THILO FROTSCHER
Software Architect & Trainer

Enterprise Java, Services und Systemintegration

MVC (JSR 371)

Actionbasierte Web-Anwendungen mit Java EE

http://www.frotscher.com @thfro thilo@frotscher.com

THILO FROTSCHER
Software Architect & Trainer

Vorstellung

- Softwarearchitekt, Entwickler und Trainer
- Fachliche Schwerpunkte
  - Java Plattform
  - Backend-Architekturen
  - Kommunikation und Systemintegration
- Kundenspezifische Inhouse-Schulungen
- (Co-) Autor mehrerer Bücher & zahlreicher Artikel
- Sprecher auf internationalen Konferenzen

© 2016 Thilo Frotscher

@thfro

Wie alles begann…

- Servlet API
- JavaServer Pages
- „Kreativer“ Einsatz durch Entwickler
- J2EE Blueprints von Sun Microsystems
  - M=EJB, V=JSP, C=Servlet
  - Java Pet Store

THILO FROTSCHER
Software Architect&Trainer

Model-View-Controller

- Zahlreiche Web-Frameworks setzen MVC um
  - Java, JavaScript, .NET
- Java-Welt: Große Anzahl von Web-Frameworks
  - Herausforderung: Auswahl treffen
  - Artikel, Talks, Kriterienkataloge…
- Ansätze von MVC-Frameworks
  - actionbasiert (oder requestbasiert)
  - komponentenbasiert

© 2016 Thilo Frotscher

THILO FROTSCHER
Software Architect&Trainer

Actionbasiert vs. Komponentenbasiert

Actionbasiertes Framework	Komponentenbasiertes Framework
Fokus auf Requests	Fokus auf Seite / Komponenten
Manuelle Verarbeitung von Parametern	Autom. Verarbeitung von Parametern
Entwickler erstellt HTML/CSS/JS	Komponente erstellt HTML/CSS/JS
Manuelle Validierung von Form-Daten	Autom. Validierung von Form-Daten
Manuelle Umwandlung von Form-Daten	Autom. Umwandlung von Form-Daten
Voller Zugriff auf HTTP Req/Resp	Kein Zugriff auf HTTP Req/Resp
Beispiele: Struts, Spring MVC	Beispiele: JSF, Wicket

© 2016 Thilo Frotscher

Euer Favorit?

© 2016 Thilo Frotscher

Web-Anwendungen mit Java EE

- Java EE bietet bislang nur Unterstützung für den komponentenbasierten Ansatz => JSF
- Ebenfalls umsetzbar mit Java EE: REST-Backend für JavaScript Frameworks (Controller im Client)
- Java EE 8 wird ein neues, actionbasiertes Web-Framework erhalten!

© 2016 Thilo Frotscher

THILO FROTSCHER
Software Architect&Trainer

Kontroverse Entscheidung

- Warum?
- Warum jetzt?
- Was wird aus JSF?
- Ist Rendern auf dem Server noch „modern“?
- Eigenständiger JSR oder bestehenden JSR erweitern? Welchen?
- Unglücklicher Name: „MVC“

© 2016 Thilo Frotscher

THILO FROTSCHER
Software Architect&Trainer

MVC (JSR 371)

- Aktueller Stand:
  Early Draft (Second Edition) vom 02.10.2015
- Ziele
  - Schlanke Spezifikation (Ed1: 25 Seiten, Ed2: 33 Seiten)
  - Möglichst viele Technologien wiederverwenden:
    CDI, Bean Validation, JSP, Facelets
- Zentrale Entscheidung
  - MVC wird auf JAX-RS basieren

© 2016 Thilo Frotscher

THILO FROTSCHER
Software Architect & Trainer

MVC mit MVC ☺

- Model:
  - CDI `@Named` Bean oder `Models`
- Controller
  - JAX-RS Ressourcen-Methode mit speziellen Regeln
  - CDI-only (nicht JAX-RS managed, keine EJBs)
  - Lifecycle: Per Request
- View
  - JSP oder Facelets

THILO FROTSCHER
Software Architect & Trainer

Controller (1)

```
@Path("hello")
public class HelloController {

  @GET
  @Controller
  public String hello() {
    return "hello.jsp";
  }
}
```

Default Media Type: text/html
alternativ @Produces verwenden

Rückgabetyp	Bedeutung
`void`	Methode muss mit `@View` markiert werden
`String`	Rückgabewert wird als Pfad zu einer View interpretiert
`Viewable`	Rückgabewert kapselt Informationen über eine View und optional deren `ViewEngine`
`Response`	JAX-RS Response deren Entity Type einer der drei obigen Typen ist
(Java Type)	`toString()` wird als Pfad zur View interpretiert

```
@Controller
@Path("hello")
public class HelloController {

  @GET @Path("void")
  @View("hello.jsp")
  public void helloVoid() {
  }

  @GET @Path("string")
  public String helloString() {
    return "hello.jsp";
  }

  @GET @Path("viewable")
  public Viewable helloViewable() {
    return new Viewable("hello.jsp");
  }

  @GET @Path("response")
  public Response helloResponse() {
    return Response.status(Response.Status.OK).entity("hello.jsp").build();
  }

  @GET @Path("myview")
  public MyView helloMyView() {
    return new MyView("hello.jsp"); // toString() -> "hello.jsp"
  }
}
```

© 2016 Thilo Frotscher

Viewable

```
public class Viewable {

    private String view;
    private Models models;
    private Class<? extends ViewEngine> viewEngine;

    public Viewable(String view) {
        this(view, null, null);
    }

    public Viewable(String view, Class<? extends ViewEngine> viewEngine) {
        this(view, null, viewEngine);
    }

    public Viewable(String view, Models models) {
        this(view, models, null);
    }

    public Viewable(String view, Models models,
                    Class<? extends ViewEngine> viewEngine) {
        this.view = view;
        this.models = models;
        this.viewEngine = viewEngine;
    }

    // getters & setters...
}
```

© 2016 Thilo Frotscher

THILO FROTSCHER
Software Architect&Trainer

Redirects

mit JAX-RS

```
@GET
@Controller
public Response redirect() {
  return Response.seeOther(URI.create("see/here")).build();
}
```

mit MVC

```
@GET
@Controller
public String redirect() {
  return "redirect:see/here";
}
```

Beide Alternativen führen zum gleichen Ergebnis
(HTTP Status Code 302)

THILO FROTSCHER
Software Architect&Trainer

Redirect Scope

```
@Controller
@Path("submit")
public class MyController {

  @Inject
  private MyBean myBean;

  @POST
  public String post() {
    myBean.setValue("Redirect about to happen");
    return "redirect:/submit";
  }

  @GET
  public String get() {
    // mybean.value accessed in JSP
    return "mybean.jsp";
  }
}
```

```
@RedirectScoped
public class MyBean {

  private String value;

  // getter & setter...
}
```

`myBean` ist verfügbar während des POST-Requests
und auch im anschließenden GET-Request
==> Zustand zwischen zwei Requests, managed by CDI

THILO FROTSCHER
Software Architect & Trainer

Modelle (1)

CDI @Named Bean

```
@Named("greeting")
@RequestScoped
public class Greeting {

  private String message;

  // getter & setter...
}
```

Controller

```
@Path("hello")
public class HelloController {

  @Inject
  private Greeting greeting;

  @GET
  @Controller
  public String hello() {
    greeting.setMessage("Hi there!");
    return "hello.jsp";
  }
}
```


THILO FROTSCHER
Software Architect & Trainer

Modelle (2)

- Modell auf Basis von `Models`

```
@Path("hello")
public class HelloController {

  @Inject
  private Models models;

  @GET
  @Controller
  public String hello() {
    models.set("greeting", new Greeting("Hi there!");
    return "hello.jsp";
  }
}
```

Views

- JSR: Unterstützung für JSP (Default) & Facelets

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <h1>${greeting.message}</h1>
  </body>
</html>
```

- Facelets: `FacesServlet` in *web.xml* „einschalten“
- Alternative Technologien durch „View Engines“

View Engine für Jade

```
@ApplicationScoped
public class JadeViewEngine implements ViewEngine {

  @Inject
  private ServletContext servletContext;

  @Override
  public boolean supports(String view) {
    return view.endsWith(".jade");
  }

  @Override
  public void processView(ViewEngineContext context)
                                        throws ViewEngineException {
    try {
      String viewName = "/WEB-INF/views/" + context.getView();
      URL template = servletContext.getResource(viewName);
      String html = Jade4J.render(template, context.getModels(), true);
      context.getResponse().getWriter().write(html);

    } catch (IOException e) {
      throw new ViewEngineException(e);
    }
  }
}
```

Jade ist eine beliebte Templating Engine für node.js

jade4J ist eine Portierung von Jade für Java

Validierung

- Integration mit Bean Validation wie bei JAX-RS

```
@Controller
@Path("form")
public class FormController {

  @POST
  public Response formPost(@Valid @BeanParam FormDataBean form) {
    return Response.status(OK).entity("data.jsp").build();
  }
}
```

- Definition der Validierungsregeln in `FormDataBean`, sowie ggf. in einem Validator

Fehlerbehandlung

Variante 1: JAX-RS `ExceptionMapper`

- Für Form-Validierung kein ideales Modell
  - alle Exceptions des gleichen Typs landen im gleichen Exception Mapper
  - z.B. auch jede `ConstraintViolationException`
- Formulare mit vielen Eingabefeldern und entsprechenden Validierungen führen zu Spaghetti-Code im Exception Mapper

THILO FROTSCHER
Software Architect & Trainer

Exception Handling durch Controller

```
@Controller
@Path("form")
@Produces("text/html")
public class FormController {

  @Inject
  private BindingResult br;

  @Inject
  private ErrorDataBean error;

  @POST
  @ValidateOnExecution(type = ExecutableType.NONE)
  public Response formPost(@Valid @BeanParam FormDataBean form) {
    if (br.isFailed()) {
      // populate ErrorDataBean...
      return Response.status(BAD_REQUEST).entity("error.jsp").build();
    }
    return Response.status(OK).entity("data.jsp").build();
  }
 }
```

Framework-Interface

Anwendungsklasse

THILO FROTSCHER
Software Architect & Trainer

MVC View Engines

- Aktuell zusätzlich unterstützte View-Technologien:

<FreeMarker>

Velocity

handlebars

thymeleaf

AsciiDoc

JADE
NODE TEMPLATE ENGINE

React

Erste Kontaktaufnahme

- Referenzimplementierung: Ozark (https://ozark.java.net/)
- Maven Archetype (https://github.com/making/mvc-1.0-blank)
- Script zur Erstellung eines Docker Image
  => Starten von Ozark in Docker Container
- Beispielanwendung von Christian Kaltepoth (https://github.com/chkal/todo-mvc)

© 2016 Thilo Frotscher

Zusammenfassung

- Java EE 8 erhält ein neues, actionbasiertes Web-Framework namens MVC
- Alternative (kein Ersatz!) für JSF
  - JSF bleibt wichtiger Bestandteil von Java EE
- Gemeinsamkeiten mit JSF
  - Modell: CDI
  - Validierung: Bean Validation
  - Verbindung zwischen View und Model: EL
  - Unterstützte View-Technologien: JSP & Facelets

© 2016 Thilo Frotscher

Zusammenfassung

- MVC basiert stark auf existierenden Technologien
  - JAX-RS, CDI, Bean Validation, JSP, Facelets,
  - Einstieg sollte vergleichsweise leicht gelingen
- Referenzimplementierung: Ozark
- Bereits heute zahlreiche View Engines verfügbar
- Stand: Early Draft! (2nd Edition, 02.10.2015)
  - Q1 2016 Public Review (?!?)
  - Q3 2016 Proposed Final Draft
  - H1 2017 Final Release

© 2016 Thilo Frotscher

Thilo Frotscher

**Kundenspezifisches
Training und Beratung zu
Java EE, Services & Systemintegration**

thilo@frotscher.com

@thfro

© 2016 Thilo Frotscher