



Keep Learning with Oracle University



Classroom Training
Learning Subscription
Live Virtual Class
Training On Demand



Cloud
Technology
Applications
Industries



education.oracle.com



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle Confidential –

Session Surveys

Help us help you!!

- Oracle would like to invite you to take a moment to give us your session feedback. Your feedback will help us to improve your conference.
- Please be sure to add your feedback for your attended sessions by using the Mobile Survey or in Schedule Builder.

Introduction to MVC 1.0

CON-4176

Santiago Pericas-Geertsens
JSR 371 Co-Spec Lead
Oracle



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Big thanks to the Expert Group!

- Mathieu Ancelin (Individual)
- Ivar Grimstad (Individual)
- Neil Griffin (Liferay, Inc)
- Joshua Wilson (RedHat)
- Rodrigo Turini (Caelum)
- Stefan Tilkov (innoQ Deutschland)
- Guilherme de Azevedo Silveira (Individual)
- Frank Caputo (Individual)
- Christian Kaltepoth (Individual)
- Woong-ki Lee (TmaxSoft, Inc.)
- Paul Nicolucci (IBM)
- Kito D. Mann (Individual)

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

- 1 The MVC Pattern
- 2 The M, the V and the C
- 3 View Engines
- 4 Validation
- 5 Scopes
- 6 Events

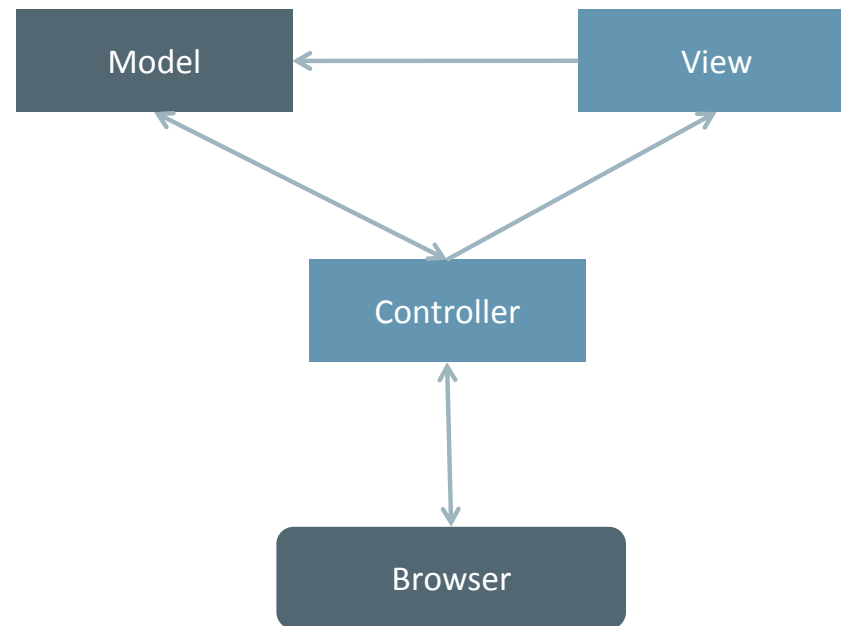
The MVC Pattern

In computing terms, it is “old as dirt”

The MVC Pattern

- Architectural pattern to build (Web) user interfaces
- Dates back to Smalltalk-76!
- Conceptually separates problem into 3 parts

The MVC Pattern



Isn't JSF also MVC?

- Yes, of course, but it is **component-based**
- The MVC 1.0 API is **action-based** instead

What is **action-based** then?

- Controllers are first-class
- Developers program *all* controllers *all* the time
- There are a lot less hidden tricks
 - So you can't blame the framework that easily!
- Better fit in a REST world

The M, the V and the C

Don't we have all these parts already?

Mappings to Java EE

- **Models** are your application objects
 - User instantiated or CDI beans
- **Views** are templates to create model representations
 - JSP, Facelets, Freemarker, Velocity, Handlebars, etc.
- **Controllers** are ...

What are controllers then?

- **Controllers** must be methods, actual running code
- **Controllers** must be accessible using URIs
- **Controllers** must provide access to HTTP request data
- **Controllers** must easily refer to models and views

Controllers are JAX-RS resource methods

- With slightly different semantics
- That typically return representations in `text/html`
- With improve validation support
- And a few other things ...

Hello Example: Greeting

The M

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

    private String message;

    public String getMessage() { return message; }

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

A CDI Model

Hello Example: hello.jsp

The V

```
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
    <h1>${greeting.message}</h1>
  </body>
</html>
```

A CDI Model

Hello Example: HelloController

The C

```
@Path("hello")
public class HelloController {
    @Inject
    private Greeting greeting;

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

A CDI Model

A JSP View

Anatomy of a Controller

- Resource methods with default `@Produces("text/html")`
- Convenient semantics for return type:
 - `Void`: View path specified using `@View`
 - `String`: View path or redirect
 - `javax.mvc.Viewable`: Combine views and processing instructions
 - `javax.ws.rs.core.Response`: Control over status codes and headers
 - Other Java Types: Value from `toString()` interpreted as view path

Anatomy of a Controller

Examples

```
@GET @Controller
@View("hello.jsp")
public void hello() {
    greeting.setMessage("Hello there!");
}
```

```
@GET @Controller
public Response hello () {
    return Response.ok()
        .header("mvc", "rocks")
        .entity("hello.jsp")
        .build();
}
```

Model Flavors

- To CDI or not to CDI
- May depend on *view engine* support
- API provides `Model`s class for name/value pairs
 - `Model`s is a CDI bean

Model Flavors

Using Models class

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

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

In request
scope

View Flavors

- Mandated by the spec: JSPs and Facelets
- All others supported as *extensions*
- Already contributed to RI (Ozark) by community:
 - AsciiDoc, Freemarker, Handlebars, Jade, JSR 223 (JavaScript)
 - Mustache, StringTemplate, Thymeleaf, Velocity

View Engines

An extension point or SPI

View Engine SPI

- Remember BYODKM?
- In MVC, you can BYOVE or “Bring Your Own View Engine”
- Single interface and CDI discovery
- Optionally a CDI producer for a engine-specific config object
 - Can be *overridden* by application developers

ViewEngine Interface

```
public interface ViewEngine {  
    boolean supports(String view);  
    void processView(ViewEngineContext context)  
        throws ViewEngineException;  
}
```

Can process
view?

Process view
using context

Freemarker Implementation

```
@ApplicationScoped
public class FreemarkerViewEngine extends ViewEngineBase {
    @Inject
    private freemarker.template.Configuration configuration;

    public boolean supports(String view) {
        return view.endsWith(".ftl");
    }

    public void processView(ViewEngineContext context)
        throws ViewEngineException {
        try {
            Template template = configuration.getTemplate(resolveView(context));
            template.process(context.getModels(),
                new OutputStreamWriter(context.getResponse().getOutputStream()));
        } catch (TemplateException | IOException e) { ... } } }
```

Uses producer

Checks view
path extension

View Path Resolution

- How is a view path like “hello.jsp” resolved?
- By default packaged under /WEB-INF/views/
 - Not accessible as static resources
- Default overridable using property:
`javax.mvc.engine.ViewEngine.viewFolder`
- Resolution done by each view engine

What about View Engine conflicts?

- Multiple view engines installed for same type of views
- Use `javax.annotation.Priority` to disambiguate:

```
@Priority(OUTRAGEOSLY_HIGH)
```

```
@ApplicationScoped
```

```
public class FreemarkerViewEngine extends ViewEngineBase {
```

```
    ...
```

```
}
```

Validation

Validate, validate and validate

Validation

- Provided by JSR 349 – Bean Validation 1.1
- Already integrated as part of JAX-RS 2.0
- REST validation and MVC validation *not* the same:
 - MVC requires finer control and recovery
 - Single exception handler too constrained

Validation Example

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

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

May throw
ConstraintValidation
Exception

Declares @Min,
@NotNull, etc.

Validation Example: Exception Handling JAX-RS

```
public class FormViolationMapper
    implements ExceptionMapper<ConstraintViolationException> {

    public Response toResponse(ConstraintViolationException e) {
        Set<ConstraintViolation<?>> s = e.getConstraintViolations();

        // process violations ...

        return Response.status(BAD_REQUEST)
            .entity("error.jsp").build();
    }
}
```

Maps Exception
to Response

So what is the problem then?

- Violations from *all* controllers flow to *same* handler
- Difficult to recover when context is lost
- Difficult to provide controller-specific logic
- One size does *not* fit all here!

MVC Validation

```
@Path("form") @Controller
public class FormController {
    @Inject
    private BindingResult br;

    @POST
    public Response formPost(@Valid @BeanParam FormDataBean f) {
        if (br.isFailed()) {
            // process violations ...
            return Response.status(BAD_REQUEST)
                .entity("error.jsp").build();
        }
        return Response.status(OK).entity("data.jsp").build();
    }
}
```

Injection indicates
MVC error handling

Summary of Validation

- JAX-RS exception mappers are available
 - May not satisfy your requirements
- Special validation mode using `BindingResult`
 - Controller code must be guarded for errors
- `BindingResult` also tracks binding errors
 - Problems in `ParamConverter`'s

Scopes

Standard CDI scopes plus ...

MVC Leverages CDI


- All Controllers are in **request** scope by default (JAX-RS)
- All **built-in** CDI scopes are available
 - Application, session, request and conversation
- MVC also defines **@RedirectScoped** as CDI extension
 - Supports POST-Redirect-GET pattern

Redirect Scope Example

```
@Named("bean")
@RedirectScoped
public class Bean implements Serializable {

    private String value = "Initial value";

    public String getValue() {
        return value;
    }
    public void setValue(String value) {
        this.value = value;
    }
}
```



Named bean in
redirect scope

Redirect Scope Example

```
@Path("redirect")
@Controller
public class RedirectController {

    @Inject Bean bean;

    @POST
    @Path("from")
    public String post() {
        bean.setValue("Redirect about to happen");
        return "redirect:/redirect/to";
    }
    ...
}
```

@RequestScoped

@RedirectScoped

Redirect Scope Example

```
@Path("redirect")
@Controller
public class RedirectController {

    @Inject Bean bean;

    @GET
    @Path("to")
    public String get() {
        return "redirect.jsp";
    }
    ...
}
```

Same instance
after redirect

Refers to bean
by name

Events

What's going on in my system?

Events in MVC

- MVC reuses the existing CDI event system
- Apps can listen for events using Observers
- Can be used for monitoring, debugging, tuning, etc.
- Spec defines an order for events

List of Events (EDR2)

- BeforeControllerEvent | AfterControllerEvent
- BeforeProcessViewEvent | AfterProcessViewEvent
- ControllerRedirectEvent
- *Maybe more to come ...*

Events Example

`@ApplicationScoped`

```
public class EventObserver {  
  
    void onBeforeController(@Observes BeforeControllerEvent e) {  
        println("URI: " + e.getUriInfo().getRequestURI());  
    }  
  
    void onAfterController(@Observes AfterControllerEvent e) {  
        println("Controller: " +  
            e.getResourceInfo().getResourceMethod());  
    }  
}
```

Event Observer

Event firing is
synchronous

Summary

Not to worry, we're almost done here

Summary

- MVC provides alternative to component-based JSF
- Tightly integrated with JAX-RS
 - Hybrid classes mixing controllers and resources
- Now on Early Draft Release 2 (EDR2)
- Reference implementation: <http://ozark.java.net>
- Subscribe to users@mvc-spec.java.net

Q/A

Time to start building MVC apps!

Santiago Pericas-Geertsen
JSR 371 Spec Lead
Oracle



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.