Stateful Web Application Development with Spring Web Flow

John Case

Senior Consultant

Centare Group, Ltd.

http://www.centare.com

Centare Group, Ltd

- Centare Group Mission
  - Help clients achieve success through the delivery of appropriate technical solutions
- Specialists in:
  - Software application development
  - System integration
  - Information delivery (business intelligence)
- Strategic Partnerships
  - SpringSource, Actuate, Microsoft Gold Certified Partner

centare
group, ltd.

About Me

- Senior Consultant with Centare Group
- 7+ years professional Java development experience
- First used Spring 1.2 in fall of 2005
- First used WebFlow in summer of 2007
- Currently working with Spring 2.5

centare
group, ltd.

Agenda

- Stateful web application development
- Model business process as a single unit
- Processes are reusable across different areas of the application
- Testing

centare group, ltd.

Stateful Web Application

- Most web based applications are stateful
- Any application in which the user enters all data required for a single business transaction across multiple pages
  - Shopping Cart Checkout
  - Booking a flight online
- Partial data from early pages is either stored in HTTP Session or written to a database
  - Explicitly managed by the application

centare group, ltd.

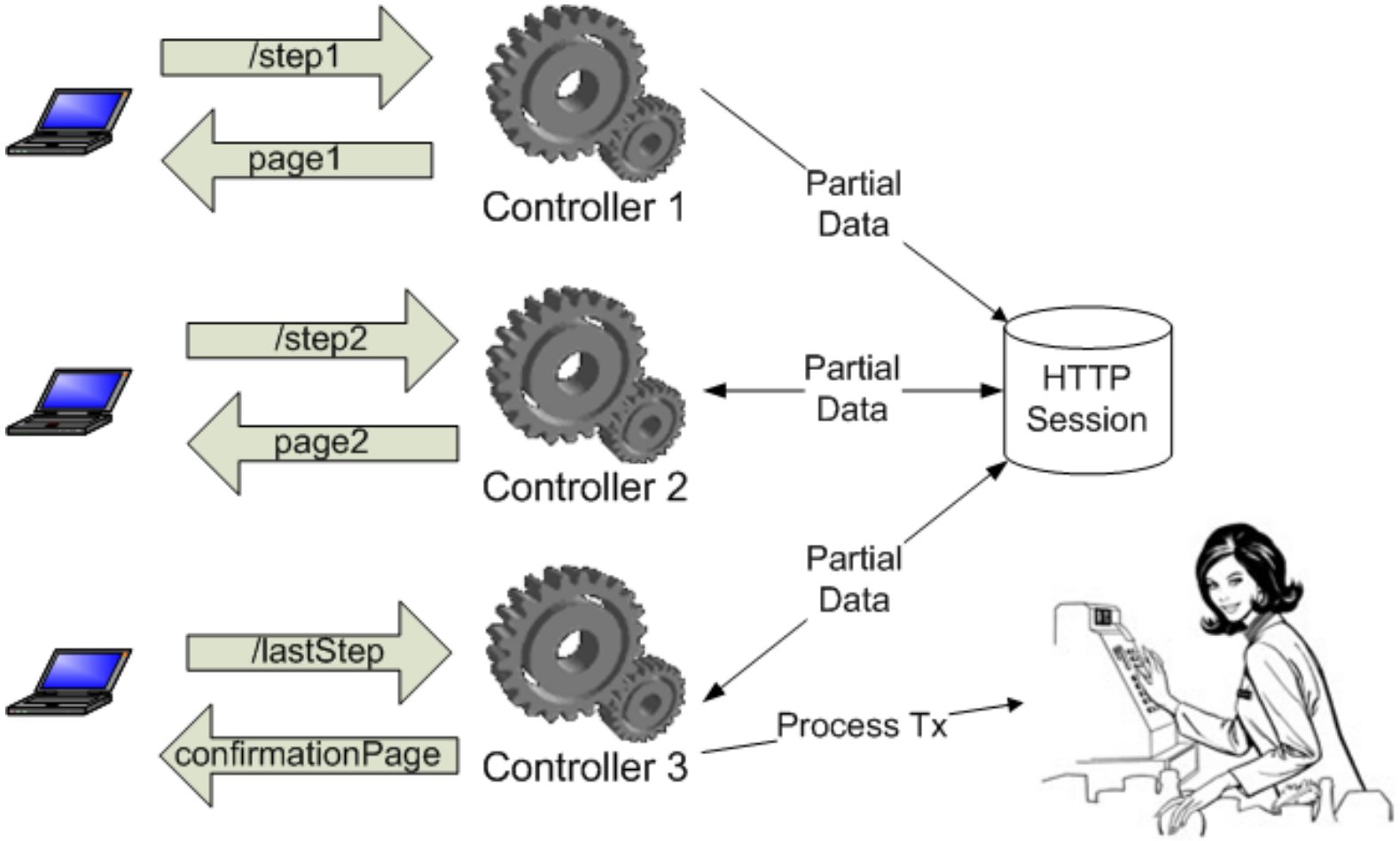
Stateful Web Application Challenges

- HTTP is an inherently stateless protocol
  - Each request is an event unto itself
- Traditional Web Frameworks rely on a request driven approach
  - Each page submits to a distinct controller
  - Each controller only responsible for a single step in the overall process

Stateful Web Application Challenges

- Many navigation paths can be difficult to manage
  - No central place to manage all navigation
- Request and Session scopes often not sufficient
  - Client side redirects clear out request scope
  - Double post and back button problems
- Difficult to test

Traditional Stateful Web Application



centare group, ltd.

Agenda

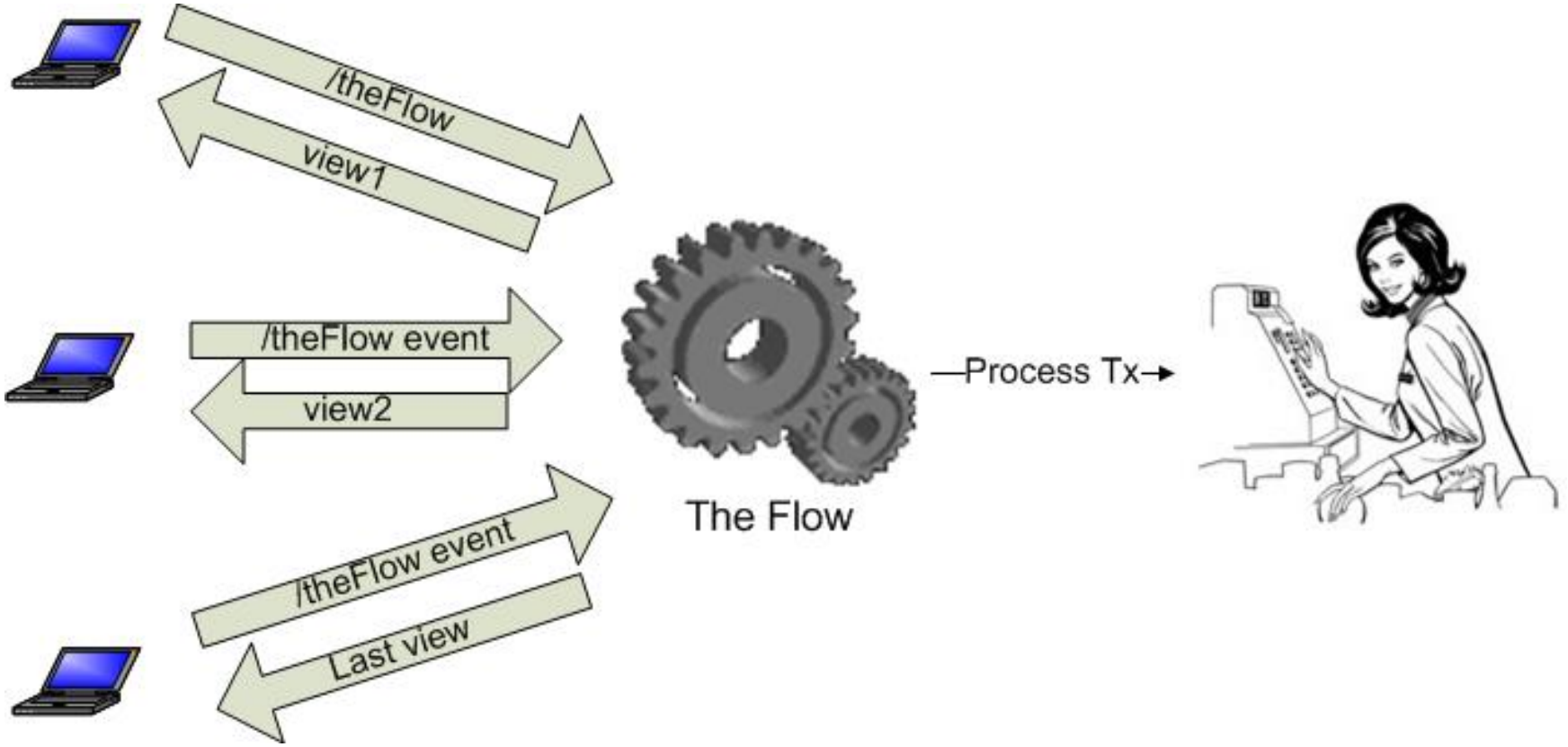
- Stateful web application development
- Model business process as a single unit
- Processes are reusable across different areas of the application
- Testing

centare group, ltd.

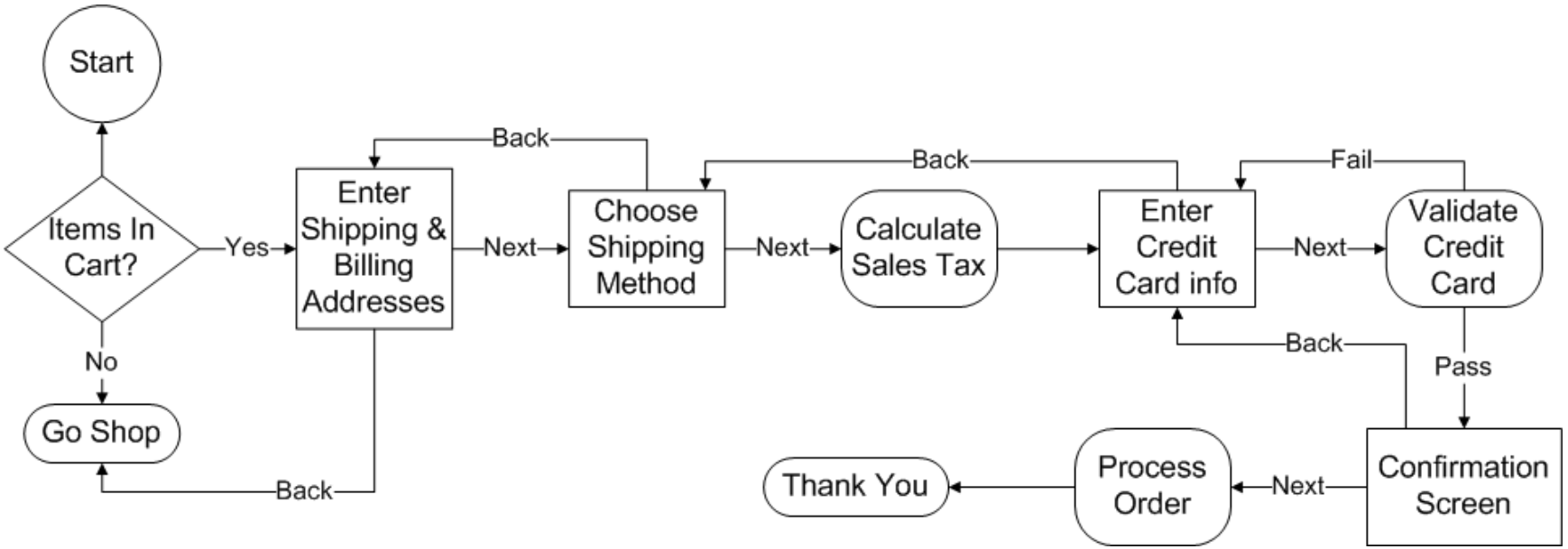
Process Driven Application

- Define all navigation in a central location
- Treat the entire interaction as one algorithm
- Let the framework decide what to do and where to go based on events triggered by the user

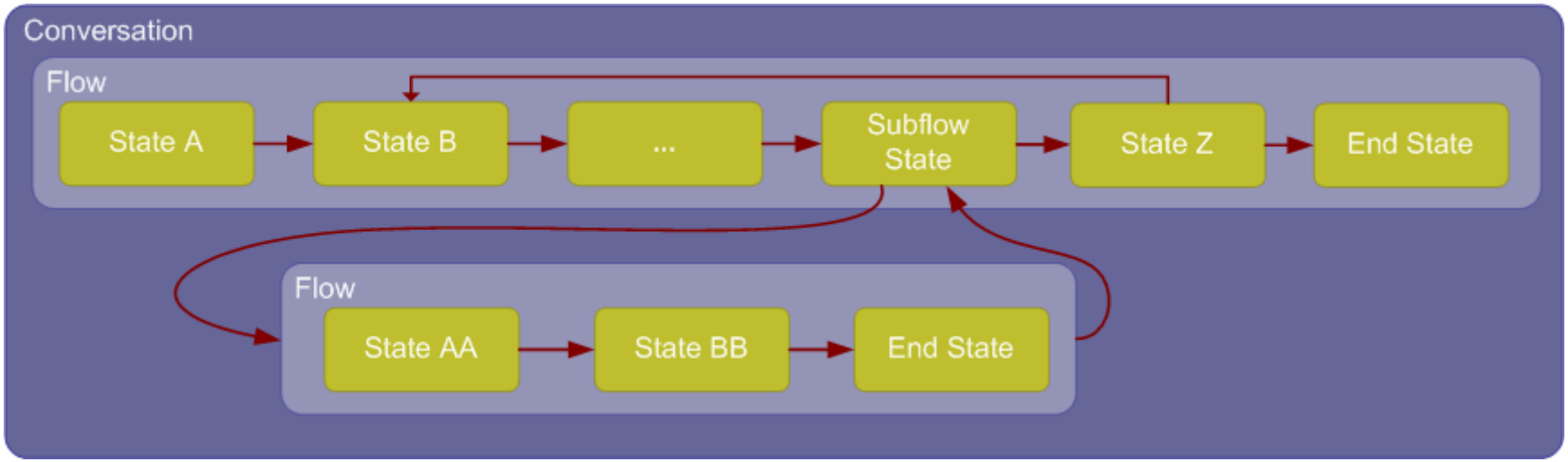
Process Driven Application



Shopping Cart Flow Diagram



High Level Concepts



centare group, ltd.

States

- View State
  - Pause execution and render a view to obtain user input
- Action State
  - Execute Java code
  - Can transition to different states depending on outcome
- Decision State
  - Branch logic

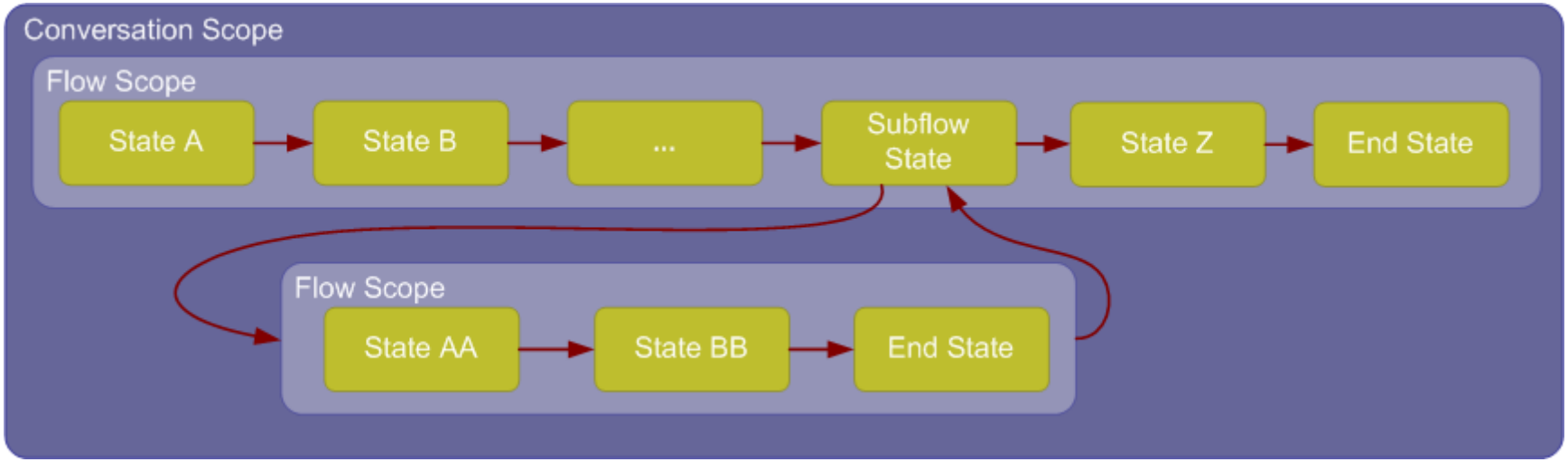
States

- Subflow State
  - Transfer execution to a different flow
  - Execution will return to this state when the subflow has completed
- End State
  - Final state of the flow
  - May or may not define a view to render

Variables

- WebFlow allows you to define variables as part of the process model
- Variables can be stored in one of several different scopes
  - Conversation, Flow, Flash, View, Request

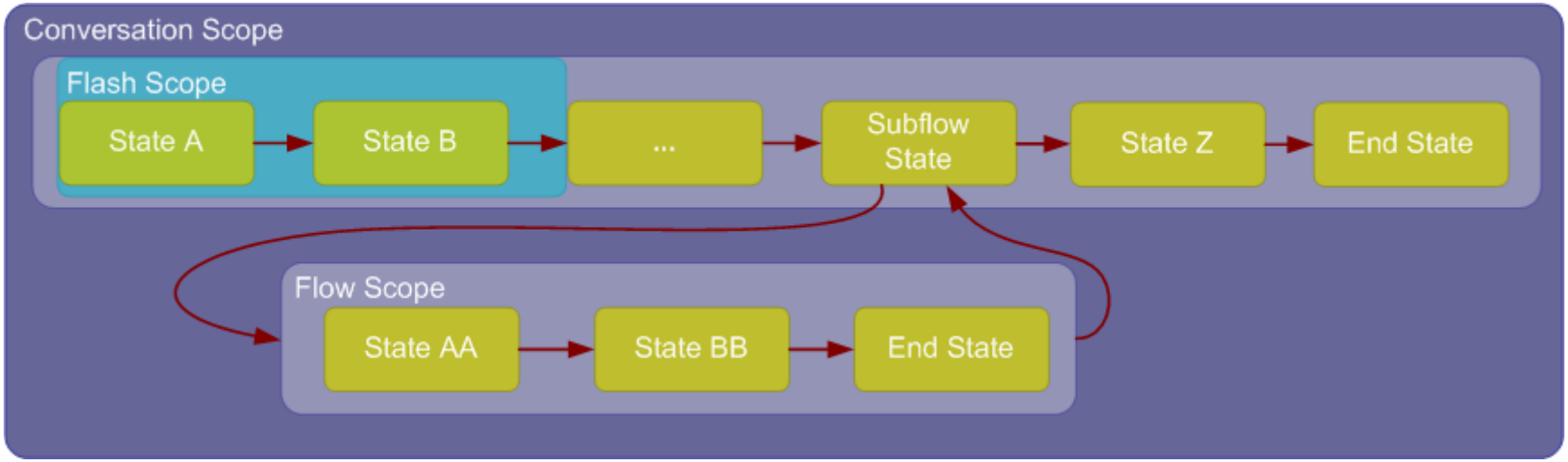
Conversation And Flow Scopes



- Conversation scope available to all flows
- Flow scope limited only to a single flow

centare group, ltd.

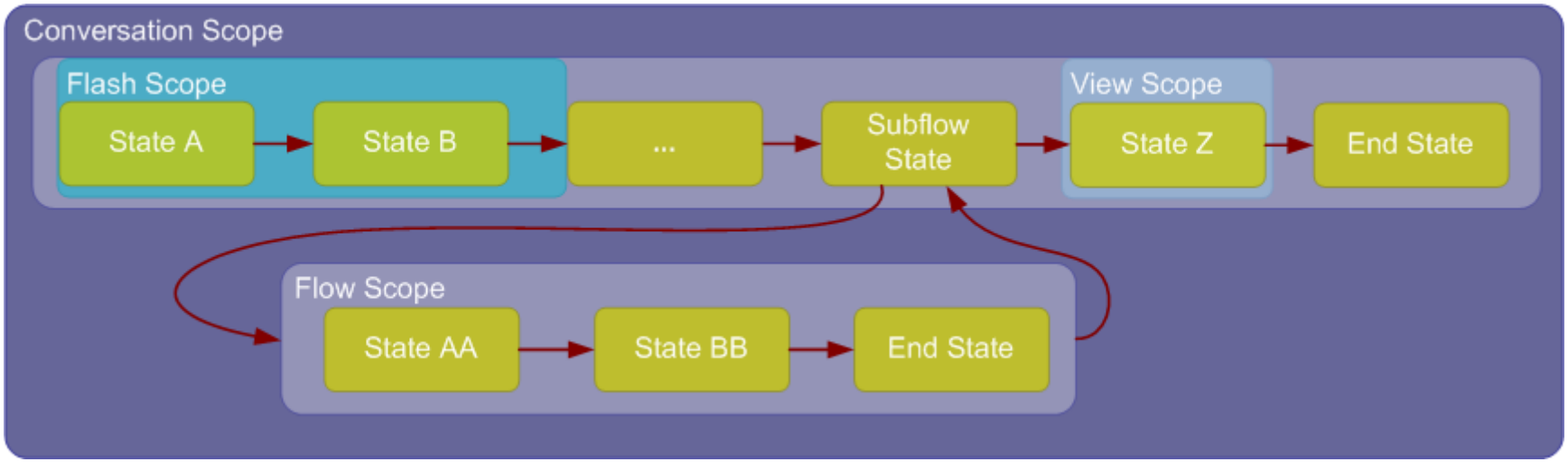
Flash Scope



- Cleared once a view is rendered
- Useful for temporarily saving results from action states
- Live through client-side redirects

centare group, ltd.

View Scope



- Created once a view-state is entered
- Destroyed when that view is exited
- Useful on pages that submit many AJAX requests

Request and Session Scopes

- Standard request scope still available
  - Same life span as always
  - Does not live across client-side redirects
- Session scope is not easily accessible
  - Is larger than conversation scope
  - Does not make sense from within WebFlow
  - Interact with session via input and output attributes of the flow

Modeling the Process

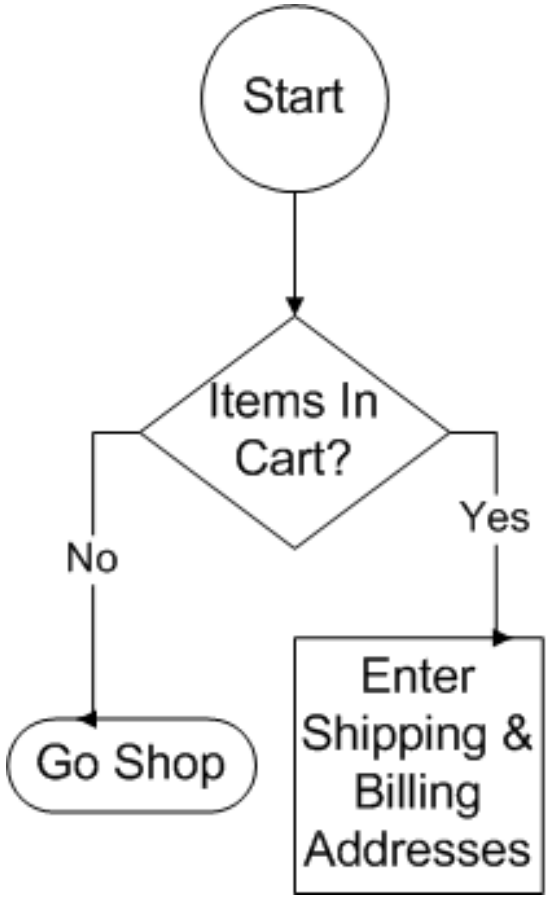
- Domain Specific Language (DSL) provided out of the box
  - XML
- It is possible to define a flow programmatically
- Implement FlowAssembler and FlowBuilder to create your own flow definition language

Packaging

- A flow is a self contained unit
- Flow definition and associated JSP pages packaged in the same directory
- Flows are registered with the FlowController by name

Shopping Cart Flow Definition

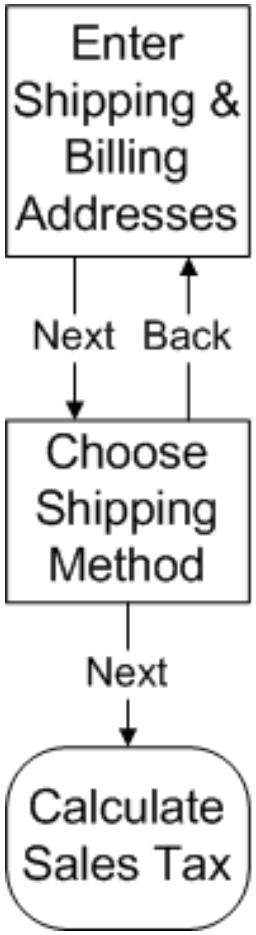
```
<flow xmlns="…">

   <input name="cart"
          required="true"
          type="com.mycomp…" />

   <decision-state id="hasItems">
          <if test="cart.empty"
                then="goShopping"
                else="enterAddresses"/>
    </decision-state>
```



centare group, ltd.

Shopping Cart Flow Definition

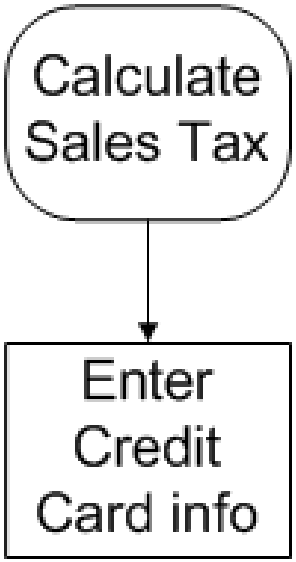
```
<view-state id="enterAddresses“ >
    <transition
        on="next“
        to="shipMethod" />
</view-state>

<view-state id="shipMethod">
    <transition
        on="back“
        to="enterAddresses" />
    <transition
        on="next“
        to="calculateTax" />
</view-state>
```



centare group, ltd.

Shopping Cart Flow Definition

```
<action-state id="calculateTax">
    <evaluate
            expression="basketService.calcTax(cart)" />
    <transition to="ccInfo" />
</action-state>

<view-state id="ccInfo">
    …
</view-state>
```



centare group, ltd.

Evaluate

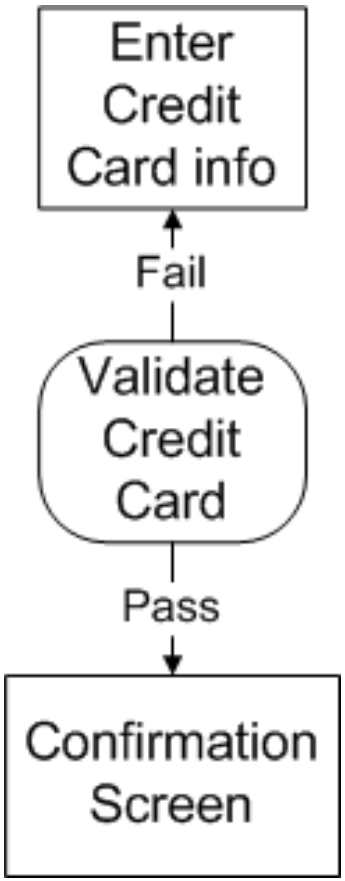
- Primary connection point between WebFlow and the back end services
- Expression in the evauate tag is Unified EL
  - Same expression language as JSP 2.1
  - Can directly reference Spring beans by name
  - Access to scoped variables
    - flashScope.foo
  - Expect to see more of this in Core Spring 3.0

Evaluate

- Use in action states
- Use in view states
  - on-entry
  - on-render
  - on-exit

Shopping Cart Flow Definition

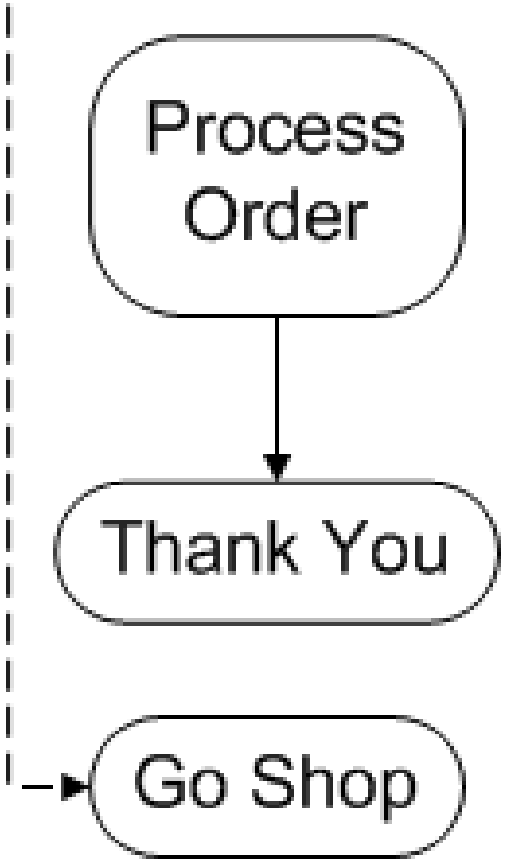
```
<action-state id="validateCC">
    <evaluate
        expression="basketService.validateCC(cart)" />
    <transition on="yes" to="confirm" />
    <transition on="no" to="ccInfo" />
</action-state>
```



centare group, ltd.

Shopping Cart Flow Definition

```
<action-state id="processOrder">
    <evaluate expression="…"/>
    <transition to="thankYou" />
</action-state>

<end-state id="thankYou" view="thankYou">
    <output name="confirmationNumber"
            value="cart.confirmationNumber" />
</end-state>

<end-state id="goShopping"
    view="externalRedirect:servletRelative:/home" />
```



centare group, ltd.

Coding the View

- Standard HTML forms
- Can use Spring MVC form tags
- Variables from all scopes are available to EL expressions as request scoped variables
- Always POST to the flow controller
  - EL Variable: ${flowExecutionURL}

Triggering an Event: Named Button

```
    </select>
</div>

<input type="submit"
       name="_eventId_back"
       value="Back" />

<input type="submit"
       name="_eventId_next"
       value="Next" />
```

Triggering an Event: Link

```
Nunc lorem augue, tincidunt
at, venenatis et, faucibus
vestibulum, tortor.
<div>

<a href="${flowExecutionUrl}
          &eventId=next">
    Next
</a>
```

corper turpis. Sed pede ac
Nunc lorem augue, tincidunt
at, venenatis et, faucibus
vestibulum, tortor.

Next

centare
group, ltd.

Agenda

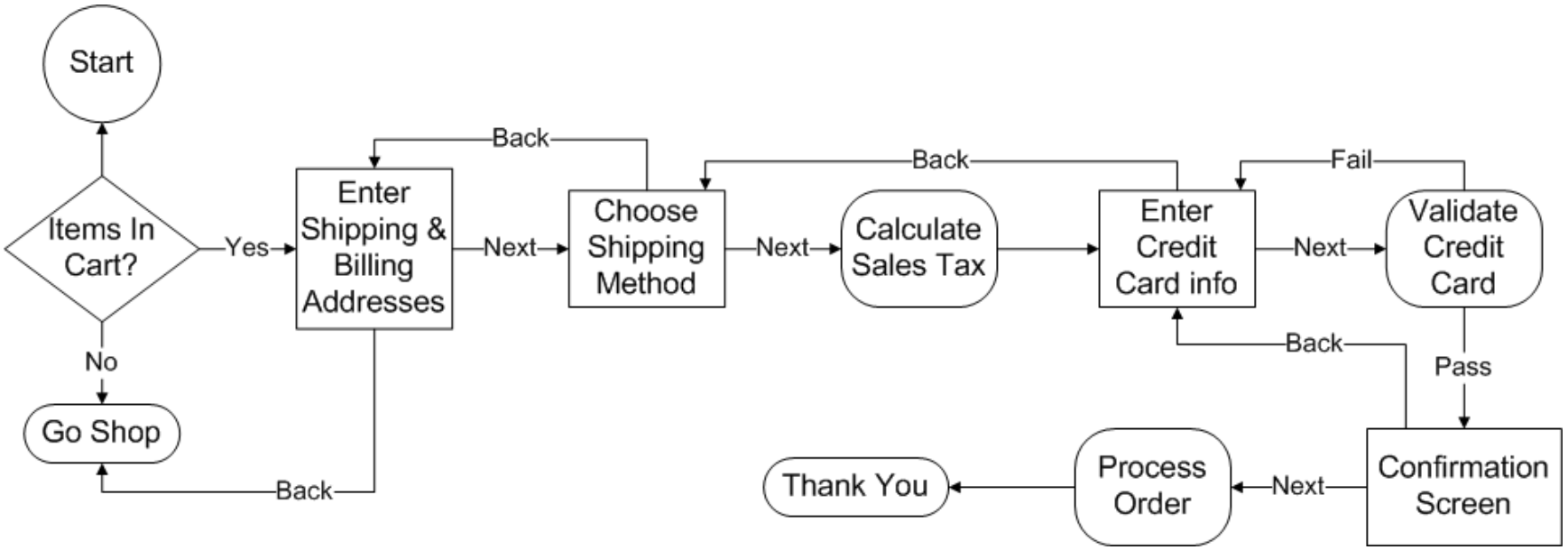
- Stateful web application development
- Model business process as a single unit
- Processes are reusable across different areas of the application
- Testing

centare group, ltd.

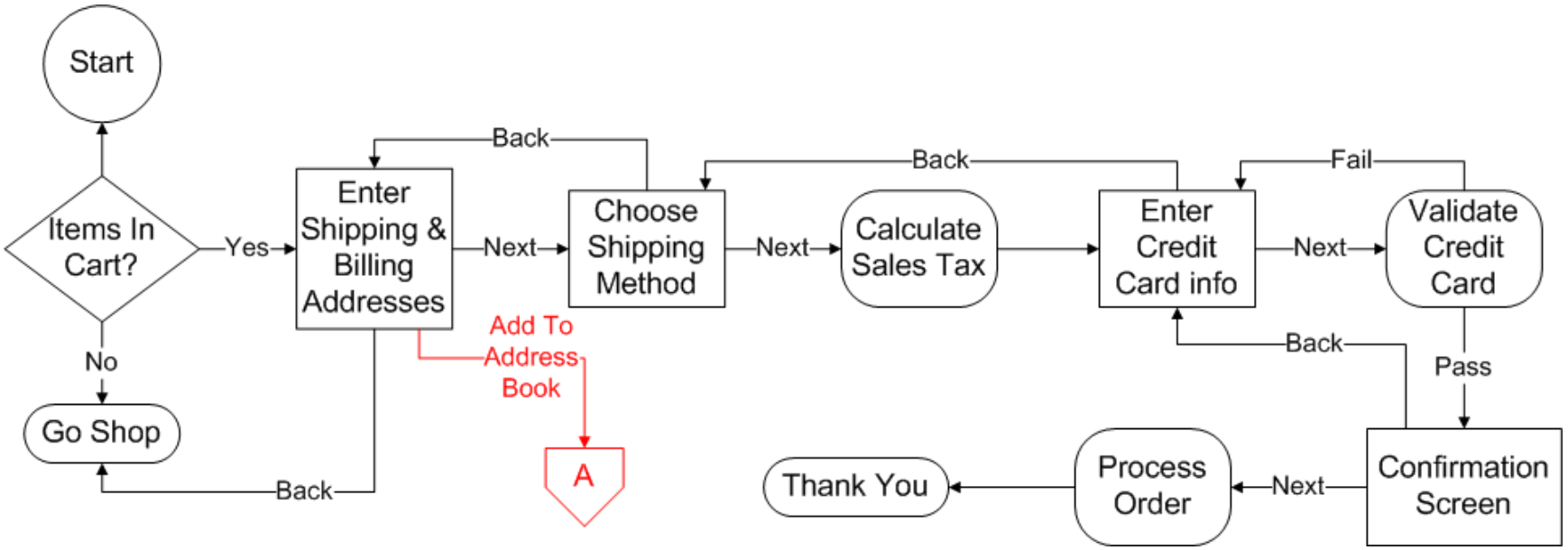
Subflows

- No different than a flow
- Call a flow from another flow
- Has its own variable namespace
  - Flow scope not shared between flows
  - Conversation scope is shared between flows
- Analogous to a method call in Java code

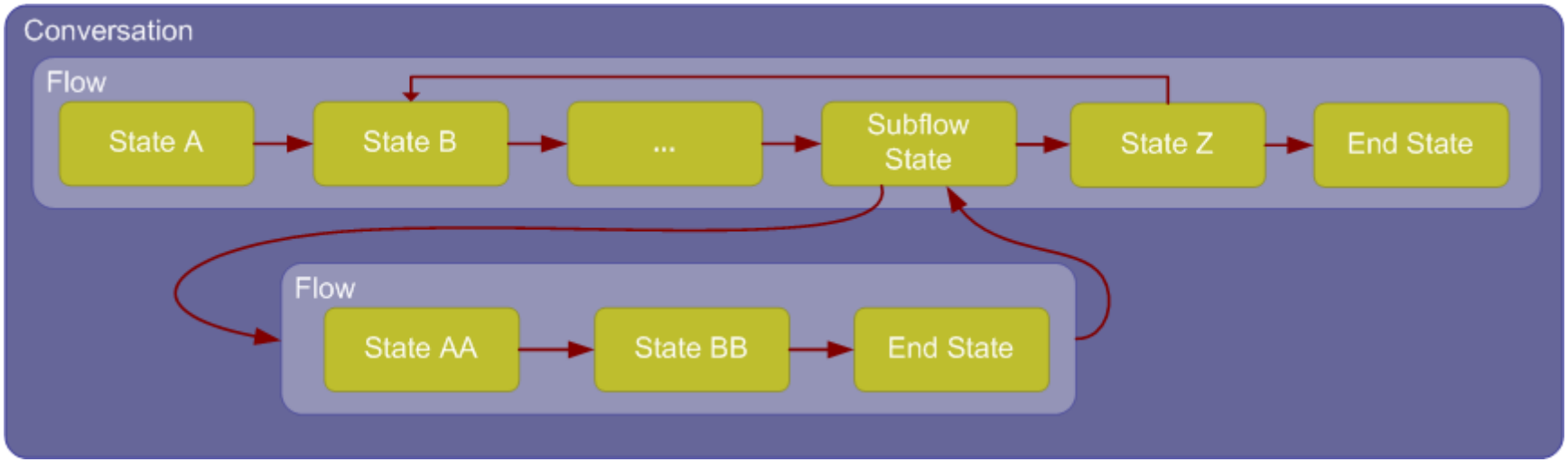
Shopping Cart Flow Diagram



Shopping Cart Flow Diagram



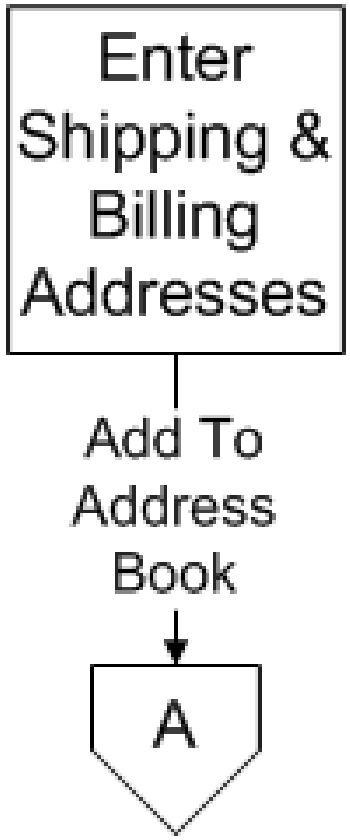
High Level Concepts



centare group, ltd.

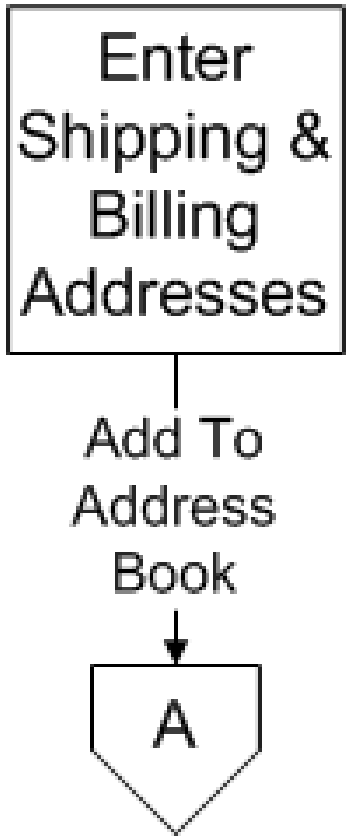
Subflow Definition

```
<view-state id="enterAddresses">
    <transition on="next" to="shipMethod" />
    <transition on="addrBook" to="addrBook" />
</view-state>

<subflow-state id="addrBook“
            subflow="addressBookFlow">
    <attribute name="user" value="cart.user" />
    <transition to=“enterAddresses” />
</subflow-state>
```



Securing WebFlow

```
<view-state id="enterAddresses">
    <transition on="next" to="shipMethod" />
    <transition on="addrBook" to="addrBook" />
</view-state>

<subflow-state id="addrBook“
            subflow="addressBookFlow">
    <secured attributes=“ROLE_AUTHENTICATED” />
    <attribute name="user" value="cart.user" />
    <transition to=“enterAddresses” />
</subflow-state>
```



Securing WebFlow

- Integrates with Spring Security
- <secured> tag can be applied to
  - States
  - Transitions
  - Flows
- AccessDeniedException thrown
  - Handled by Spring Security according to its own configuration

Agenda

- Stateful web application development
- Model business process as a single unit
- Processes are reusable across different areas of the application
- Testing

centare group, ltd.

Testing WebFlow

- Testing a multi page web process can be very difficult
  - Usually involves setting up a test server and actually running the application
  - Difficult to isolate the process flow definition from the execution
  - Difficult to test middle steps without executing all previous steps

Testing WebFlow

- WebFlow ships with the AbstractXmlFlowExecutionTests class
  - JUnit test
  - All tests run inside of a full Spring container, with all objects wired up
- At minimum, give it the location of your XML flow definition
- Provides other useful hooks
  - Register stubs and mocks with the Spring container
  - Provide path to parent flow definition if necessary

AbstractXmlFlowExecutionTests

- Drive Flow Execution
  - startFlow(), setCurrentState(), resumeFlow()
- Assertions
  - assertCurrentStateEquals()
  - assertFlowExecutionEnded()
  - more
- Scoped Variable Access
  - Get and set

centare group, ltd.

MockExternalContext

- Use this class to populate whatever external state you expect for the portion of the flow under test
  - setCurrentUser()
  - setEventId()
  - getRequestParameterMap().put(“name”, “value”)

Example Test: Making Transitions

```
public void testSomething() throws Exception {
  setCurrentState("enterAddresses");

  MockExternalContext context =
          new MockExternalContext();
  context.setEventId("next");

  resumeFlow(context);

  assertCurrentStateEquals("shipMethod");
}
```

centare group, ltd.

Summary

- Stateful web application development
  - Process driven approach very different from request driven approach of traditional MVC frameworks
- Model business process as a single unit
  - WebFlow uses a Domain Specific Language (DSL) analogous to a flow chart to model the process in a single file

Summary

- Processes are reusable across different areas of the application
  - Flows can be called by other flows just like a method call in Java code. Flows can accept input parameters and can return output values.
- Easily Testable
  - AbstractXmlFlowExecutionTests and MockExecutionContext provide a valuable harness for testing flow definitions in isolation from application logic.

centare group, ltd.

Thank You!

centare
group, ltd.