MVC 1.0

Das neue Webframework in Java EE 8

Christian Kaltepoth / @chkal

Christian Kaltepoth
Senior Developer @ ingenit

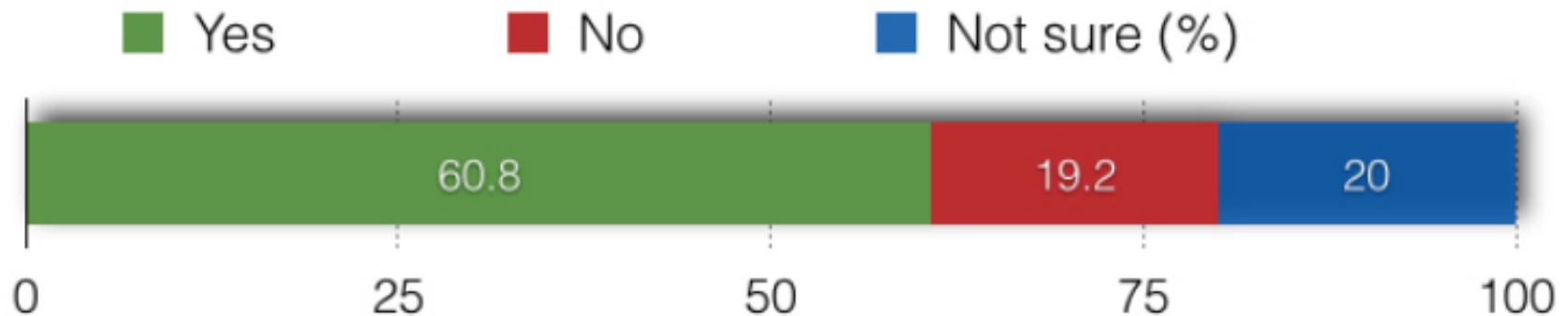christian@kaltepoth.de / @chkal

http://blog.kaltepoth.de

Warum
MVC 1.0?

Java EE 8 Community Survey

Should Java EE provide support for MVC, alongside JSF?

■ Yes　■ No　■ Not sure (%)

| 60.8 | 19.2 | 20 |

0　25　50　75　100

JavaServer Faces

vs.

MVC 1.0

JavaServer Faces

=

Component Oriented

MVC 1.0

=

Action Oriented

Component Oriented
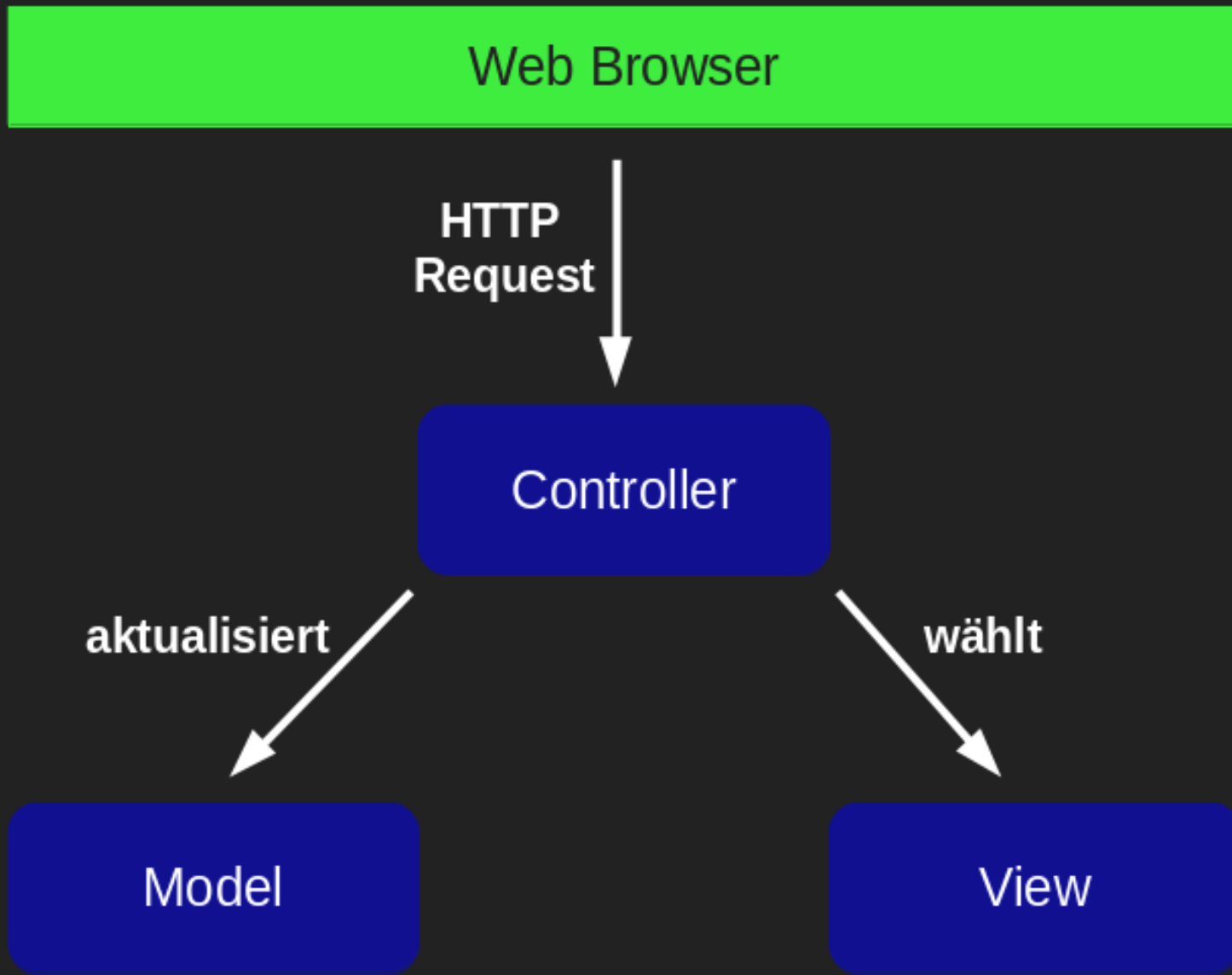
vs.

Action Oriented

Das MVC Entwurfsmuster

Controller

Model

View
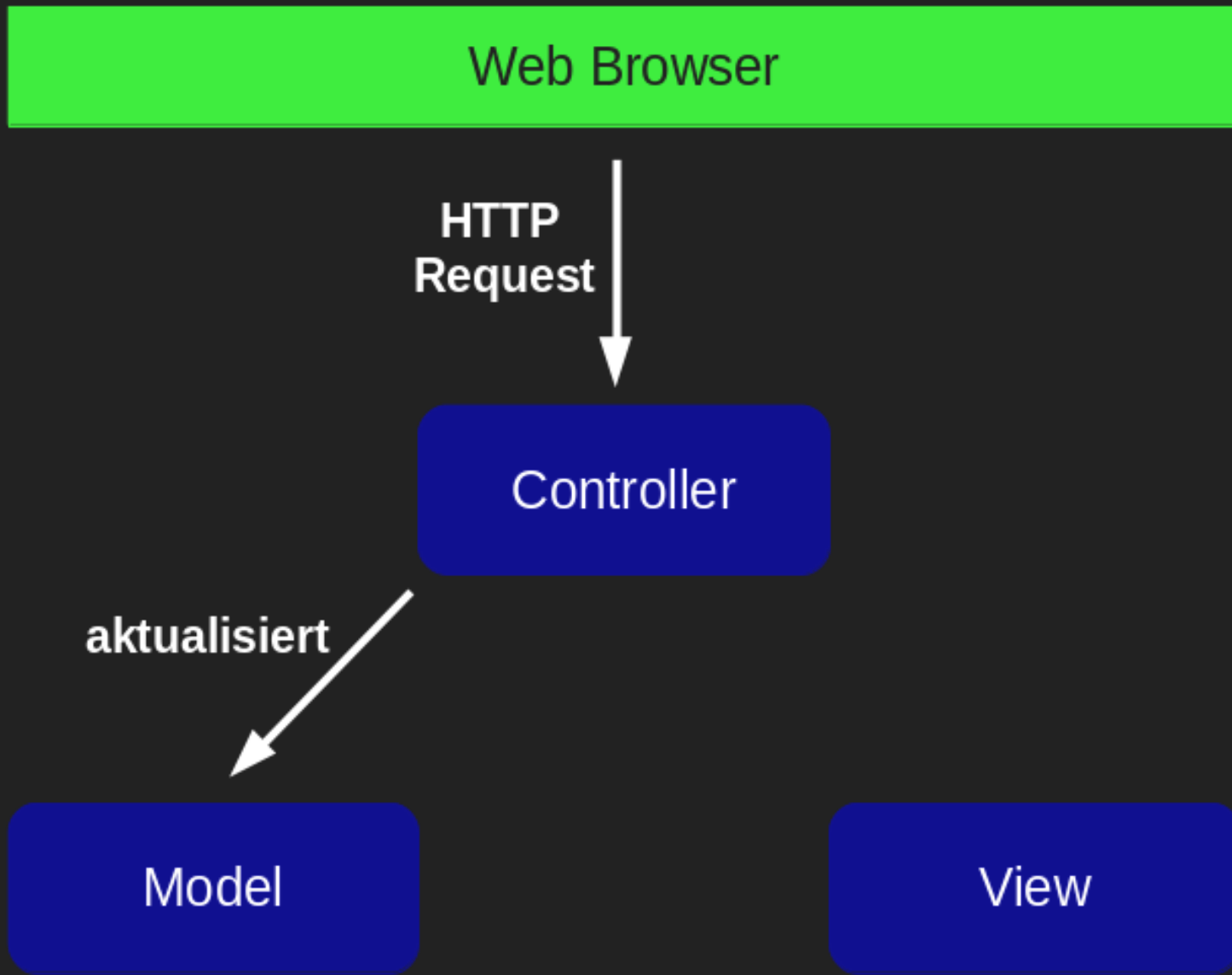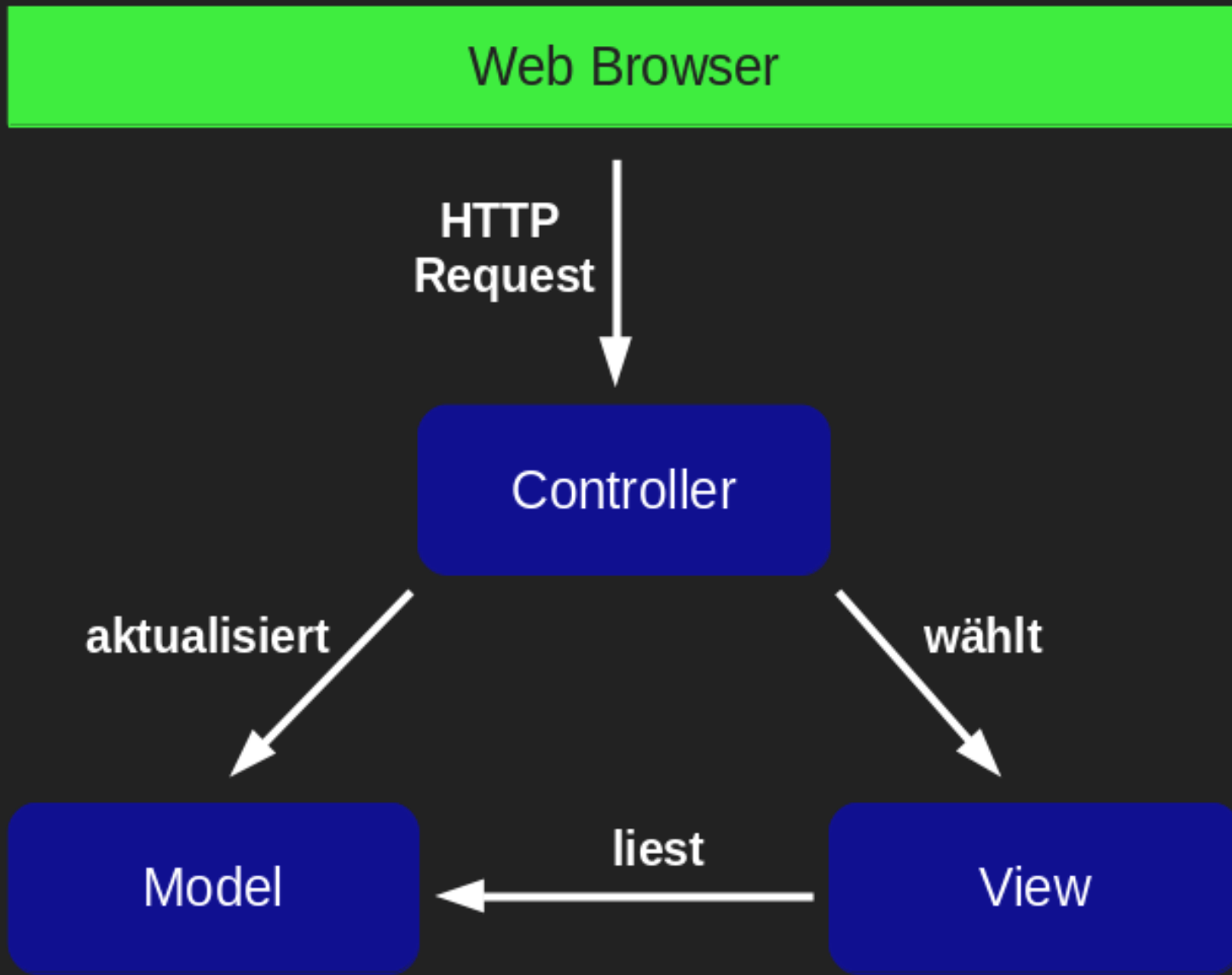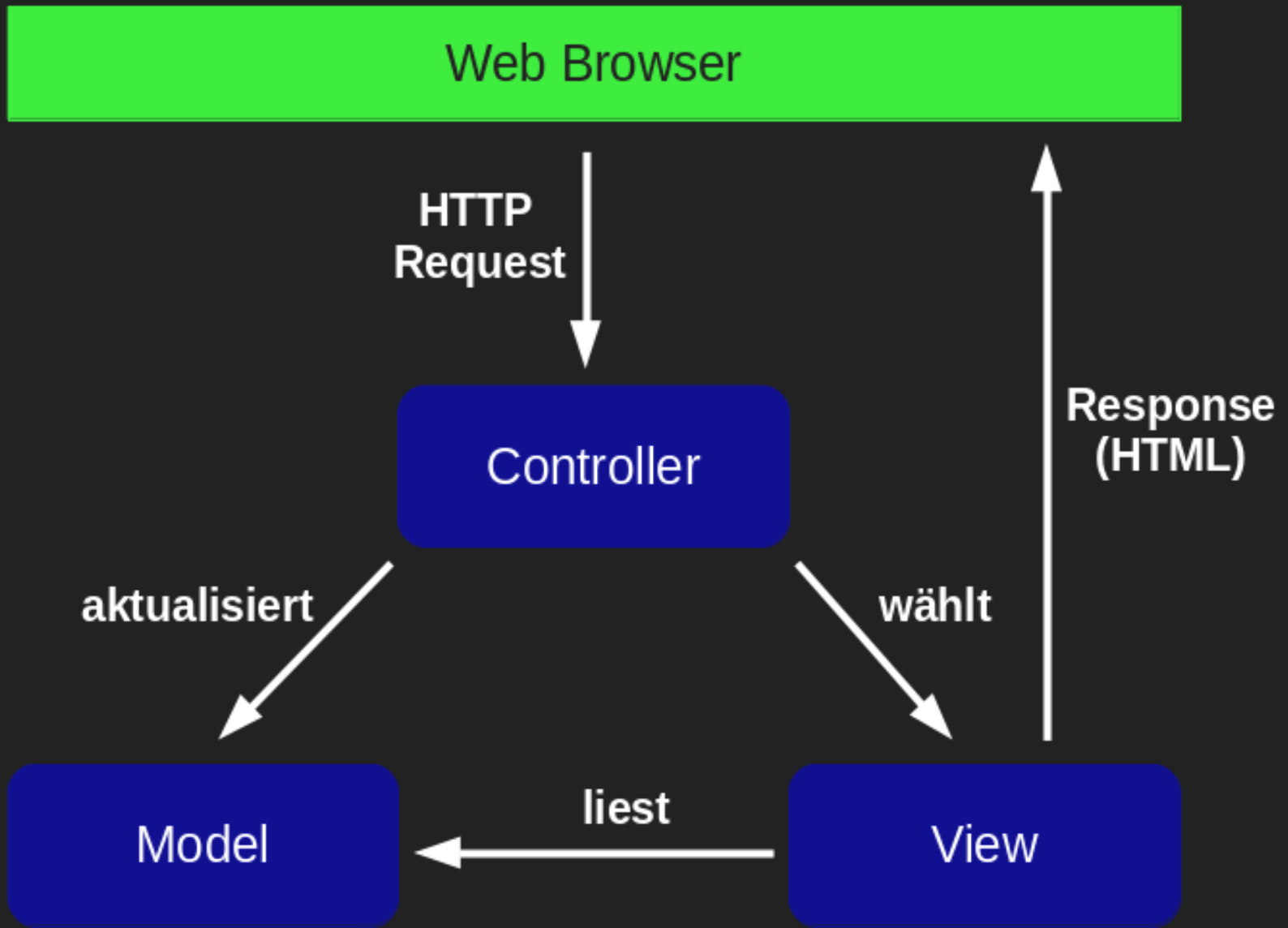
MVC 1.0

basiert auf

Java EE 8

MVC im Kontext

MVC 1.0

JAX-RS

CDI

Servlet

Bean Validation

JAX-RS
(in < 3 Minuten)

JAX-RS Fakten

- "Java API for RESTful Web Services"
- Erstes Release 2008
- Seit 1.1 Bestandteil von Java EE 6
- JAX-RS 2.1 -> JSR 370 -> Java EE 8

JAX-RS Beispiel

```java
@Path("/hello")
public class HelloResource {

  @GET
  public String greet() {
    return "Hello world";
  }

}
```

JAX-RS Beispiel

```java
@Path("/hello")
public class HelloResource {

  @GET
  public String greet( @QueryParam("name") String name ) {
    return "Hello " + name;
  }

}
```

Hello World

mit

MVC 1.0

Controller

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

  @GET
  public String render() {
    return "helloworld.jsp";
  }

}
```

View

```html
<!DOCTYPE html>
<html>
  <head>
    <title>MVC Demo</title>
  </head>
  <body>
    <h1>Hello world</h1>
  </body>
</html>
```

Controller

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

  @GET
  @View("helloworld.jsp")
  public void render() {
    // ...
  }

}
```

Das Modell

- javax.mvc.Models
- Basierend auf CDI

javax.mvc.Models

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

  @Inject
  private Models models;

  @GET
  public String greet() {
    models.put( "message", "Hello world!" );
    return "helloworld.jsp";
  }

}
```

javax.mvc.Models

```html
<!DOCTYPE html>
<html>
  <head>
    <title>MVC Demo</title>
  </head>
  <body>
    <h1>${message}</h1>
  </body>
</html>
```

CDI Models

```java
@Named
@RequestScoped
public class Greeting {

    private String message;

    public String getMessage() {
        return message;
    }

    public void setMessage( String message ) {
        this.message = message;
    }

}
```

CDI Models

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

  @Inject
  private Greeting greeting;

  @GET
  public String greet() {
    greeting.setMessage( "Hello world!" );
    return "helloworld.jsp";
  }

}
```

CDI Models

```html
<!DOCTYPE html>
<html>
  <head>
    <title>MVC Demo</title>
  </head>
  <body>
    <h1>${greeting.message}</h1>
  </body>
</html>
```

Views in MVC 1.0

- JavaServer Pages
- Facelets

JSP als View Technologie

```java
models.put( "messages", Arrays.asList(
  "Hello W-JAX 2015",
  "MVC 1.0 rocks"
) );
```

```jsp
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<c:if test="${not empty messages}">
  <ul>
    <c:forEach var="message" items="${messages}">
      <li>${message}</li>
    </c:forEach>
  </ul>
</c:if>
```

ViewEngine SPI

Custom ViewEngines

- Thymeleaf
- FreeMarker
- Velocity
- Handlebars
- Mustache
- StringTemplate
- Jade
- AsciiDoc
- JSR223
- React

Beispiel: Thymeleaf

Thymeleaf

```java
@Controller
@Path("/thymeleaf")
public class ThymeleafController {

  @Inject
  private Models models;

  @GET
  public String render() {

    models.put( "messages", Arrays.asList(
      "Text #1", "Text #2", "Text #3"
    ) );

    return "thymeleaf.html";

  }

}
```

Thymeleaf

/WEB-INF/views/thymeleaf.html

```html
<!-- ... -->

<table>
  <tr th:each="msg: ${messages}">
    <td th:text="${msg}">
      Some text
    </td>
  </tr>
</table>

<!-- ... -->
```

```html
<table>
  <tr>
    <td>Text #1</td>
  </tr>
  <tr>
    <td>Text #2</td>
  </tr>
  <tr>
    <td>Text #3</td>
  </tr>
</table>
```

Formulare

Einfache Forms

/WEB-INF/views/form.jsp

```html
<form action="./form" method="POST">

    Bitte geben Sie Ihren Namen ein:
    <input type="text" name="name"/>

    <input type="submit" value="Absenden"/>

</form>
```

Einfache Forms

```
<form action="${mvc.basePath}/form" method="POST">

    Bitte geben Sie Ihren Namen ein:
    <input type="text" name="name"/>

    <input type="submit" value="Absenden"/>

</form>
```

Einfache Forms

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

    @Inject
    private Models models;

    @POST
    public String post( @FormParam("name") String name ) {

        models.put( "message", "Hello " + name );
        return "form.jsp";

    }

}
```

Komplexe Forms

```java
public class HelloForm {

    @FormParam("name")
    private String name;

    @FormParam("age")
    private Integer age;

    @FormParam("address")
    private String address;

    /* getter + setter */

}
```

Komplexe Forms

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

    @Inject
    private Models models;

    @POST
    public String post( @BeanParam HelloForm form ) {

        models.put( "message", "Hello " + form.getName() );
        return "form.jsp";

    }

}
```

Validation

JSR 303
Bean Validation

Validierung

```java
public class HelloForm {

    @FormParam("name")
    @Size(min = 3, message = "Geben Sie Ihren Namen ein")
    private String name;

    @FormParam("age")
    @NotNull(message = "Geben Sie Ihr Alter ein")
    @Min(value = 18, message = "Sie müssen 18 Jahre sein")
    private Integer age;

    /* getter + setter */

}
```

Validierung

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

  @Inject
  private BindingResult bindingResult;

  @POST
  public String post( @BeanParam @Valid ValidationForm form ) {

    if( bindingResult.isFailed() ) {
      models.put( "messages", bindingResult.getAllMessages() );
      return "form.jsp";
    }

    // Verarbeitung des Forms...
  }

}
```

Security

CSRF

Cross Site Request Forgery

CSRF Beispiel

https://www.example.com/tweet

```html
<form action="/tweet" method="POST">

   Bitte geben Sie einen Text ein:
   <input type="text" name="status"/>

   <input type="submit" value="Absenden"/>

</form>
```

CSRF Beispiel

http://www.bad-guys.com/

```html
<form id="form" style="display: none;" method="POST"
      action="https://www.example.com/tweet">
  <input type="text" name="status"
         value="PHP ist die beste Sprache der Welt!"/>
</form>
```

```html
<a href="javascript:void(0)"
   onclick="document.getElementById('form').submit();">
  Gratis iPhone
</a>
```

Page Token Pattern

- Server erstellt geheimes Token
- Token wird in Hidden-Field gerendert
- Prüfung des Token bei Submit

MVC 1.0 CSRF Modes

OFF	Kein Prüfung (Default)
EXPLICIT	Prüfung des Tokens mit @CsrfValid
IMPLICIT	Prüfung bei jedem POST-Request

CSRF Beispiel

```html
<form action="/tweet" method="POST">

    <!-- CSRF Page Token -->
    <input type="hidden" name="${mvc.csrf.name}"
        value="${mvc.csrf.token}"/>

    Bitte geben Sie einen Text ein:
    <input type="text" name="status"/>

    <input type="submit" value="Absenden"/>

</form>
```

Prüfung mit @CsrfValid

```java
@Controller
@Path("/tweet")
public class TweetController {

  /* ... */

  @POST
  @CsrfValid
  public String post( @FormParam("status") String s ) {

    /* ... */

  }

}
```

Empfehlung

IMPLICIT verwenden!

MVC kann mehr...

- Einfache Redirects aus Controller
- CDI Scope: @RedirectScoped
- MvcContext / #{mvc}
- CDI Events
- ViewEngine SPI
- HTML/JS Encoding/Escaping

Vollständiges Beispiel

https://github.com/chkal/todo-mvc/

MVC 1.0 TODO

- Internationalisierung / Lokalisierung
- Typsicheres Erstellen von URLs
- ...

Zeitplan

- Q3 2014 Expert Group formed
- Q1 2015 Early Draft
- Q4 2015 Early Draft 2
- Q1 2016 Public Review
- Q3 2016 Proposed Final Draft
- H1 2017 Final Release

Feedback erwünscht

Mailinglist	https://java.net/projects/mvc-spec/lists
JIRA	http://java.net/jira/browse/MVC_SPEC
Ozark (RI)	https://ozark.java.net/

Danke!

Fragen?

https://github.com/chkal/wjax15-mvc
https://github.com/chkal/todo-mvc

Christian Kaltepoth / @chkal