



Java EE / EE4J MVC (JSR-371)

Daniel Dias

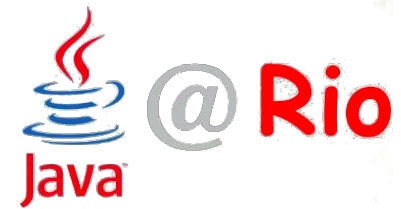
@danieldiasjava

daniel.dias@soujava.org.br

<http://danieldiasjava.wordpress.com/>

<http://linkedin.com/in/danieldiassantos>

Quem Sou Eu ?



Tecnólogo em Análise e Desenvolvimento de Sistemas

Pós-Graduado em Desenvolvimento Java
ambas pela Universidade Estácio Sá.

Contribuidor da JSR-371 (MVC 1.0)

Contribui em alguns projetos Open Source relacionados
a Java.



Links

meetup

<https://goo.gl/Xp81bR>

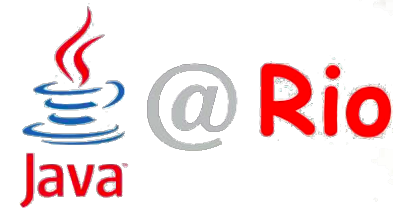


<https://goo.gl/MAB7CU>

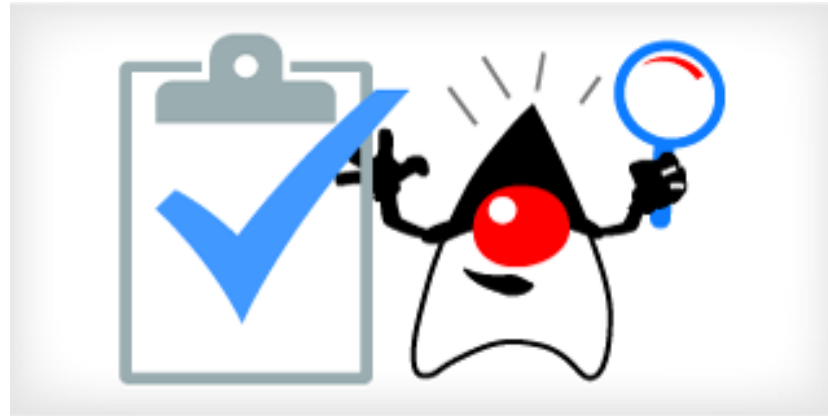


<https://goo.gl/fUeryb>

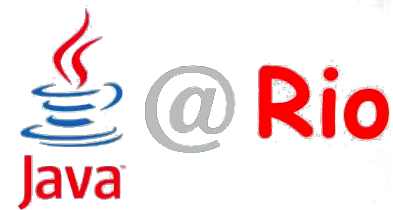
Agenda



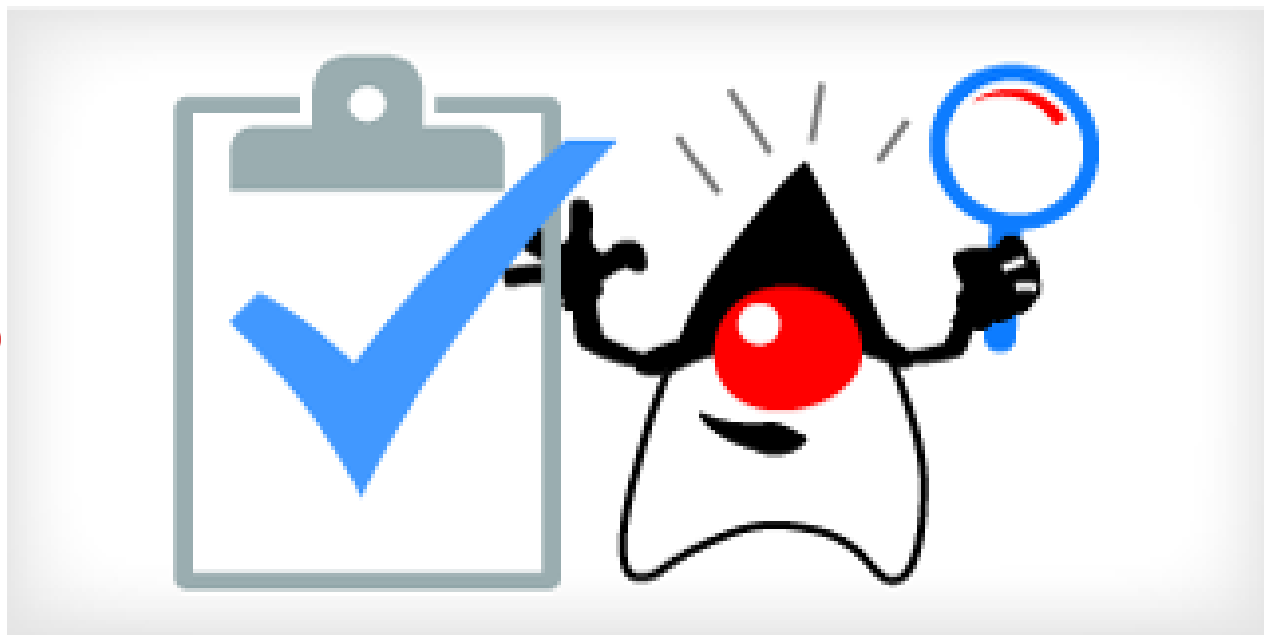
- 🦎 Contexto
- 🦎 MVC 1.0 JSR
- 🦎 Demo



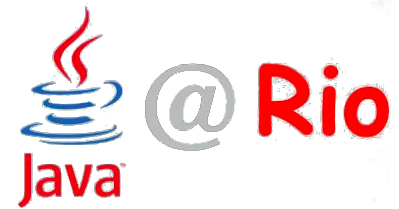
Agenda



Contexto



O que é MVC ?



Padrão de arquitetura de software que separa o **modelo**, a **interface do usuário** e a **lógica de controle** de uma aplicação em três componentes distintos.

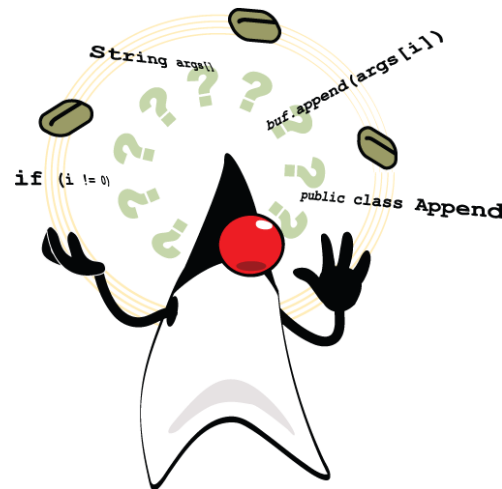
Cada um dos componentes tem a sua responsabilidade.



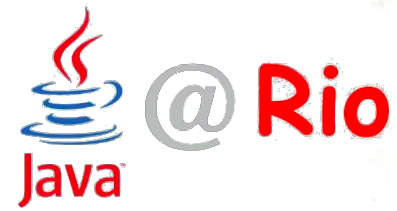
MODEL

É a representação específica da informação com a qual o sistema opera.

A lógica assegura a integridade dos dados que permite deriva-los.



VIEW

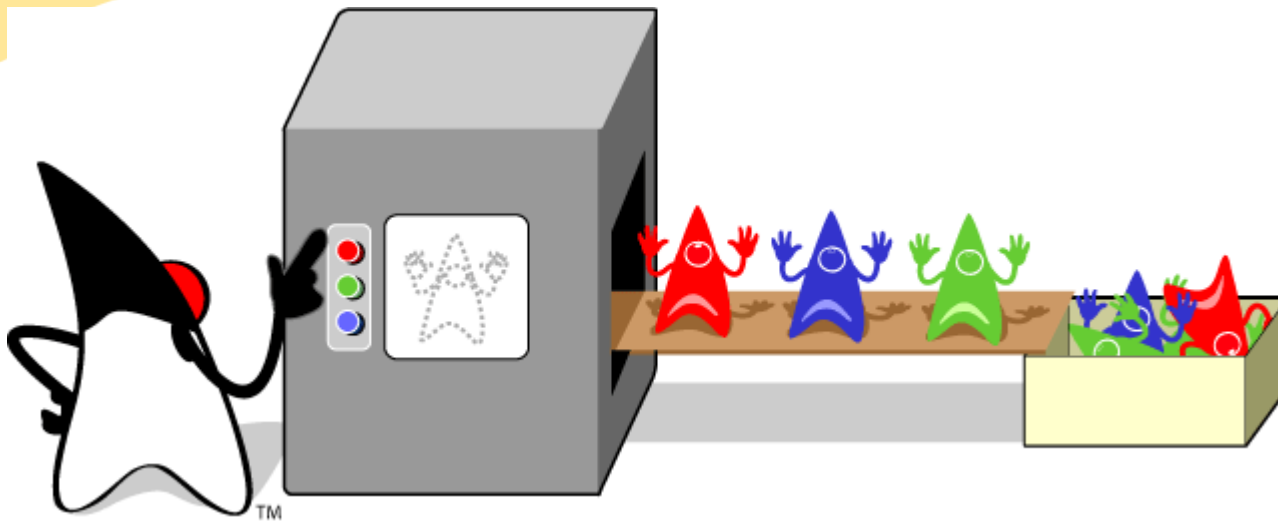


Representa o modelo em um formato adequado para interagir e acessar os dados, geralmente chamado de **“Interface do Usuário”** JSP, HTML, XML, JSON, etc).

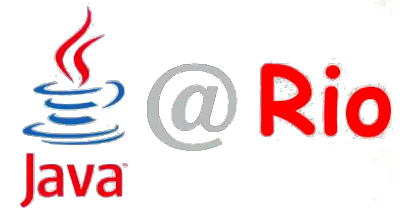
```
{
  "_id" : ObjectId("58d38f6420a01a1f27a83534"),
  "name" : "Daniel Dias dos Santos",
  "address" : {
    "state" : "RJ"
  },
  "age" : 25,
  "jugs" : [
    {
      "name" : "SouJava",
      "description" : "Description"
    }
  ]
}
```


CONTROLLER

É a ligação entre a **View** e o **Model**, é responsável por receber e responder a eventos, normalmente ações do usuário e invoca mudanças no modelo e provavelmente na visão.



Diferentes Estilos de MVC

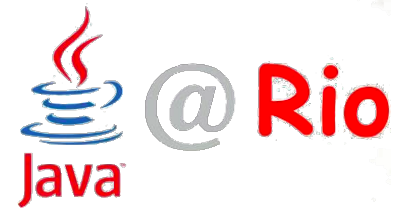


 MVC Baseado em Componentes (Pull/Puxar)

 MVC Baseado em Ação (Push/Empurrar)



MVC Baseado em Componentes



Um estilo específico do MVC que se tornou popular por frameworks de componentes como :

- JSF (e os Faces da Vida)
- Apache Wicket
- Tapestry
- Seam
- Apache Click



As solicitações **HTTP** são agrupadas e normalmente tratadas por **frameworks de componentes**, com pouca ou nenhuma interação com o código da aplicação, ou seja, o **controle** é fornecida pelo **framework** em vez da **aplicação**.

MVC Baseado em Ação



É um típico framework web que lidam com as requisições de entrada através de controladores e ações.

As solicitações **HTTP** são encaminhadas para controladores, onde elas são transformadas em **ações** pelo código da aplicação.

Frameworks de Terceiros :

- *Spring MVC (Líder)*
- *VRaptor*
- *Struts 2*
- *Struts 1*
- Nenhuma implementação padrão do Java EE.



Comparação



Componentes

Desenvolvimento fácil e com resultados rápidos.

Pouco controle e acesso a HTML, CSS, JS e HTTP.

Processamento automático de parâmetros da requisição.

Difícil de entender completamente(ex: JSF ciclo de vida)

Página centrada

Difícil de combinar com outras tecnologias

Validação e Conversão automática.

Ações

Não oculta o mecanismo de request/response do HTTP.

Desenvolvedor responsável por todo o HTML/JS/CSS

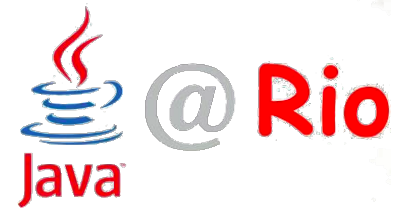
Processamento manual de parâmetros da requisição.

Requisição centrada

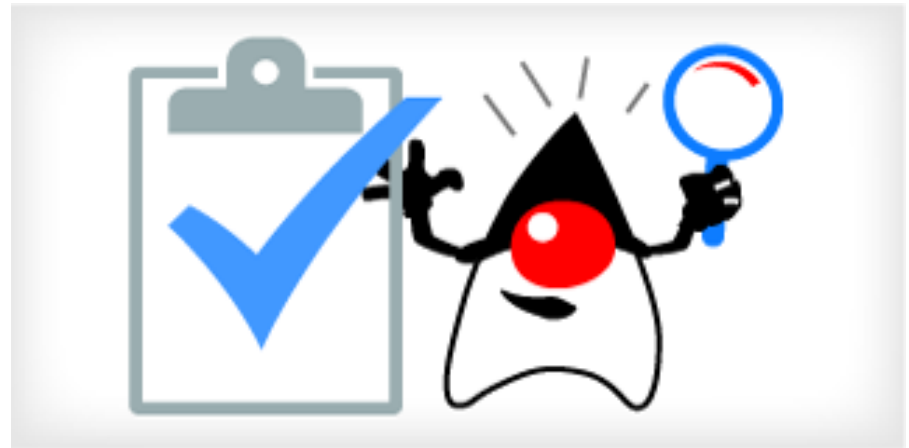
Validação e Conversão manual.



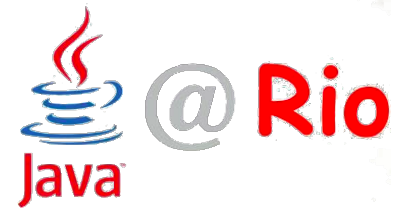
Agenda




MVC 1.0 (JSR)



MVC 1.0




<https://jcp.org/en/jsr/detail?id=371>

 Framework orientada a ação construída sobre as camadas da **JAX-RS**.

 Aproveitar as tecnologias Java EE existentes.

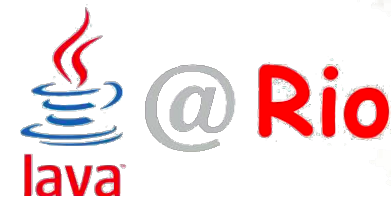
 Integração com **CDI** e **Bean Validation**

 Fornece suporte interno para **JSP** e **Facelets**.

 Não se destina a substituição do frameworks baseado em componentes como JSF, mas simplesmente oferecer uma alternativa para construção de aplicações web no Java EE.



TIME



Specification Leads

Ivar Grimstad

Grimstad, Ivar

Christian Kaltepoth

ingenit GmbH & Co. KG

Expert Group

Mathieu Ancelin

Caelum
: Rodrigo Turini

Frank Caputo

Ivar Grimstad

IBM
: Paul Nicolucci

ingenit GmbH & Co. KG
: Christian Kaltepoth

innoQ Deutschland GmbH
: Stefan Tilkov

Liferay, Inc.
: Neil Griffin

Mann, Kito D.

Oracle
: Santiago Pericasgeertsen

Oracle
: Manfred Riem

Red Hat
: Joshua Wilson

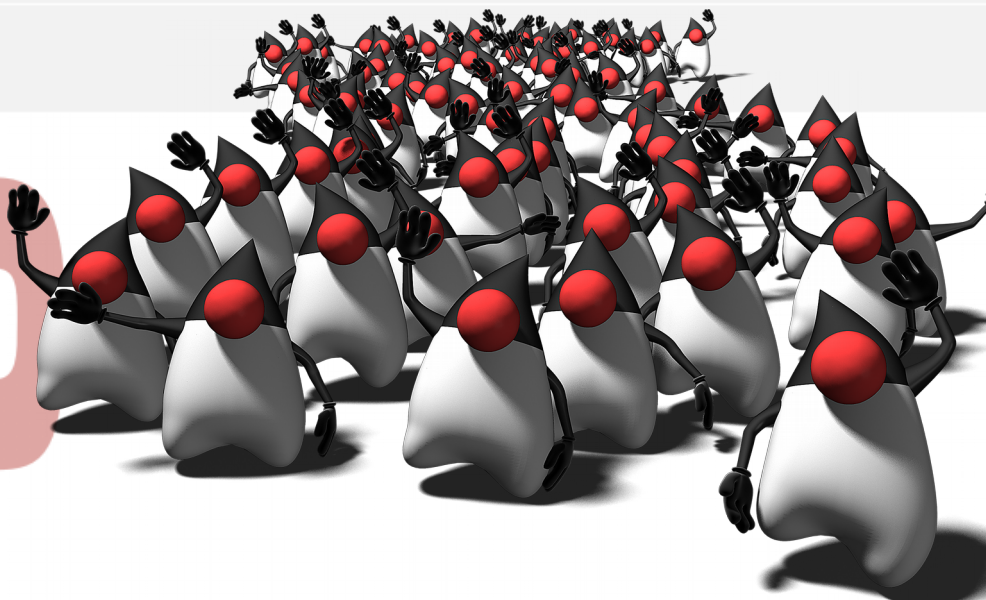
Guilherme de Azevedo Silveira

TmaxSoft, Inc.
: Woong-ki Lee

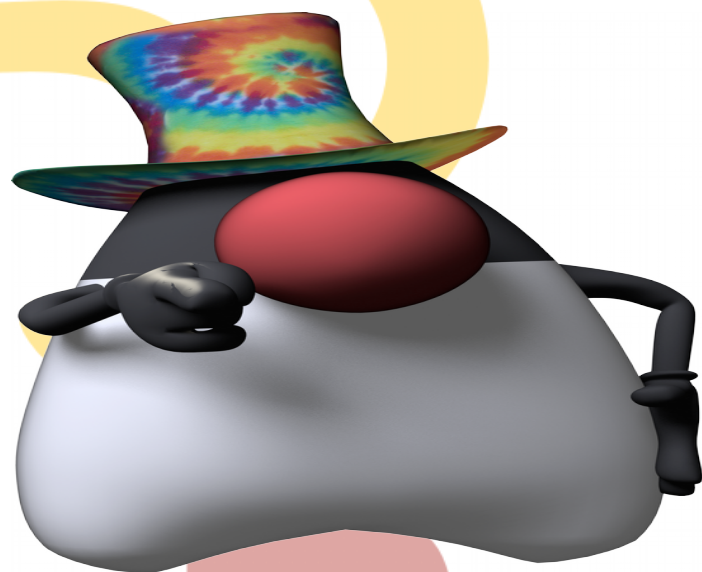
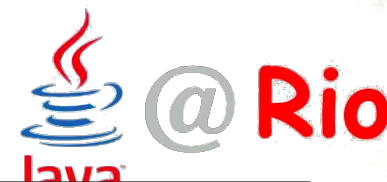
Usta, Rahman

Contributors

Daniel Dias dos Santos



Participe / Contribua



Pagina do Projeto

(<https://www.mvc-spec.org>)

GitHub

(<https://github.com/mvc-spec>)

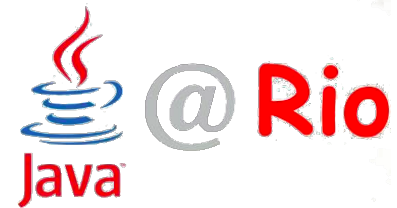
Implementação de Referencia (ozark)


Lista de Emails

(<https://groups.google.com/forum/#!forum/jsr371-users>)

(<https://groups.google.com/forum/?hl=en#!forum/ozark-users>)

MVC 1.0 / MODELS



 **javax.mvc.Models** : interface que define um mapa entre nomes e objetos :

```
@Inject
private Models models;

models.put("mensagem", "Ola Mundo!");
```

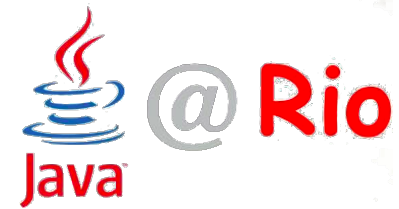
 **Opcional: Models CDI-Based(Recomendado)**

```
@Named("hello")
public class HelloWorld {
    private String mensagem;
    // Getter e Setter
}

@Inject
private HelloWorld helloWorld;

helloWorld.setMensagem("Ola Mundo!");
```

MVC 1.0 / VIEWS



As **Views** ficam dentro da pasta **WEB-INF/views**



Utiliza várias tecnologias de visualização diferentes.

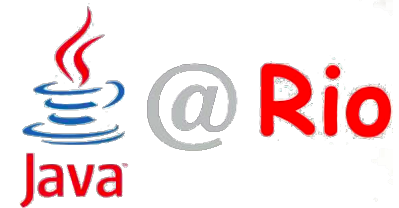
- **JSP** e **Facelets** obrigatórios


- Ozark também suporta :


FreeMarker, Velocity, Thymeleaf, Mustache, Handlebars e etc.


```
<%@ page contentType="text/html; charset=UTF8" language="java" %>
<!DOCTYPE html>
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <h1>CDI- Based ${hello.mensagem}</h1>
    <h1>Models ${mensagem}</h1>
  </body>
</html>
```

MVC 1.0 / CONTROLLER



 Um controller no MVC é um método de recurso do JAX-RS aplicada por **@Controller**.

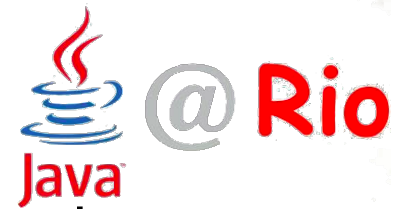
 O **@Controller** também pode se aplicado em nível de classe, onde a mesma se aplica a todos os métodos da classe.

 Possível criar classes híbridas (**@Controller** no nível do método)


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

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

MVC 1.0 / CONTROLLER




O controller pode retornar a View em 4 tipos de métodos :

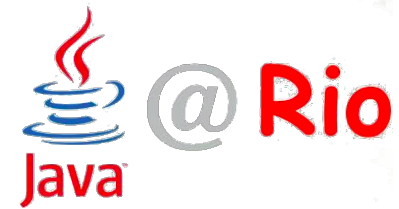
 **String**: Retorna o caminho da View.

 **void**: Requer a anotação `@javax.mvc.View`, utilizada para retornar uma View padrão quando é retornado **null**.

 **Response**: Tipico `javax.ws.rs.core.Response` fornecendo acesso ao response.

 **Viewable**: Encapsula informações sobre a **view**. Opcionalmente pode incluir referências a **Modelos** e Objetos **ViewEngine**. (**Vai sair da SPEC**)

MVC 1.0 / CONTROLLER



Tipos de Retorno

```
@Path("hello")
public class HelloController {
    @Inject
    private Models models;

    @GET
    @Controller
    public String hello() {
        models.put("mensagem", "Ola Mundo!");
        return "hello.jsp";
    }
}
```

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

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

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

    @GET
    public Response hello() {
        return Response.status(OK).entity("hello.jsp").build();
        Ou
        return Response.status(Response.Status.OK)
            .entity("hello.jsp").build();
    }
}
```

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


    @GET
    public Viewable hello() {
        return new Viewable("hello.jsp");
    }
}
```


MVC 1.0 / PARÂMENTROS DA VIEW PARA CONTROLLER



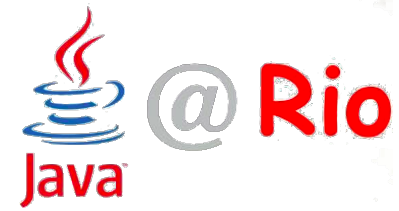
 **@PathParam:** Usado para extrair parâmetros de caminho.

 **@FormParam:** Usado para processar campos do formulário.

 **@QueryParam:** Usado quando queremos injetar na URL parâmetros de consulta – ex: **@QueryParam("nome") String nome, @QueryParam("idade") String idade**

 **@BeanParam:** Usado para injetar vários parâmetros de requisição em um bean.

MVC 1.0 / VALIDAÇÃO



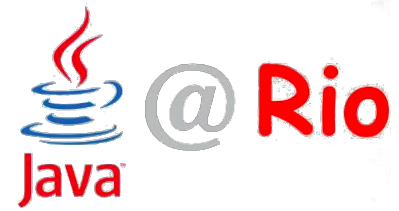
- Injeta **BindingResult** na classe circundante
- Anotar método com **@ValidateOnExecution**
- **BindingResult** dá acesso a informações detalhadas sobre quaisquer violações.

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

    @Inject
    private BindingResult bindingResult;

    @POST
    @Path("add")
    @ValidateOnExecution(type = ExecutableType.NONE)
    public String String add(@Valid @BeanParam Pessoa pessoa) {
        if(bindingResult.isFailed()) {
            String errors = bindingResult.getAllValidationErrors()
                .stream()
                .map(ValidationError::getMessage).collect( Collectors.joining("<br>" ));
            models.put("error", errors);
            return index();
        }
        this.pessoaDAO.salvar(pessoa);
        models.put("sucesso","Salvo com Sucesso");
        return "redirect:hello/lista";
    }
}
```

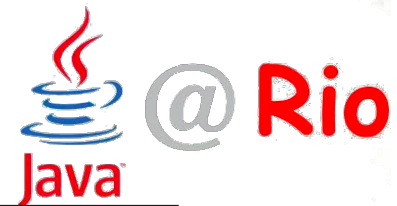

MVC 1.0 / BOOT



MVC Application == JAX-RS application + um ou mais @Controllers

```
@ApplicationPath("soujava")  
public class MVCApplication extends Application {  
}
```

MVC 1.0 / REQUISITOS



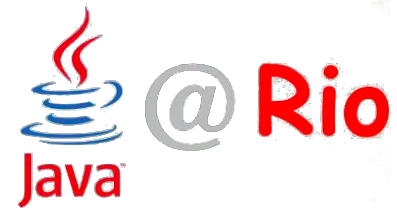
```
<repositories>
  <repository>
    <id>sonatype-oss-snapshots</id>
    <url>https://oss.sonatype.org/content/repositories/snapshots</url>
    <releases>
      <enabled>>false</enabled>
    </releases>
    <snapshots>
      <enabled>>true</enabled>
    </snapshots>
  </repository>
</repositories>
```

```
<dependency>
  <groupId>javax.mvc</groupId>
  <artifactId>javax.mvc-api</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
<dependency>
  <groupId>org.mvc-spec.ozark</groupId>
  <artifactId>ozark-core</artifactId>
  <version>1.0.0-m03-SNAPSHOT</version>
</dependency>
<dependency>
  <groupId>org.mvc-spec.ozark</groupId>
  <artifactId>ozark-resteasy</artifactId>
  <version>1.0.0-m03-SNAPSHOT</version>
</dependency>
```

```
<dependency>
  <groupId>javax.mvc</groupId>
  <artifactId>javax.mvc-api</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
<dependency>
  <groupId>org.mvc-spec.ozark</groupId>
  <artifactId>ozark-core</artifactId>
  <version>1.0.0-m03-SNAPSHOT</version>
</dependency>
<dependency>
  <groupId>org.mvc-spec.ozark</groupId>
  <artifactId>ozark-jersey</artifactId>
  <version>1.0.0-m03-SNAPSHOT</version>
</dependency>
```

{ RIO }

MVC 1.0 / REFERÊNCIAS



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

<https://jcp.org/en/jsr/detail?id=371>



TriadWorks

<http://blog.triadworks.com.br/mvc-1-0-jsr-para-um-framework-mvc-action-based-na-java-ee-8>



Caelum

<http://blog.caelum.com.br/primeiros-passos-do-mvc-1-0/>

Mais Exemplos : <https://github.com/mvc-spec/ozark/tree/master/test>

DEMO

GitHub : <https://goo.gl/BR8cQk>



@Rio



```
bart@: ERROR - Problem getting trends for service 5/054 | http://exp.co/goto
out waiting for a free available connection.
boneCP.getConnection(boneCP.java:583) ~[bonecp-0.7.1.RELEASE.jar:0.7.1.RELEASE]
ile.server.common.stats.storage.SqlConnectionPool.getConnection(SqlConnection
ile.server.common.stats.storage.select.invocation.BaseInvocationEventsFetcher
ile.server.app.servlet.dashboard.LazyDashboardServlet.handleLazyRequest(LazyD
ile.server.app.servlet.dashboard.LazyDashboardServlet.doWork(LazyDashboardSer
ile.server.app.servlet.AppServlet.work(AppServlet.java:45) [AppServlet.class:
ile.server.common.servlet.CommonServlet.handleRequest(CommonServlet.java:63)
ile.server.common.servlet.CommonServlet.doPost(CommonServlet.java:36) [Common
HttpServlet.service(HttpServlet.java:647) [servlet-api.jar:na]
HttpServlet.service(HttpServlet.java:728) [servlet-api.jar:na]
ile.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:38
ile.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210) [cata
ile.filters.ExpiresFilter.doFilter(ExpiresFilter.java:1179) [catalina.jar:7.0.4
ile.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:24
ile.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:210) [cata
ile.doDefaultCharsetFilter.doFilter(AddDefaultCharsetFilter.java:88) [j
ile.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:24
BUG - Found 0 flash hits, 0 filtered out. Filtering according to: 0
```

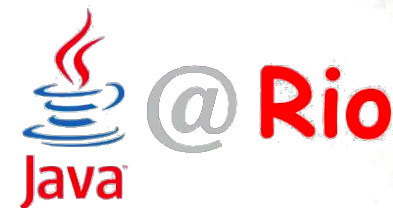
OverOps



WildFly SWARM



SOUJava{RIO}
sociedade de usuários java



#Obrigado

Daniel Dias
@danieldiasjava

daniel.dias@soujava.org.br

<http://danieldiasjava.wordpress.com/>

