3: Inline Editing

Using Spring Bean to compose the UI

Lazy Loading

Lab 6: Full Text Search

Lab 7: Infinite Scroll

Lab 8: Exception Messages

Lab 9: Server-Sent Events

Feedback

https://forms.office.com/r/M9zfP3G6ii

Comments

Add Comment

Search Comments

Show All

admin@alanda.io

Hello World

20:28 - 26. Sep 2024 - Address

```
<form hx-post="${commentAreaContext.createComment()}"
      hx-on:htmx:after-request="this.reset();">
  <textarea name="commentText"></textarea>
  <button type="submit">
    Create Comment
  </button>
</form>
<div id="${CommentAreaContext.commentListId}">
</div>
```

```java
public record CommentAreaContext(
    CommentAreaDefinition commentAreaDefinition
  ) implements ViewContext {

  public static final String CREATE_COMMENT = "/api/comment";

  public Url createComment() {
    return Url.path(CREATE_COMMENT)
        .addParam("dataItemId", commentAreaDefinition.dataItemId());
  }
}
```

```java
public class Url implements Content {

  private UriComponentsBuilder uriComponentsBuilder;

  @Override
  public void writeTo(TemplateOutput output) {
    output.writeContent(uriComponentsBuilder.build().toString());
  }
}
```

```
<form hx-post="/api/comment?dataItemId=5afbd5e5-ec95-468f-81d8-1612da3a070e"
</form>
```

```
public Url addParam(String name, Object value) {
  if (Objects.isNull(value)) {
    return this;
  }
  uriComponentsBuilder.queryParam(name, URLEncoder.encode(
    value.toString(), StandardCharsets.UTF_8));
  return this;
}
```

```java
@ViewComponent
@Controller
public class CommentAreaComponent {

  @PostMapping(CREATE_COMMENT)
  public ViewContext createComment(
      @RequestParam UUID dataItemId,
      @RequestParam String commentText
  ) {
    Comment comment = commentService.createComment(dataItemId, commentText);
    HtmxUtil.retargetToId(CommentAreaContext.commentListId);
    HtmxUtil.reswap(HxSwapType.AFTER_BEGIN);
    return commentComponent.render(new CommentDefinition(comment));
  }
}
```

```java
public class HtmxUtil {

  public static void retargetToId(String id) {
    setHeader(HtmxResponseHeader.HX_RETARGET, "#" + id);
  }

  public static void reswap(HxSwapType hxSwapType) {
    setHeader(HtmxResponseHeader.HX_RESWAP, hxSwapType.getValue());
  }
}
```

▼Response Headers

Cache-Control:	no-cache, no-store, max-age=0, must-revalidate
Content-Language:	en-US
Content-Length:	5620
Content-Type:	text/html;charset=UTF-8
Date:	Thu, 26 Sep 2024 20:45:09 GMT
Expires:	0
Hx-Reswap:	afterbegin
Hx-Retarget:	#commentList
Pragma:	no-cache
Strict-Transport-Security:	max-age=31536000; includeSubDomains
Vary:	HX-Request


alanda

Function*

Mustermann

Authority*

Landratsamt Musterstadt

Firstname*

Max

Lastname*

Mustermann

E-Mail Address*

max.mustermann@example.co

Phone Number*

+491234567890

Address

Street Name*

Musterstraße

Street Number*

1

Unit Number*

12345

Postal Code*

98765

City*

Musterstadt

State*

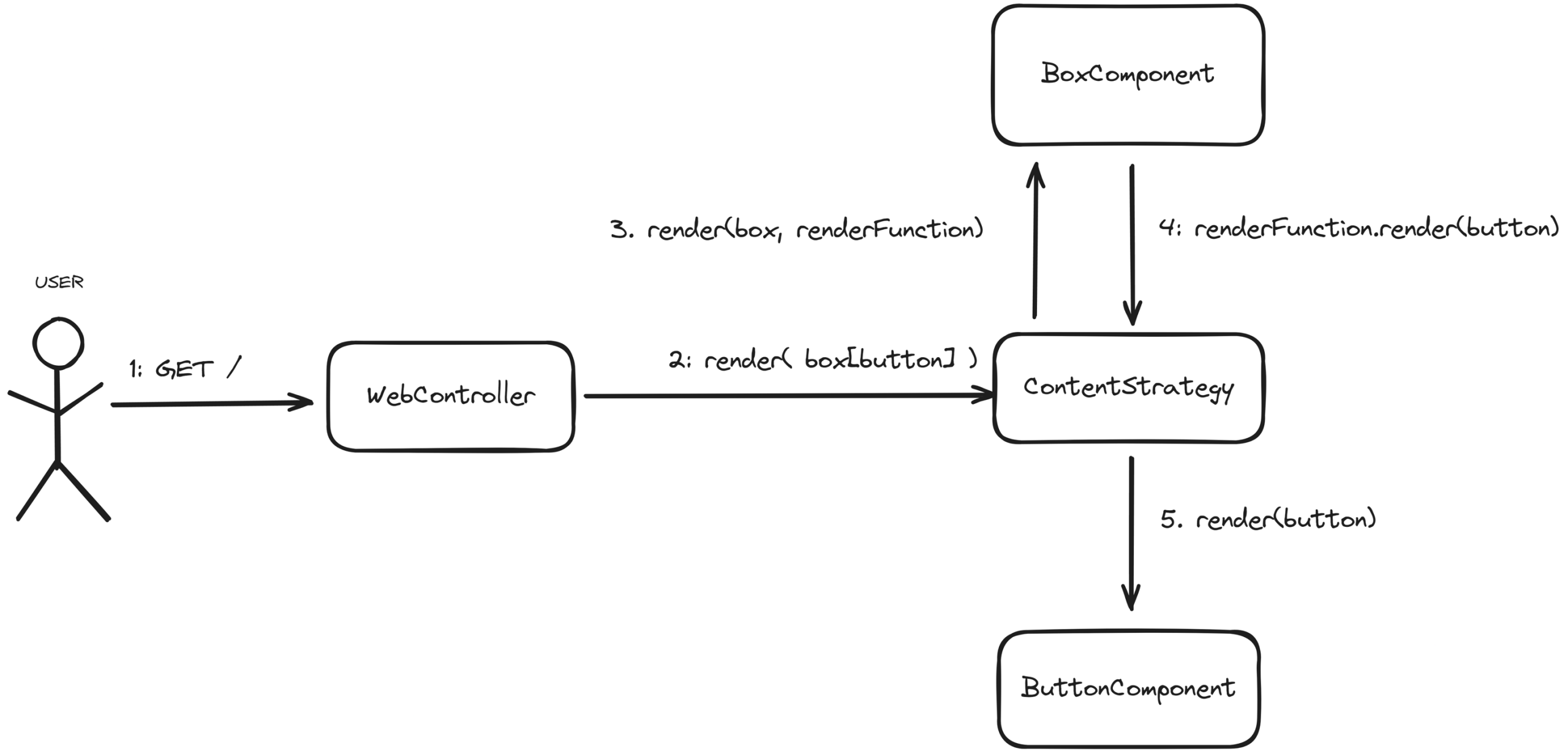
Musterstaat

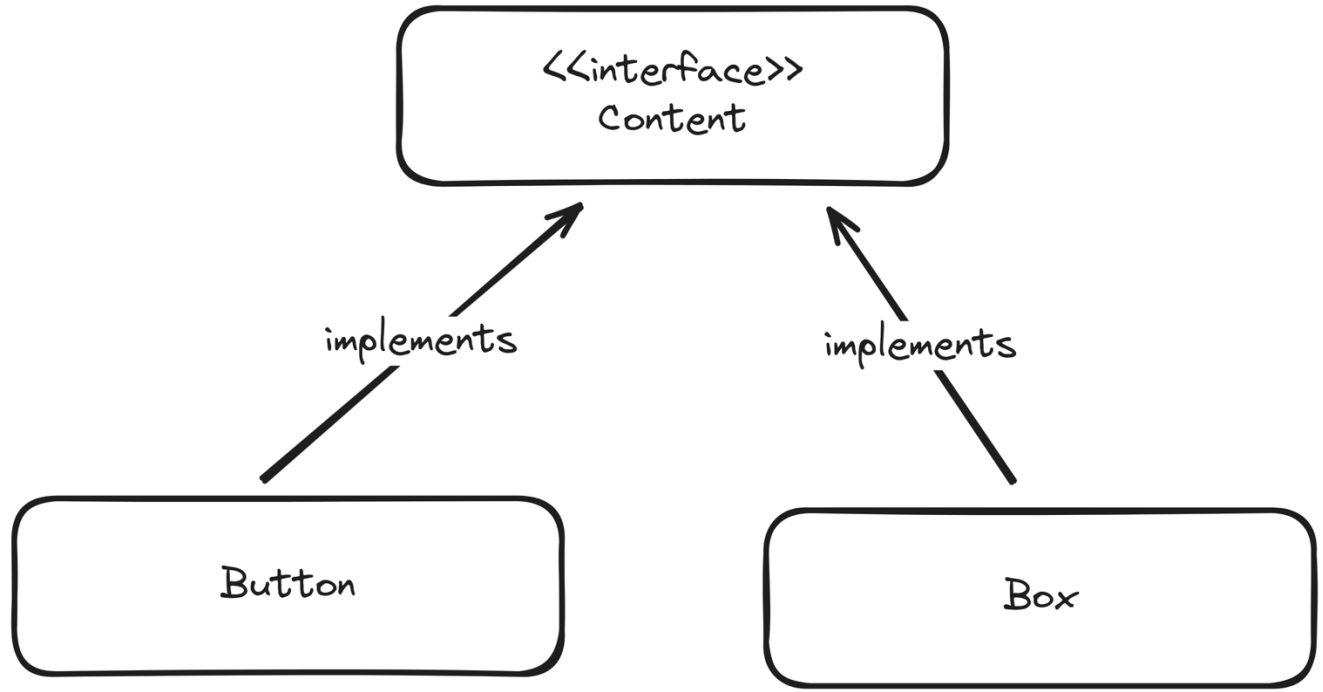
Country*

Musterland

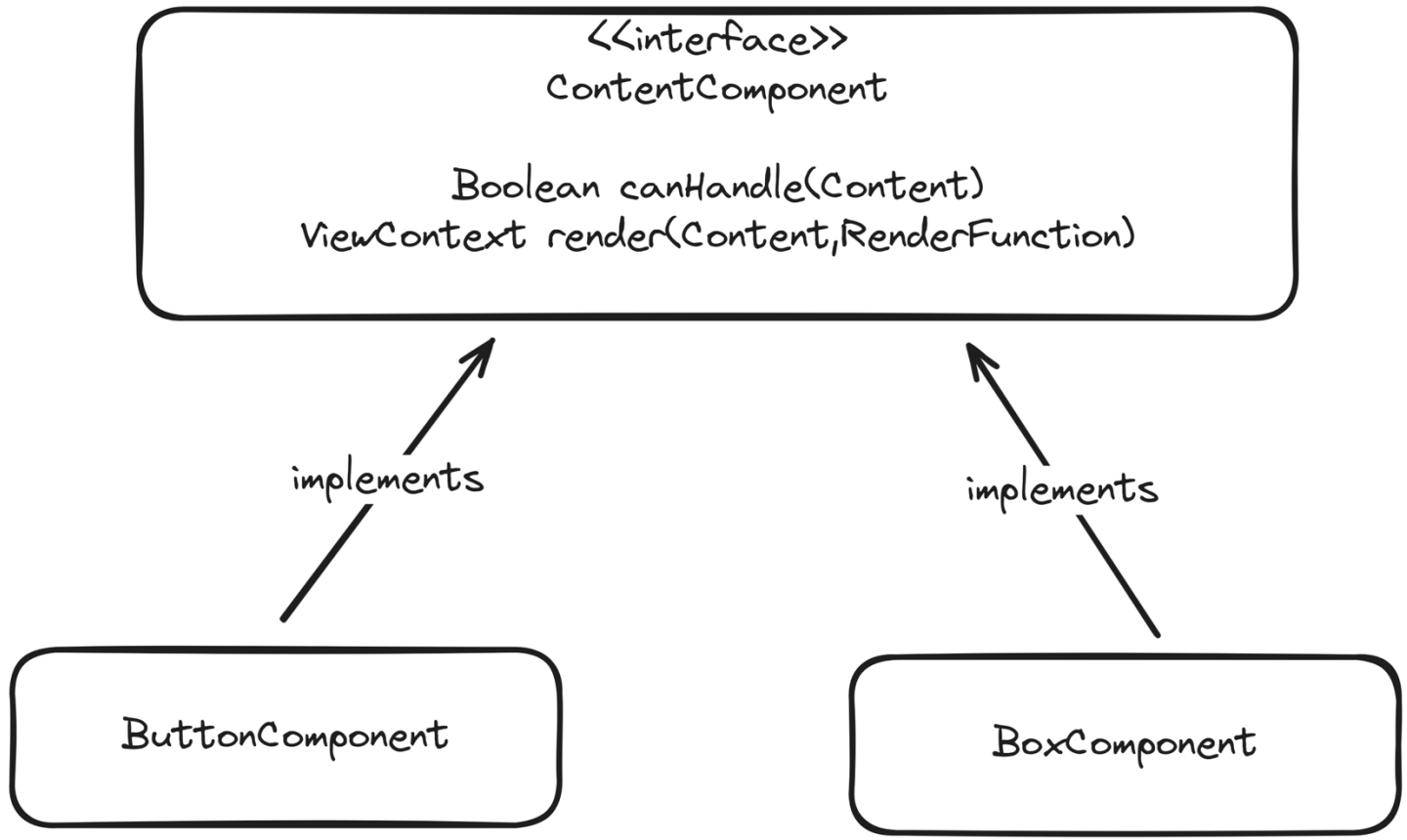
Complete

```java
@Controller
public class WebController {

  @Autowired
  private ContentStrategy contentStrategy;

  @GetMapping("/")
  ViewContext index(){
    return contentStrategy.render(
        new Box(
            new Button("Hello World")
        )
    );
  }
}
```


BoxComponent
3. render(box, renderFunction)
4: renderFunction.render(button)
USER
1: GET /
WebController
2: render(box[button])
ContentStrategy
5. render(button)
ButtonComponent




```java
public record Button(String label) implements Content {

}
```


<<interface>>
ContentComponent
Boolean canHandle(Content)
ViewContext render(Content,RenderFunction)
implements
implements
ButtonComponent
BoxComponent

```java
@ViewComponent
public class ButtonComponent implements ContentComponent {

  @Override
  public Boolean canHandle(Content content) {
    return content instanceof Button;
  }

  @Override
  public ViewContext render(
    Content content,
    RenderFunction renderFunction
  ) {
    return new ButtonContext((Button) content);
  }


  public record ButtonContext(Button button)
    implements ViewContext {}
}
```

```
@param de.tschuehly.svc.ui.field.button.ButtonComponent.ButtonContext buttonContext

<button>
    ${buttonContext.button().label()}
</button>
```

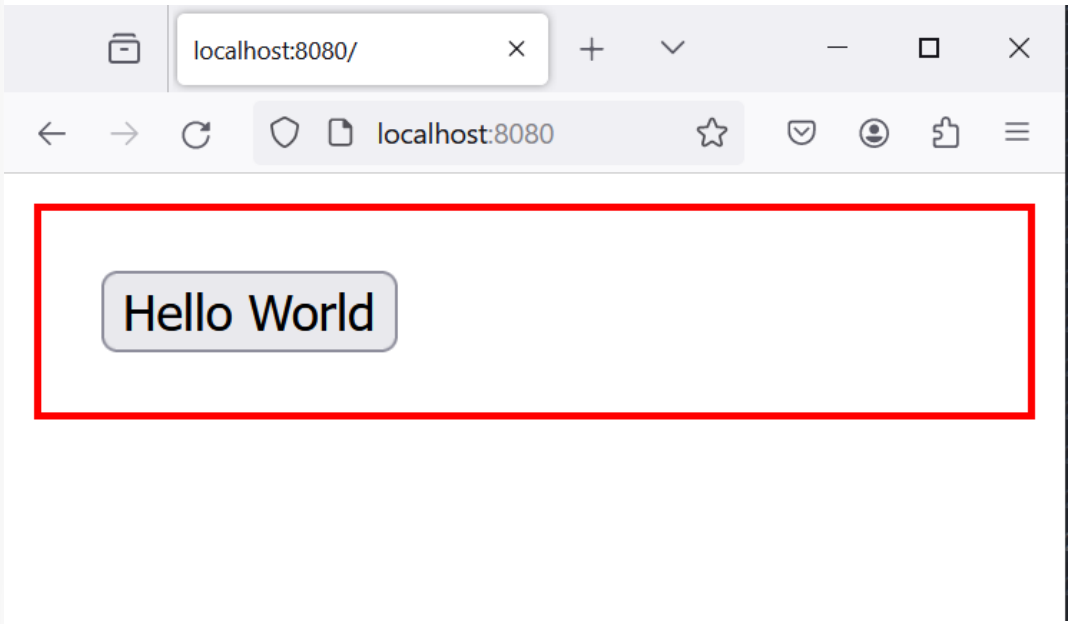
```java
@Service
public class ContentStrategy {

  @Autowired
  private final List<ContentComponent> contentComponents;

  public ViewContext render(Content content) {
    ContentComponent contentComponent = contentComponents.stream()
        .filter(it -> it.canHandle(content))
        .findFirst()
        .orElseThrow(() -> new RuntimeException("FormNotImplementedException"));
    return contentComponent.render(content, this::render);
  }
}
```

```
public record Box(
    List<Content> boxContentList
) implements Content {
  public Box(Content... contents){
    this(Arrays.stream(contents).toList());
  }
}
```

```java
@ViewComponent
public class BoxComponent implements ContentComponent {
  @Override
  public Boolean canHandle(Content content) {
    return content instanceof Box;
  }

  @Override
  public ViewContext render(Content content, RenderFunction renderFunction) {
    return new BoxContext((Box) content, renderFunction);
  }
  public record BoxContext(Box content, RenderFunction renderFunction) implements ViewContext {}
}
```

```
@param de.tschuehly.svc.ui.layout.box.BoxComponent.BoxContext boxContext

<div>
    @for(var content: boxContext.content().boxContentList())
        ${boxContext.renderFunction().render(content)}
    @endfor
</div>
```

```java
@Controller
public class WebController {

  @Autowired
  private ContentStrategy contentStrategy;

  @GetMapping("/")
  ViewContext index(){
    return contentStrategy.render(
        new Box(
            new Button("Hello World")
        )
    );
  }
}
```




BUT WHAT ABOUT THE FUTURE?

wimdeblauwe/htmx-spring-boot

#104 Functional HTMX Endpoints

15 comments

tschuehly opened on April 15, 2024

```java
@ViewComponent
public class EndpointComponent {
    public record EndpointContext(
            EndpointComponent server, Optional<UserForm> userForm
    ) implements ViewContext {
        public static String COMPONENT_ID = "endpointComponent";
    }

    public ViewContext render() {
        return new EndpointContext( server: this,  userForm: Optional.empty());
    }

    public record UserForm(
            String userName, String password) {
    }
}
```

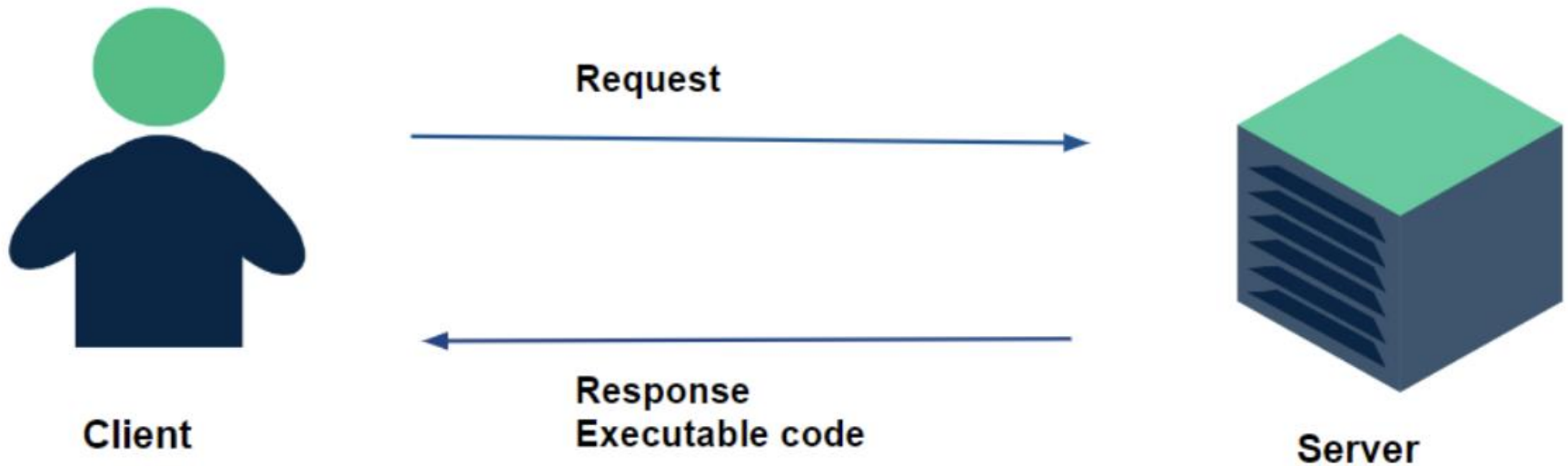
```
@param EndpointContext endpointContext
<script src="https://unpkg.com/htmx.org@1.9.12"></script>

<div id="${EndpointContext.COMPONENT_ID}">
    @if(endpointContext.userForm().isPresent())
        !{UserForm form = endpointContext.userForm().get();}
        <div>
            Hello ${form.userName()}! Your password is: ${form.password()}
        </div>
    @endif
</div>
```

```java
@ViewComponent
public class EndpointComponent {
    public HtmxEndpoint<UserForm, ViewContext> createUserEndpoint
            = new HtmxEndpoint<>(
            path: "/createUser",
            method: HttpMethod.POST,
            function: this::createUser,
            targetId: EndpointContext.COMPONENT_ID
    );

    private ViewContext createUser(UserForm userForm) {
        return new EndpointContext(
                server: this,
                userForm: Optional.ofNullable(
                        value: userForm
                )
        );
    }
```

```
@ViewComponent
public class EndpointComponent {
    public HtmxEndpoint<UserForm, ViewContext> createUserEndpoint
            = new HtmxEndpoint<>(
            path: "/createUser",
            method: HttpMethod.POST,
            function: this::createUser,
            targetId: EndpointContext.COMPONENT_ID
    );

<div id="${EndpointContext.COMPONENT_ID}">
    <form $unsafe{endpointContext.server().createUserEndpoint.call()}>
        <label>
            username
            <input type="text" name="userName">
        </label>
        <label>
            password
            <input type="text" name="password">
        </label>
        <button type="submit">
            Save
        </button>
    </form>
```

```
<div id="${EndpointContext.COMPONENT_ID}">
    <form $unsafe{endpointContext.server().createUserEndpoint.call()}>
        <label>
            username
            <input type="text" name="userName">
        </label>
        <label>
            password
            <input type="text" name="password">
        </label>
        <button type="submit">
            Save
        </button>
    </form>
```

```
▼<body>
  ▼<div id="endpointComponent">
    ▼<form hx-post="/createUser" hx-target="#endpointComponent" hx-swap="outerHTML">
      ▼<label>
          " username "
          <input type="text" name="userName">
        </label>
      ▼<label>
          " password "
          <input type="text" name="password">
        </label>
        <button type="submit"> Save </button>
      </form>
    </div>
  </body>
</html>
```


Request
Response
Executable code
Client
Server

ToastViewComponent.html

```
<div id="toast-element">
  <script th:inline="javascript" >
  </script>
</div>
```

ToastViewComponent.html

```
<div id="toast-element">
  <script th:inline="javascript" >
  </script>
</div>
```

ToastViewComponent.html

```
<div id="toast-element">
  <script th:inline="javascript" >
  </script>
</div>
```

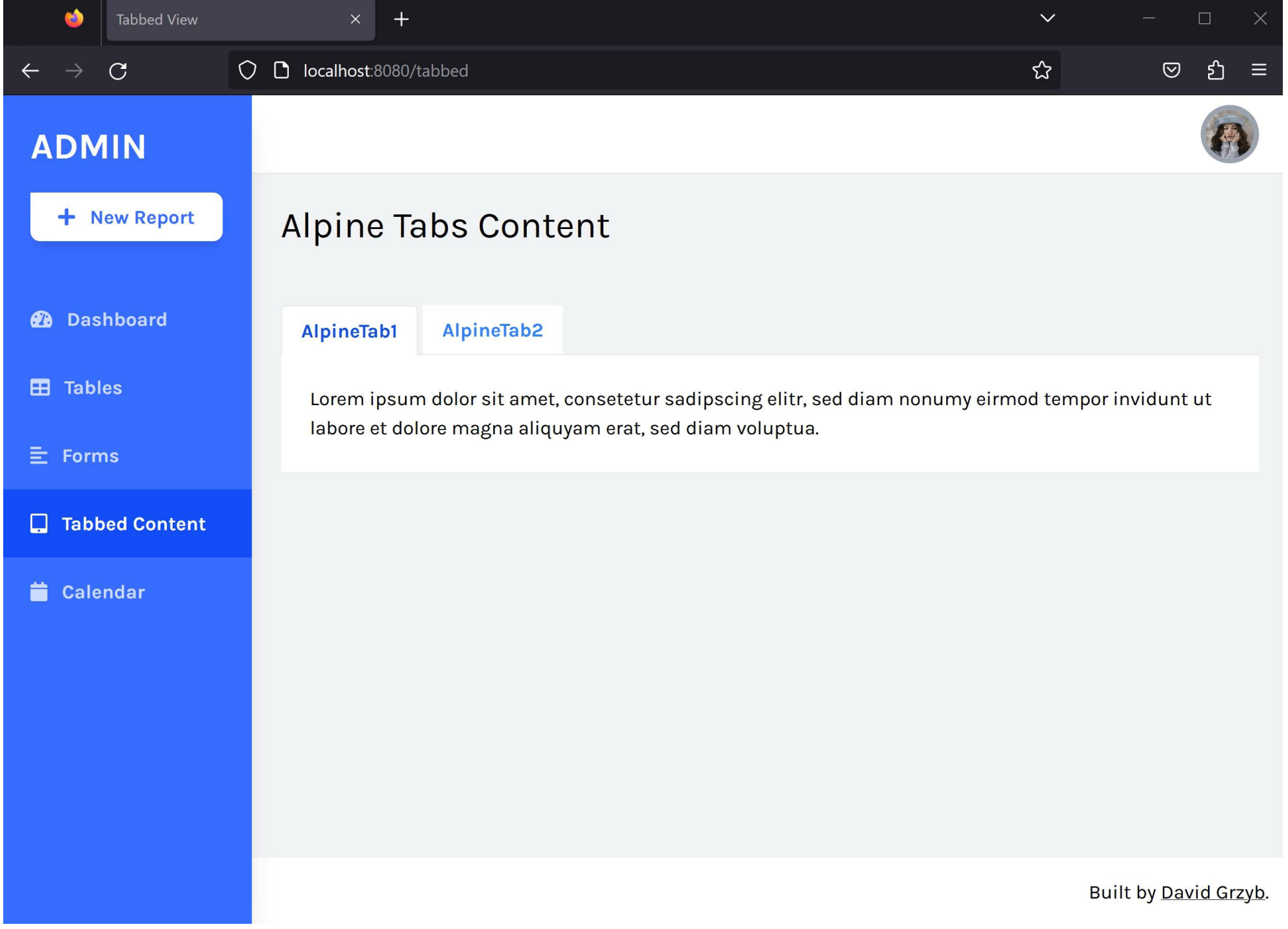
ToastViewComponent.html

```
<div id="toast-element">
  <script th:inline="javascript" >
  </script>
</div>
```

ToastViewComponent.html

```
<div id="toast-element">
  <script th:inline="javascript" >
  </script>
</div>
```


Alpine.js


Tabbed View
localhost:8080/tabbed
ADMIN
New Report
Dashboard
Tables
Forms
Tabbed Content
Calendar
Alpine Tabs Content
AlpineTab1
AlpineTab2
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
Built by David Grzyb.

AlpineTabViewComponent.kt | AlpineTabViewComponent.html

```kotlin
@ViewComponent
class AlpineTabViewComponent {
    fun render(tabs: List<Tab>){
    }

    class Tab( val tabHeader: String, val tabContent: String )
}
```

AlpineTabViewComponent.kt | AlpineTabViewComponent.html

```kotlin
@ViewComponent
class AlpineTabViewComponent {
    fun render(tabs: List<Tab>){
    }

    class Tab( val tabHeader: String, val tabContent: String )
}
```

AlpineTabViewComponent.kt | AlpineTabViewComponent.html

```kotlin
@ViewComponent
class AlpineTabViewComponent {
    fun render(tabs: List<Tab>){
    }

    class Tab( val tabHeader: String, val tabContent: String )
}
```

AlpineTabViewComponent.kt | AlpineTabViewComponent.html

```kotlin
@ViewComponent
class AlpineTabViewComponent {
    fun render(tabs: List<Tab>){
    }

    class Tab( val tabHeader: String, val tabContent: String )
}
```

AlpineTabViewComponent.kt | AlpineTabViewComponent.html

```kotlin
@ViewComponent
class AlpineTabViewComponent {
    fun render(tabs: List<Tab>){
    }

    class Tab( val tabHeader: String, val tabContent: String )
}
```

AlpineTabViewComponent.kt | AlpineTabViewComponent.html

```kotlin
@ViewComponent
class AlpineTabViewComponent {
    fun render(tabs: List<Tab>){
    }

    class Tab( val tabHeader: String, val tabContent: String )
}
```

AlpineTabViewComponent.kt | AlpineTabViewComponent.html

```kotlin
@ViewComponent
class AlpineTabViewComponent {
    fun render(tabs: List<Tab>){
    }

    class Tab( val tabHeader: String, val tabContent: String )
}
```

AlpineTabViewComponent.kt | AlpineTabViewComponent.html

```kotlin
@ViewComponent
class AlpineTabViewComponent {
    fun render(tabs: List<Tab>){
    }

    class Tab( val tabHeader: String, val tabContent: String )
}
```

```
<div x-data="{ openTab: 0 }">
    <ul>
        <li x-on:click="openTab = 0">
            <button>AlpineTab1</button>
        </li>
        <li x-on:click="openTab = 1">
            <button>AlpineTab2</button>
        </li>
    </ul>
    <div>
        <div x-show="openTab === 0">
          Lorem ipsum dolor sit amet, consetetur sadipscing
        </div>
        <div x-show="openTab === 1" style="display: none;">
          At vero eos et accusam et justo duo dolores
        </div>
    </div>
</div>
```