trainologic

JSF

trainologic

Faces Flows

- Let's say you want to implement a business flow that encapsulates several views.
- Before JSF Flows, your associated managed beans must have to be at the session scope.
- View scope is not enough as we talk about multiple views.

- What's wrong with that?

trainologic

Faces Flows

- By default, beans will stay bounded to the session until invalidation.
- Puts some pressure on the GC.

- A place for bugs as we may want to perform the business flow several times (beans already exist).

- Consider making a reservation process…

trainologic

Faces Flows

- JSF Flows allows you to group several views under a Flow.
- We can have flow-scoped managed beans which have less span than session-scoped ones.
- Supported by the EL.
- Each window/tab has its own scope (as opposed to a shared session).

- Can also use nested flows.

trainologic

Faces Flows

- A flow is based on strong convention.
- Each flow must reside in its own directory that matches the flow name.
- Inside there must be:
  - *yourflow_name***-flow.xml** – even if its empty.
  - *yourflow_name.***xhtml** – the single start page of the flow.
  - other views (accessible only from the flow).
- Outside the folder there should be:
  - *yourflow_name***-return.xhtml** – exit point of the flow,

trainologic

Configuration

- You can override the defaults and provides different names to the start view and return view(s) in the XML file.
- If you want to stay with the default, remember that the file must exist even if empty.

- As an alternative, you can also configure the flows in the *faces-config.xml* file.

trainologic

Flow Scope

- Defining beans in flow scope is different than regular managed beans.
- Instead of @ManagedBean, you must use: **@Named.**
- Taken from the CDI spec (Contextual Dependency Injection).
- And then you can use the @FlowScoped("flow_name").

- Accessing it from the EL is done with the *flowScope* implicit object.
- Let's see a basic example…

trainologic

Nested Flow

- A flow can invoke another flow.
- Done with the *<flow-call>* tag.

- A nested flow can only return to the calling flow (or one of its return pages.

- Allows for reuse of business logic flow.

trainologic

Nested Flow

- You can also pass parameters (properties) between flows.
- Defined in the XML as <outbound-parameter> (calling flow) and <inbound-parameter> (called flow) tags.

trainologic

Programmatic API

- Instead of defining flows in XMLs, you can do it in Java.

- Create a class that has a name like the flow.
- Create a method returning a *Flow* receiving a *FlowBuilder* parameter and has specific annotations.

- Let's see an example…

trainologic

Programmatic API

- Example for a nested flow:

```
public class MyFlow {
 @Produces
 @FlowDefinition
 public Flow defineFlow(@FlowBuilderParameter FlowBuilder flowBuilder) {
  String flowId = "myFlow";
  flowBuilder.id("", flowId);
  flowBuilder.viewNode(flowId, "/" + flowId + "/" + flowId + ".xhtml").markAsStartNode();

  flowBuilder.returnNode("myFlowReturn").fromOutcome("#{myFlowBean.returnValue}");
  flowBuilder.inboundParameter("param1FromFlow2", "#{flowScope.param1Value}");
  flowBuilder.flowCallNode("call2").flowReference("", ”outerFlow")
    .outboundParameter("param1FromMyFlow", "param1 MyFlow value");

  return flowBuilder.getFlow();
 }
}
```

trainologic

Stateless Views

- Starting with JSF 2.2, you can define stateless views.
- I.e., no state is stored.
- View scoped managed beans are lost.

- Set the *transient* attribute for the surrounding <f:view> element to **true** to enable stateless view.

- Test thoroughly as many components assume that their state will be persisted.

trainologic

Security Considerations

- From security POV, you have to **carefully** consider the following topics:

- Client-side state saving – although it has advantages regarding memory consumption on the server-side and disadvantages considering bandwidth, remember that the state is just non-encrypted serialized content.
- Client may tamper with the state. Consider what you put in the state!

trainologic

MyFaces

- Although the spec doesn't mention encrypted client-side state, some implementations support it.

- For myfaces, see: https://wiki.apache.org/myfaces/Secure_Your_Application

trainologic

Security Considerations

- Converters – these convert strings into objects. An attacker may tamper with the *String* to get undesired results. Always validate **after** conversion.
- I.e., always have a validator for converted objects.

- All input strings must be validated from both the business logic point (e.g., name must be up to 70 characters) and from security POV (Injection & XSS).

trainologic

Security Considerations

- Managed Beans – Sometimes it is convenient to access the global *FacesContext* object.
- You can get hold of things like request parameters.
- Note that by then, you have bypassed the entire validation stack.
- Treat every input obtained this way as tainted and validate it.

trainologic

AJAX Queue Control

- All Ajax requests shared the same queue before being sent to the server.
- Ajax has limited concurrency.
- We need to be careful no to overflow the queue.
- With JSF 2.2, you can specify the *delay* attribute which accepts a number of milliseconds to wait before queueing another request from the same source.

trainologic

Error Handling

- If we want to handle exceptions thrown by managed beans, we can create a class extending from *ExceptionHandlerFactory*.
- Return a custom exception handler class extending from *ExceptionHandlerWrapper*.
- Then we can control navigation (send to an error page) and add relevant attributes to the request scope or messages to the JSF context.
- Don’t forget to register your handler factory in the *faces-config.xml*.

trainologic

Bookmarks

- Consider the following scenario:
- On the first page you are displayed with several items.
- You select one of them and then directed to a view with detailed information about that item.
- That second view is rendered by a Faces-request Faces-response lifecycle.
- What if we want to bookmark the second view?
- I.e., support also Non-faces requests?

trainologic

View Params

- You can specify a <f:viewParam> tag inside a <f:metadata> to bind a request parameter to a managed bean property.
- Happens at the GET method of the HTTP.

- JSF 2.2 add the <f:viewAction action="…"/> tag.

trainologic

View Action

- Must be under <f:metadata>.
- The action is only triggered on a non-faces request (i.e., doesn't trigger on post-back).
- You can set it to execute also on post-back with the *onPostback* attribute.

- Support the *immediate* attribute.

trainologic

View Action

- By default is executed at the ‘Invoke Application’ phase, but that can be customized with the *phase* attribute.
- Like *action* it can control the navigation!
- Can be great for validation of view-params and their bindings.
- A good use-case is authorization!