MVC 1.0

Endlich am Ziel

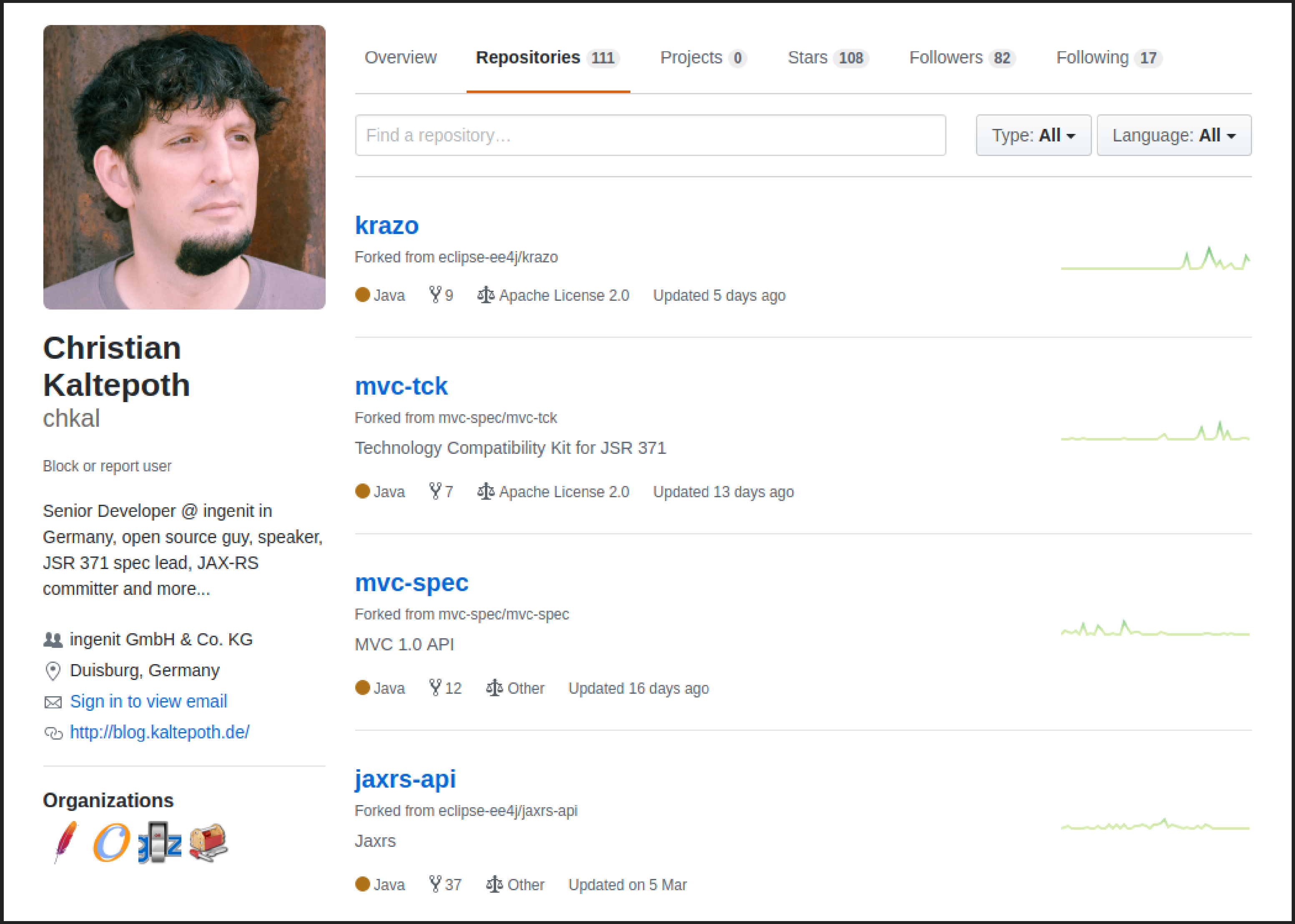
Christian Kaltepoth / @chkal

MVC 1.0

~~Endlich~~ Fast am Ziel

Christian Kaltepoth / @chkal

Über mich...



Overview | **Repositories** 111 | Projects 0 | Stars 108 | Followers 82 | Following 17

Find a repository… | Type: **All** | Language: **All**

Christian Kaltepoth
chkal

Block or report user

Senior Developer @ ingenit in Germany, open source guy, speaker, JSR 371 spec lead, JAX-RS committer and more...

ingenit GmbH & Co. KG
Duisburg, Germany
Sign in to view email
http://blog.kaltepoth.de/

Organizations

krazo
Forked from eclipse-ee4j/krazo
● Java 9 Apache License 2.0 Updated 5 days ago

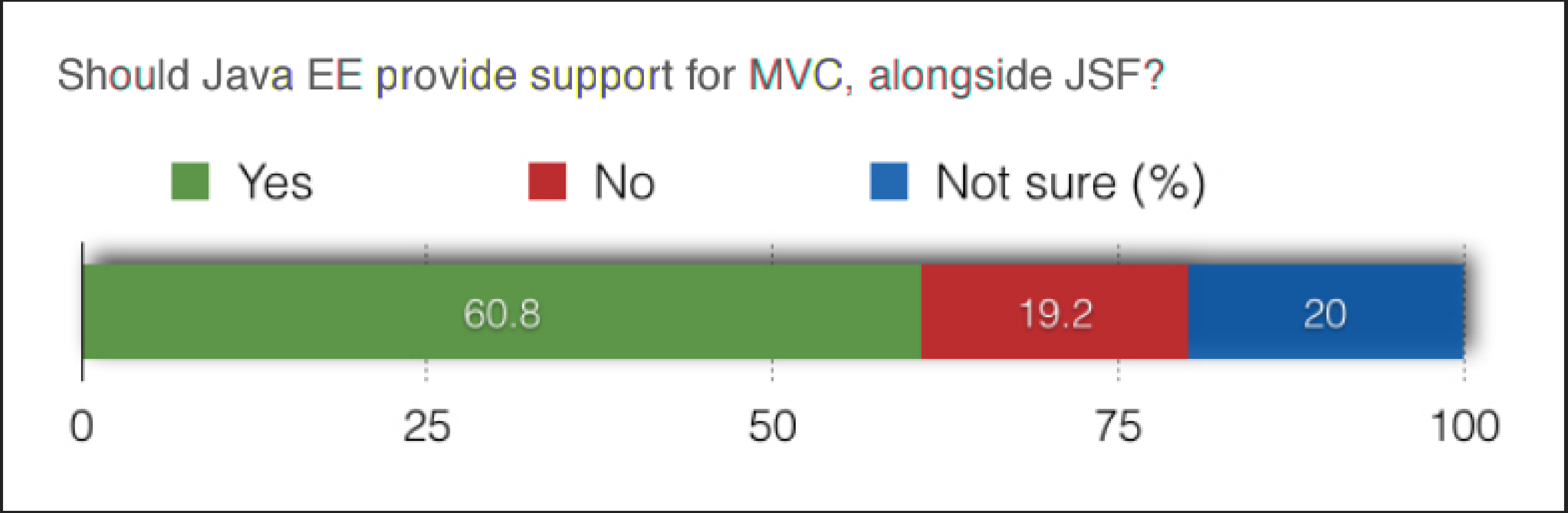
mvc-tck
Forked from mvc-spec/mvc-tck
Technology Compatibility Kit for JSR 371
● Java 7 Apache License 2.0 Updated 13 days ago

mvc-spec
Forked from mvc-spec/mvc-spec
MVC 1.0 API
● Java 12 Other Updated 16 days ago

jaxrs-api
Forked from eclipse-ee4j/jaxrs-api
Jaxrs
● Java 37 Other Updated on 5 Mar

Warum MVC 1.0?

Java EE 8 Community Survey



https://java.net/downloads/javaee-spec/JavaEE8_Community_Survey_Results.pdf

JavaServer Faces

vs.

MVC 1.0

JavaServer Faces

=

Component Oriented

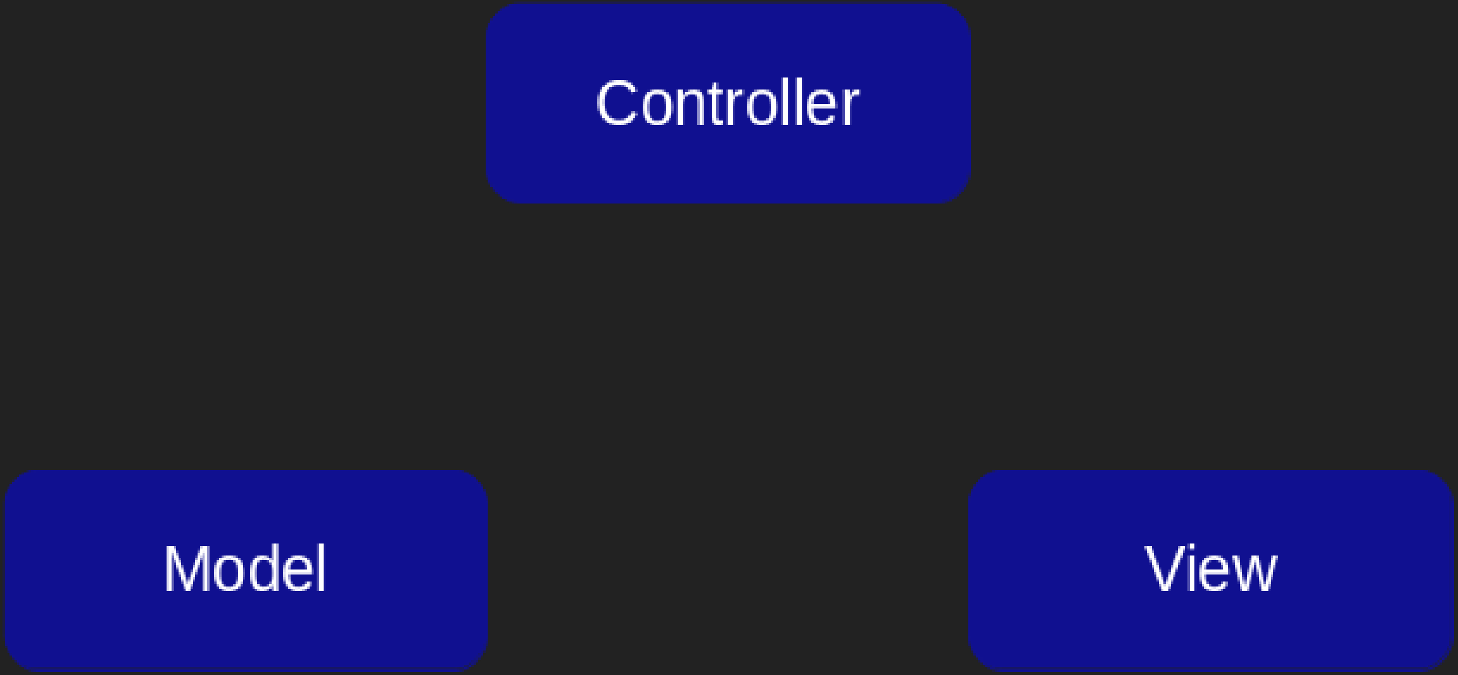
MVC 1.0

=

Action Oriented

Component Oriented
vs.
Action Oriented

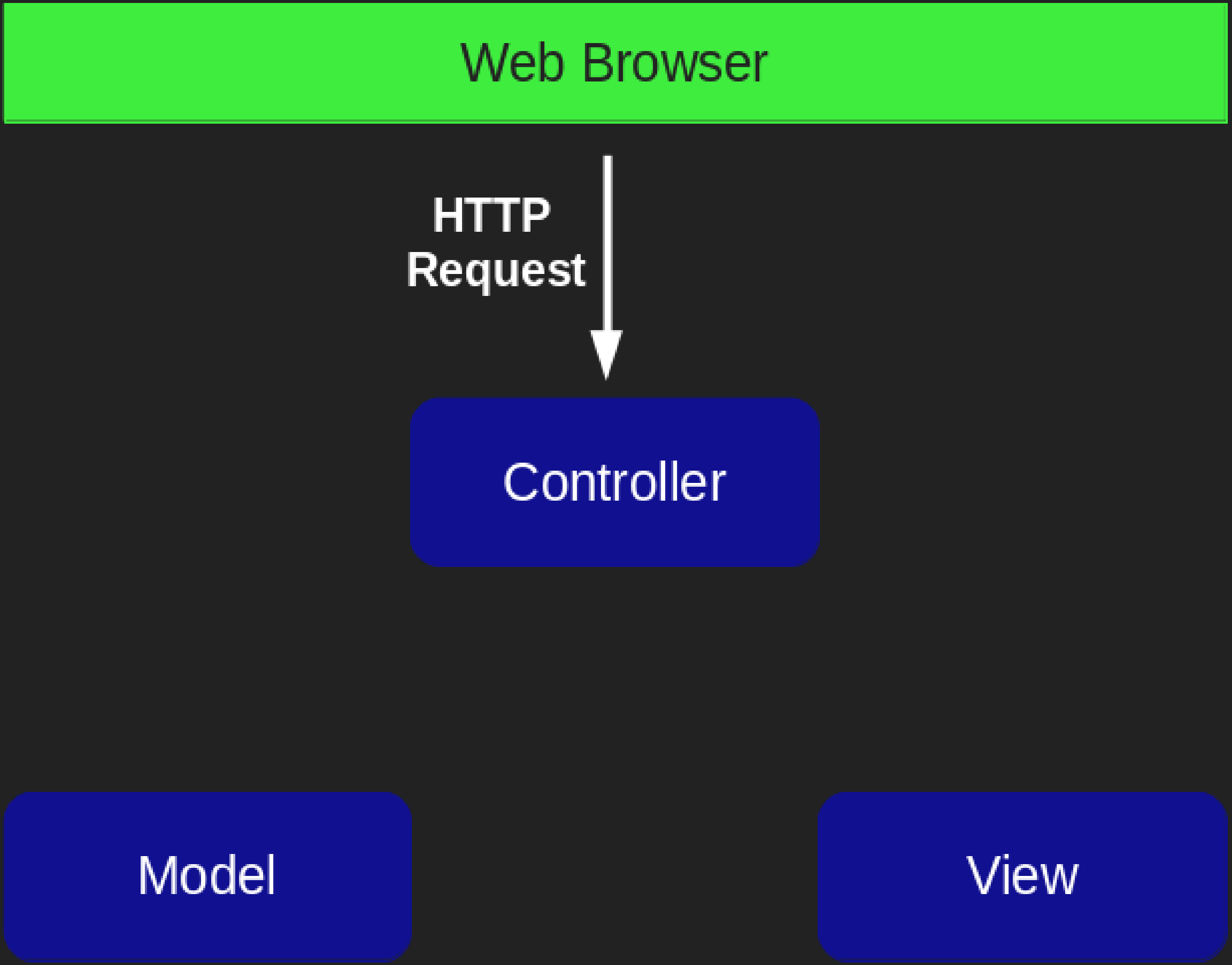
Das MVC Entwurfsmuster



Controller

Model

View



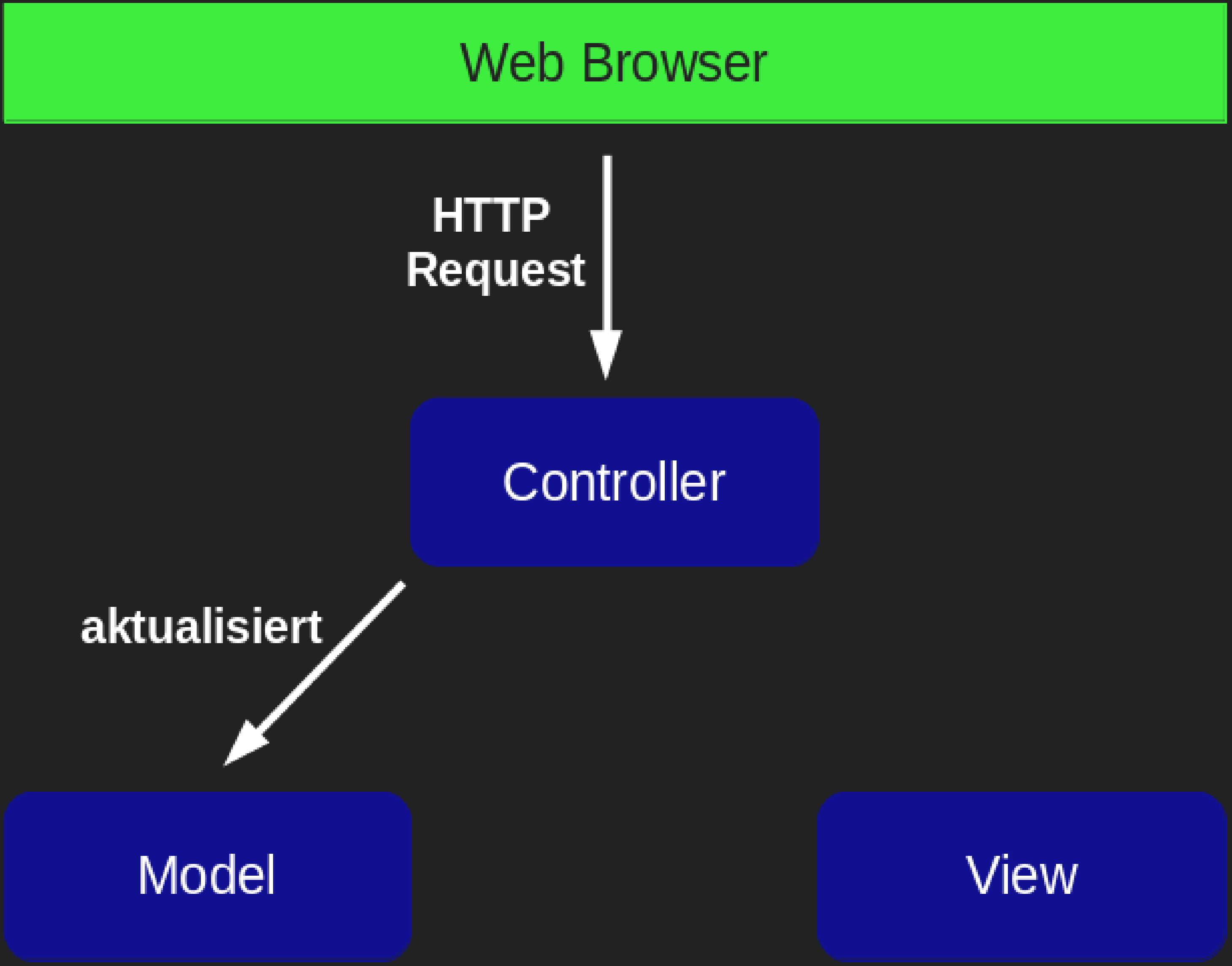
Web Browser

HTTP Request

Controller

Model

View



Web Browser

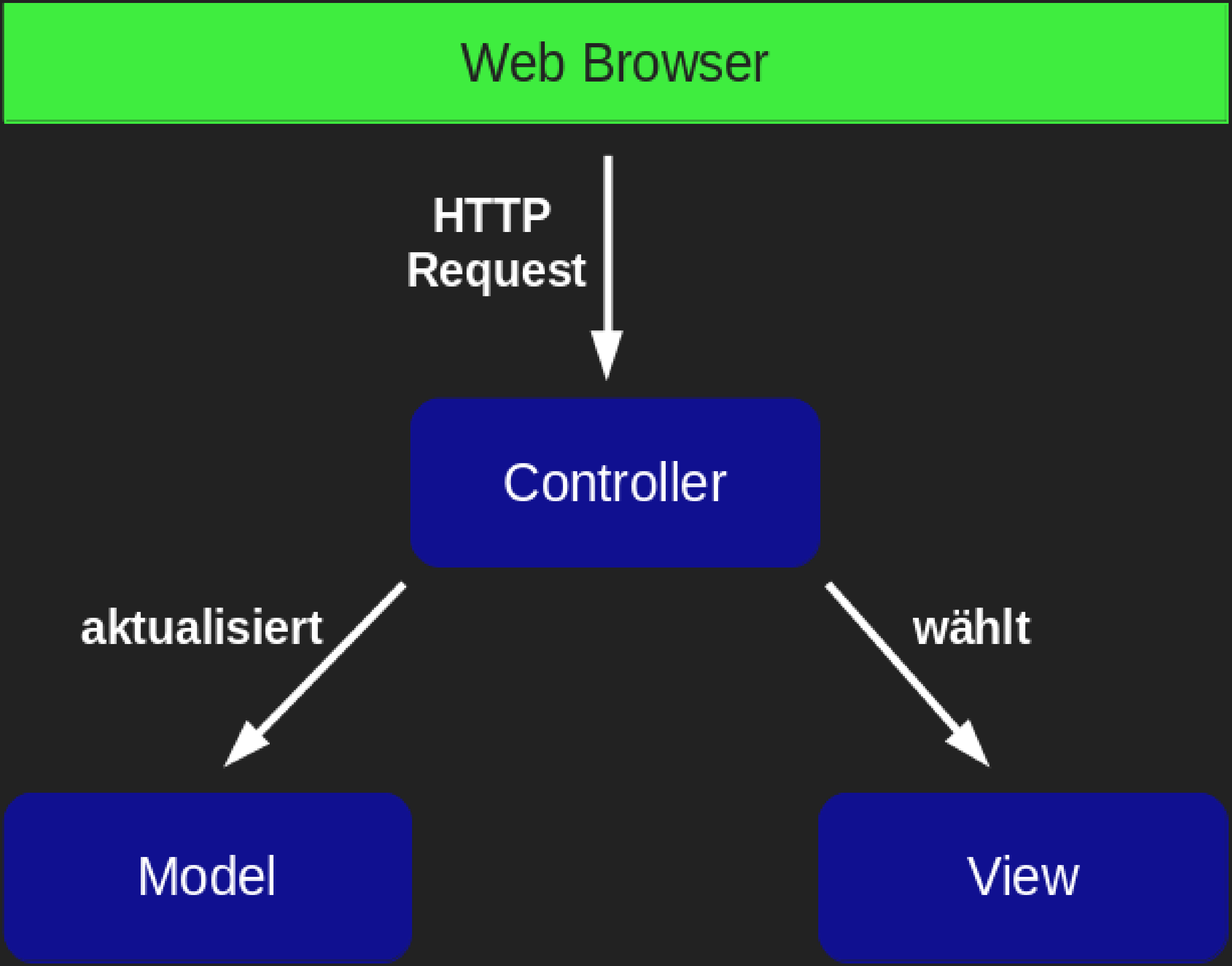
HTTP Request

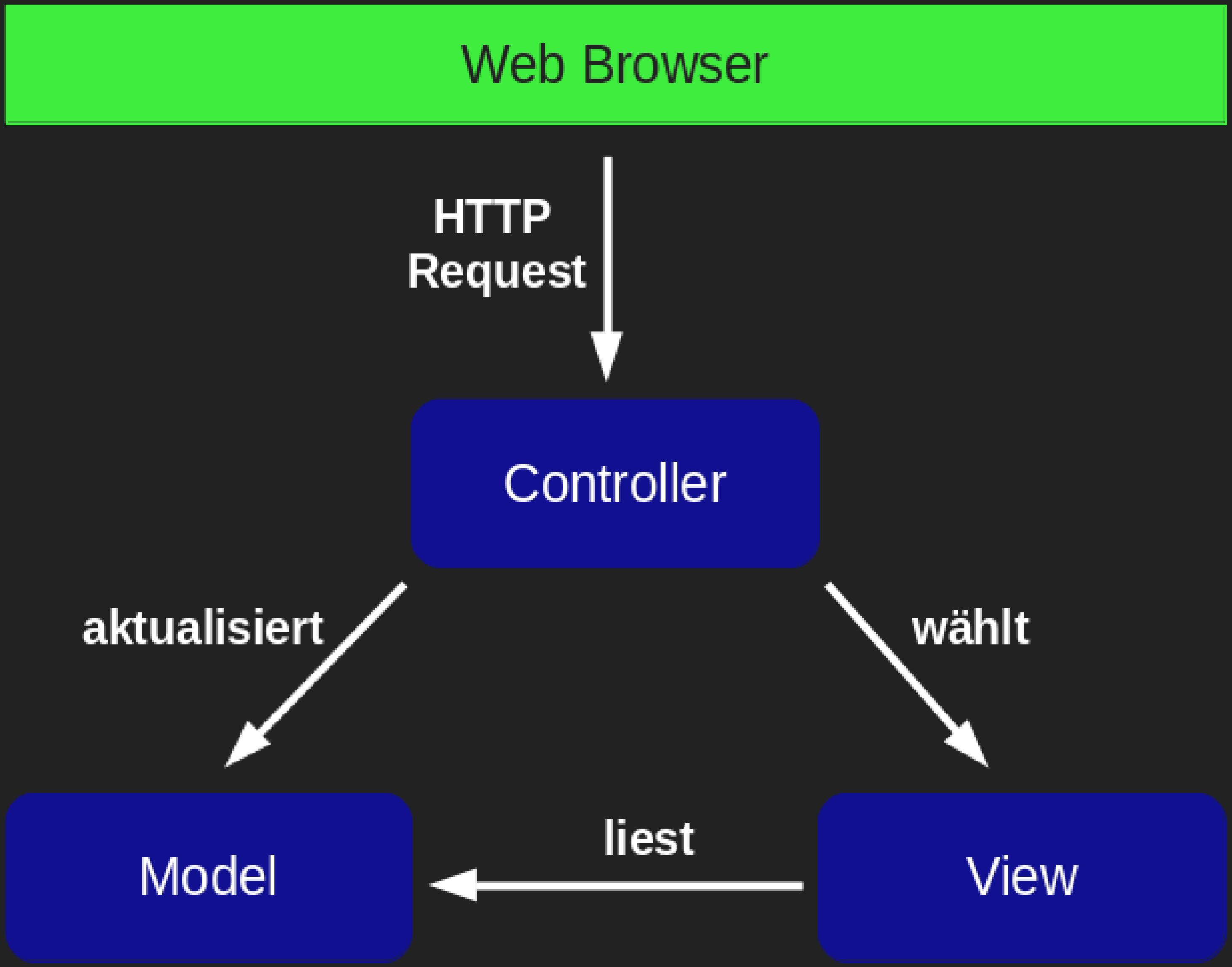
Controller

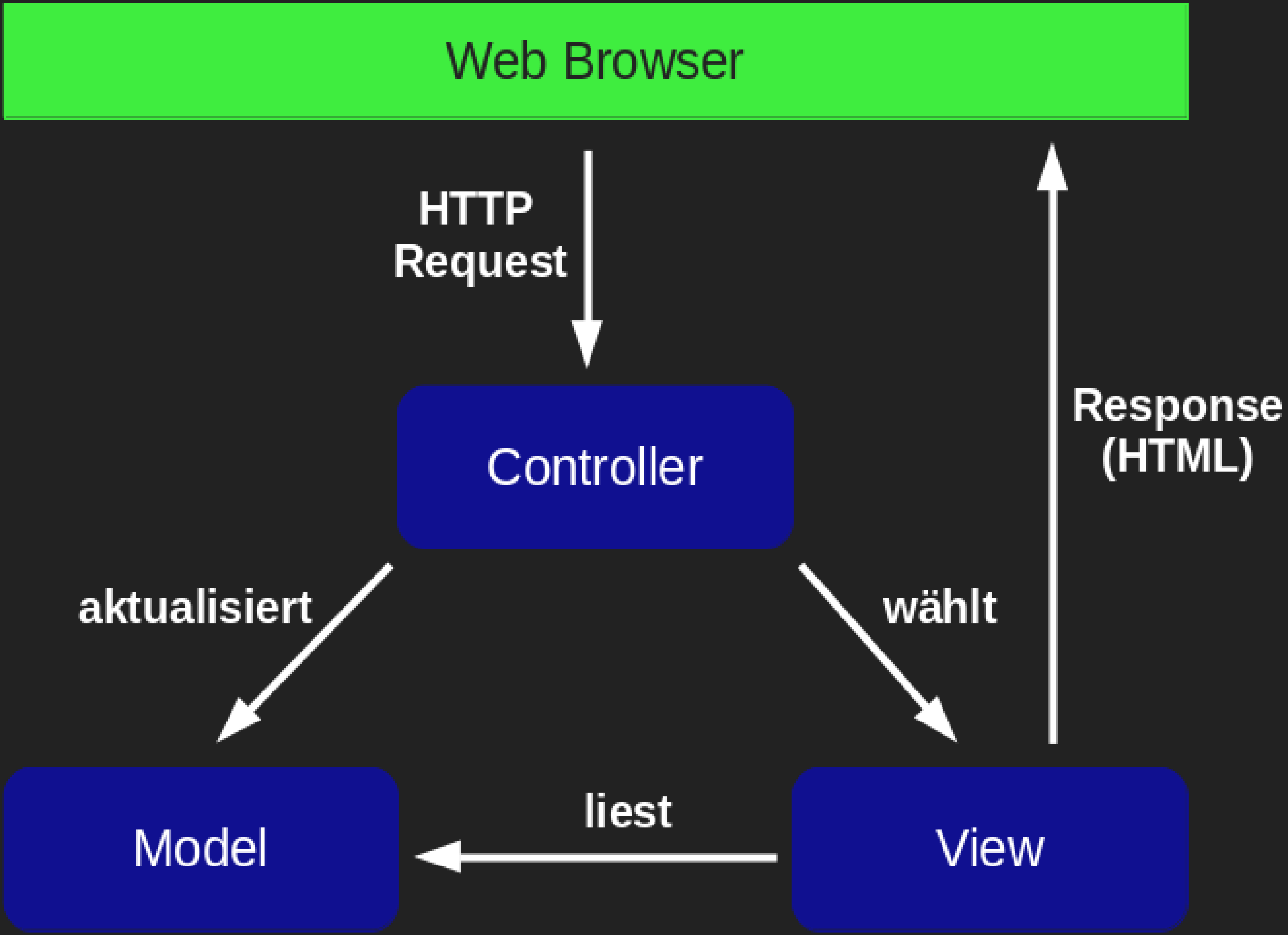
aktualisiert

Model

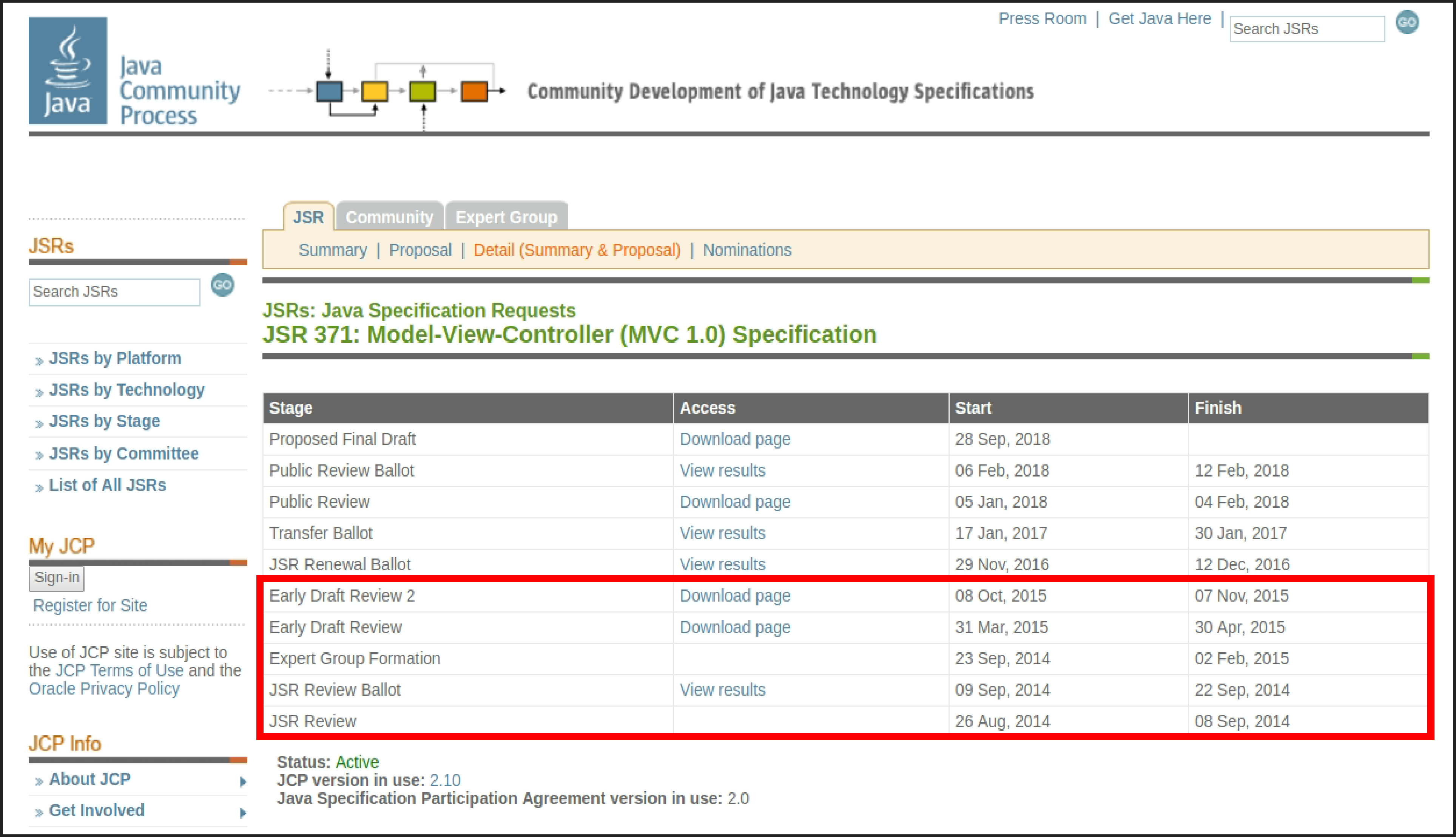
View


Web Browser
HTTP Request
Controller
aktualisiert
wählt
Model
View


Web Browser
HTTP Request
Controller
aktualisiert
wählt
liest
Model
View


Web Browser
HTTP Request
Response (HTML)
Controller
aktualisiert
wählt
Model
liest
View

JSR 371



Use of JCP site is subject to the JCP Terms of Use and the Oracle Privacy Policy

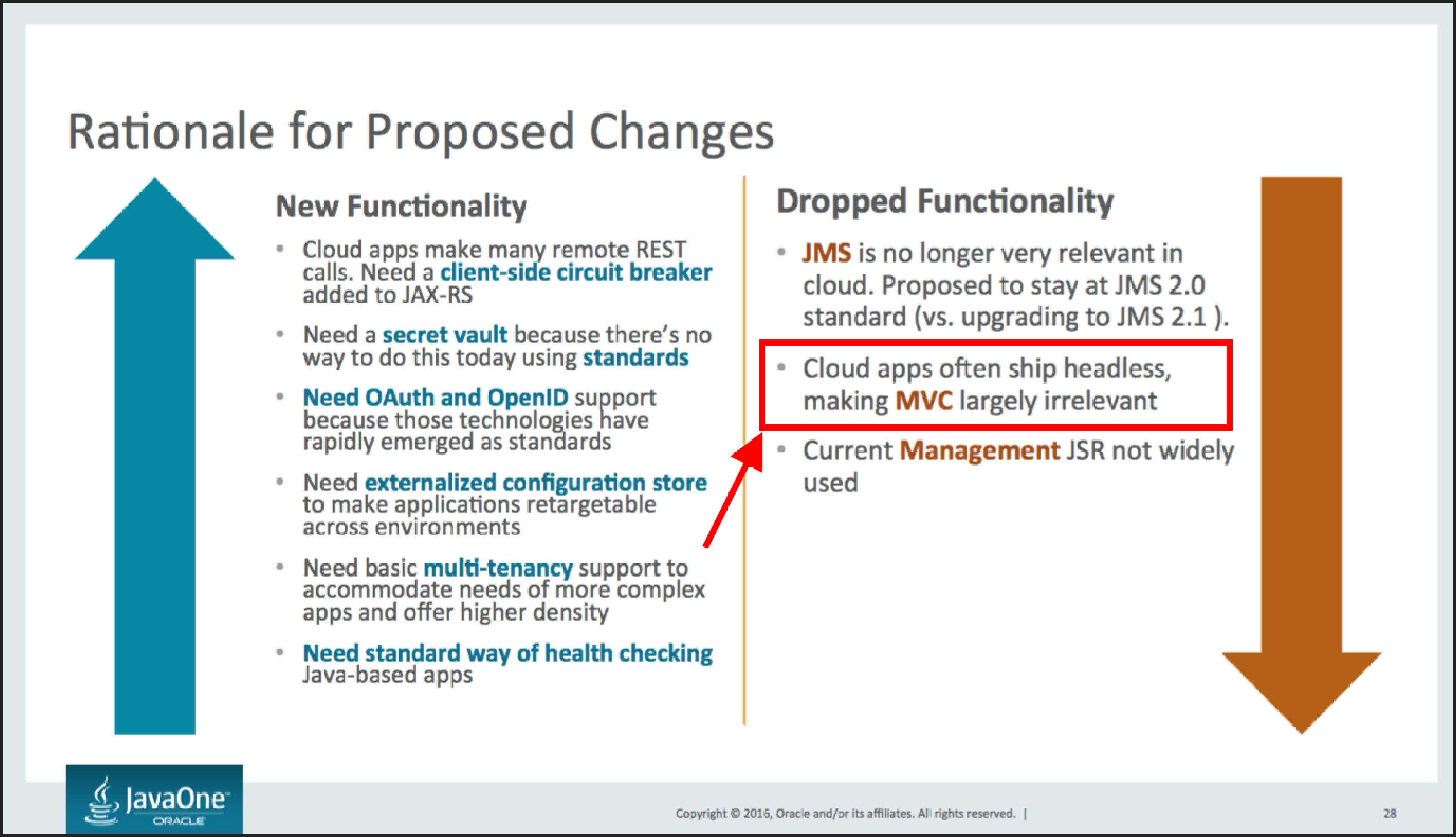
JSRs: Java Specification Requests

JSR 371: Model-View-Controller (MVC 1.0) Specification

Stage	Access	Start	Finish
Proposed Final Draft	Download page	28 Sep, 2018	
Public Review Ballot	View results	06 Feb, 2018	12 Feb, 2018
Public Review	Download page	05 Jan, 2018	04 Feb, 2018
Transfer Ballot	View results	17 Jan, 2017	30 Jan, 2017
JSR Renewal Ballot	View results	29 Nov, 2016	12 Dec, 2016
Early Draft Review 2	Download page	08 Oct, 2015	07 Nov, 2015
Early Draft Review	Download page	31 Mar, 2015	30 Apr, 2015
Expert Group Formation		23 Sep, 2014	02 Feb, 2015
JSR Review Ballot	View results	09 Sep, 2014	22 Sep, 2014
JSR Review		26 Aug, 2014	08 Sep, 2014

Status: Active
JCP version in use: 2.10
Java Specification Participation Agreement version in use: 2.0

JavaOne 2016



Rationale for Proposed Changes

New Functionality

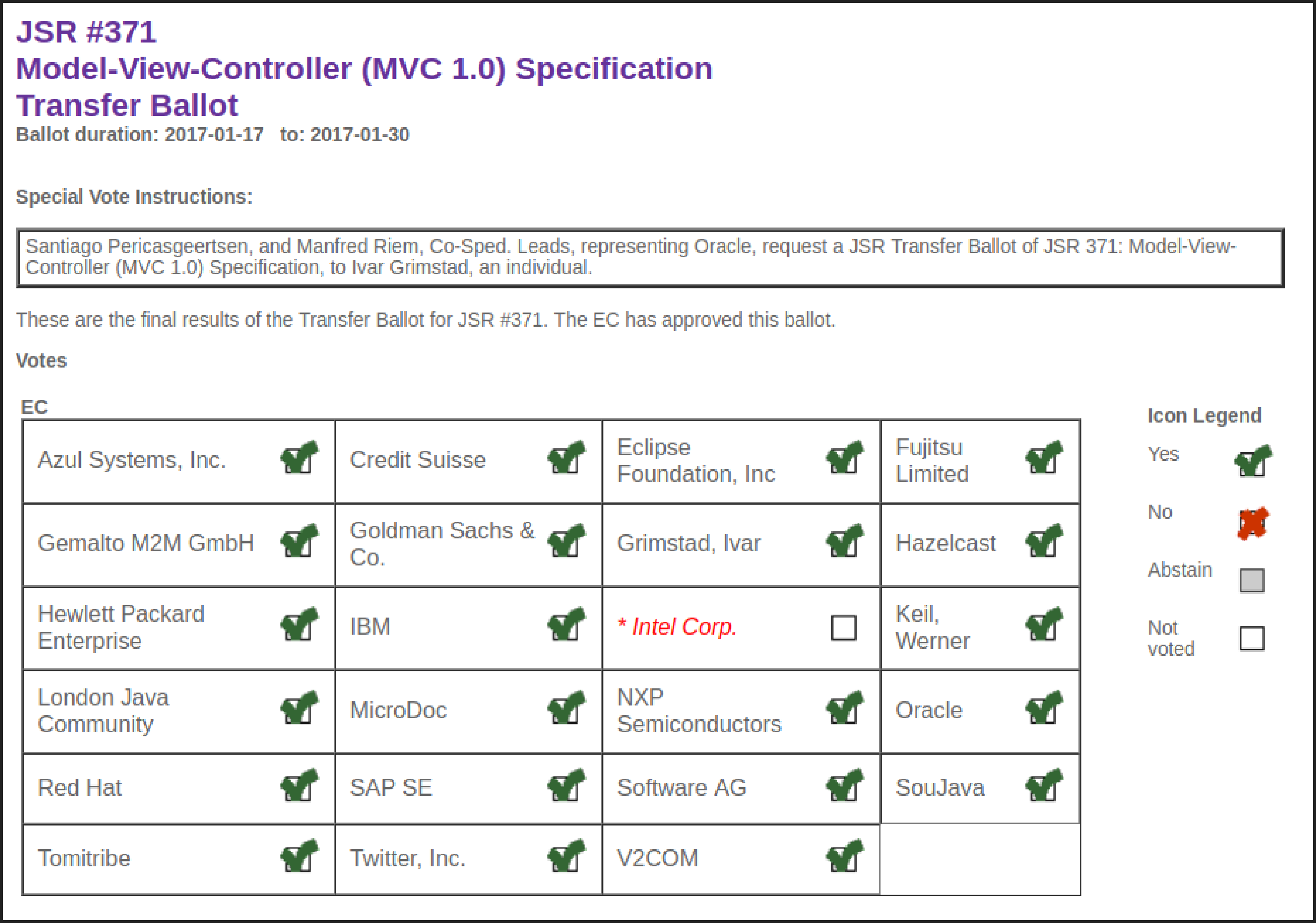
- Cloud apps make many remote REST calls. Need a **client-side circuit breaker** added to JAX-RS
- Need a **secret vault** because there's no way to do this today using **standards**
- **Need OAuth and OpenID** support because those technologies have rapidly emerged as standards
- Need **externalized configuration store** to make applications retargetable across environments
- Need basic **multi-tenancy** support to accommodate needs of more complex apps and offer higher density
- **Need standard way of health checking** Java-based apps

Dropped Functionality

- **JMS** is no longer very relevant in cloud. Proposed to stay at JMS 2.0 standard (vs. upgrading to JMS 2.1).
- Cloud apps often ship headless, making **MVC** largely irrelevant
- Current **Management** JSR not widely used

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. |

Oracle → Community



JSR #371
Model-View-Controller (MVC 1.0) Specification
Transfer Ballot

Ballot duration: 2017-01-17   to: 2017-01-30

Special Vote Instructions:

Santiago Pericasgeertsen, and Manfred Riem, Co-Sped. Leads, representing Oracle, request a JSR Transfer Ballot of JSR 371: Model-View-Controller (MVC 1.0) Specification, to Ivar Grimstad, an individual.

These are the final results of the Transfer Ballot for JSR #371. The EC has approved this ballot.

Votes

EC

Azul Systems, Inc. ✓	Credit Suisse ✓	Eclipse Foundation, Inc ✓	Fujitsu Limited ✓
Gemalto M2M GmbH ✓	Goldman Sachs & Co. ✓	Grimstad, Ivar ✓	Hazelcast ✓
Hewlett Packard Enterprise ✓	IBM ✓	* *Intel Corp.* □	Keil, Werner ✓
London Java Community ✓	MicroDoc ✓	NXP Semiconductors ✓	Oracle ✓
Red Hat ✓	SAP SE ✓	Software AG ✓	SouJava ✓
Tomitribe ✓	Twitter, Inc. ✓	V2COM ✓	

Icon Legend

Yes	✓
No	✗
Abstain	▨
Not voted	□

Was passierte dann?

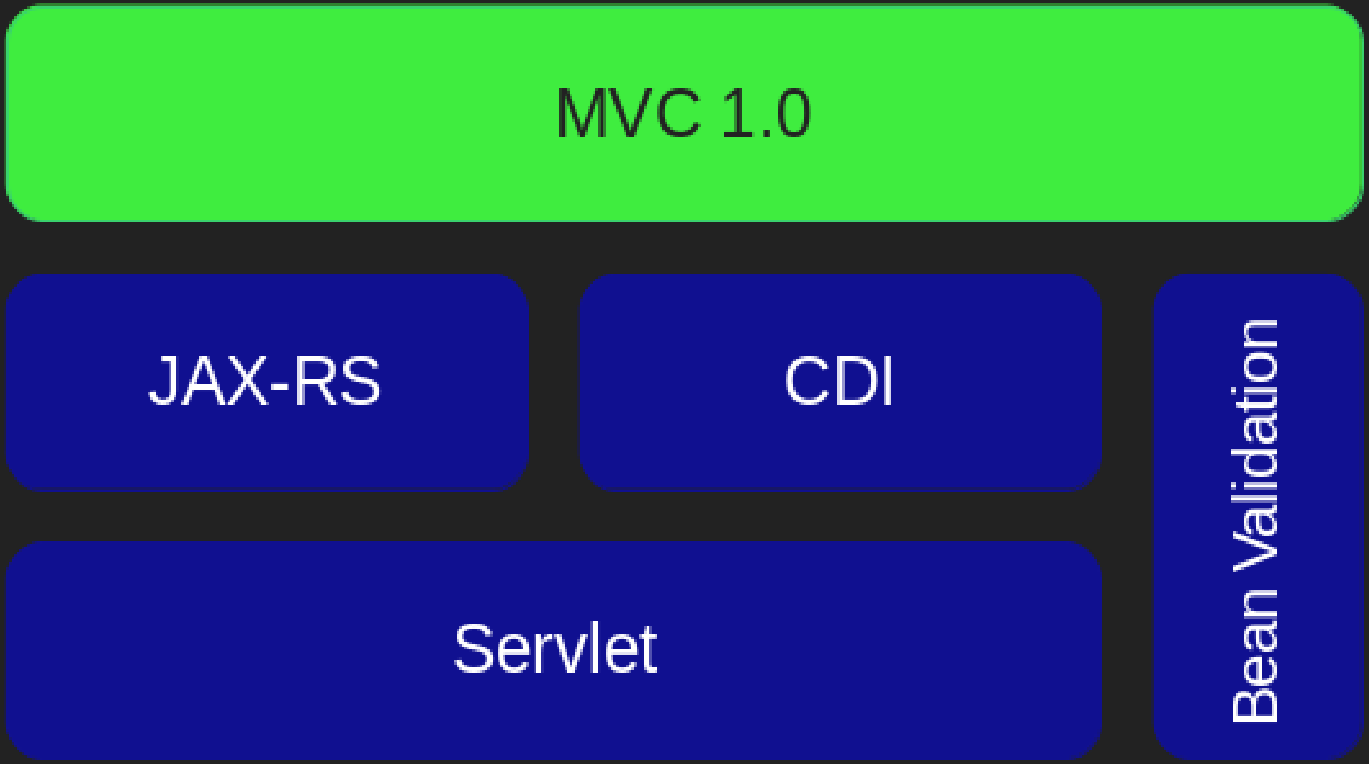
- Apache License 2.0
- Neue Infrastruktur (GitHub, CI, Webseite)
- Arbeit an Referenzimplementierung
- Erstellung des TCK
- Java Community Process:
  - Public Review (PR)
  - Proposed Final Draft (PFD)

MVC 1.0

basiert auf

Java EE 8

MVC im Kontext



JAX-RS

(in < 3 Minuten)

JAX-RS Beispiel

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

  @GET
  public JsonObject greet() {
    return Json.createObjectBuilder()
               .add("message", "Hello JAX-RS!");
  }

}
```

JAX-RS Beispiel

```
@Path("/hello")
@Produces("application/json")
public class HelloResource {

  @GET
  @Path("/{name}")
  public JsonObject greet( @PathParam("name") String name ) {
    return Json.createObjectBuilder()
               .add("message", "Hello " + name + "!");
  }

}
```

Hello World

mit

MVC 1.0

Controller

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

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

}
```

View

/WEB-INF/views/helloworld.jsp

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

Controller

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

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

}
```

Das Modell

- javax.mvc.Models
- Basierend auf CDI

javax.mvc.Models

```
@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

/WEB-INF/views/helloworld.jsp

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

CDI Models

```
@Named
@RequestScoped
public class Greeting {

  private String message;

  public String getMessage() {
    return message;
  }

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

}
```

CDI Models

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

  @Inject
  private Greeting greeting;

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

}
```

CDI Models

/WEB-INF/views/helloworld.jsp

```
<!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

```
models.put( "messages", Arrays.asList(
  "Hello JAX 2019",
  "MVC 1.0 rocks"
) );
```

```
<%@ 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

```
@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

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

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

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

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

Formulare

Formulare

/WEB-INF/views/form.jsp

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

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

  Bitte geben Sie Ihr Alter ein:
  <input type="text" name="age">

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

</form>
```

Formulare

```
public class HelloForm {

  @MvcBinding
  @FormParam("name")
  private String name;

  @MvcBinding
  @FormParam("age")
  private Integer age;

  /* getter + setter */

}
```

Formulare

```
@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 380
Bean Validation

Validierung

```
public class HelloForm {

  @MvcBinding
  @FormParam("name")
  @Size(min = 2, message = "Geben Sie Ihren Namen ein")
  private String name;

  @MvcBinding
  @FormParam("age")
  @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 HelloForm 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/send

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

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

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

</form>
```

CSRF Beispiel

http://www.hacked.com/

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

```
<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 geschrieben
- Prüfung des Token bei Submit

MVC 1.0 CSRF Modes

OFF	Kein Prüfung
EXPLICIT	Prüfung des Tokens mit @CsrfProtected
IMPLICIT	Prüfung bei jedem POST-Request

CSRF Beispiel

```
<form action="/send" 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="msg"/>

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

</form>
```

Prüfung mit @CsrfProtected

```
@Controller
@Path("/send")
public class MessageController {

  /* ... */

  @POST
  @CsrfProtected
  public String post( @FormParam("msg") String msg ) {

    /* ... */

  }

}
```

Empfehlung

IMPLICIT verwenden!

MVC kann mehr...

- Einfache Redirects aus Controller
- Internationalisierung / Lokalisierung
- CDI Scope: @RedirectScoped
- CDI Events
- ViewEngine SPI
- HTML/JS Encoding/Escaping

Zeitplan

- ☑ Q3 2014 Expert Group formed
- ☑ Q1 2015 Early Draft
- ☑ Q4 2015 Early Draft 2
- ☑ Q1 2018 Public Review
- ☑ Q4 2018 Proposed Final Draft
- □ Q2 2019 Final Release

JAKARTA EE

Feedback erwünscht

https://www.mvc-spec.org/

Danke!

Fragen?

https://www.mvc-spec.org/

Christian Kaltepoth / @chkal