Spring Web Flow 1.0 A Next-Generation Web Application Controller Framework

Keith Donald

Interface21
www.interface21.com

TS-6821

Topics in This Session

Web Flow Quick Start

Key Benefits

What's Next

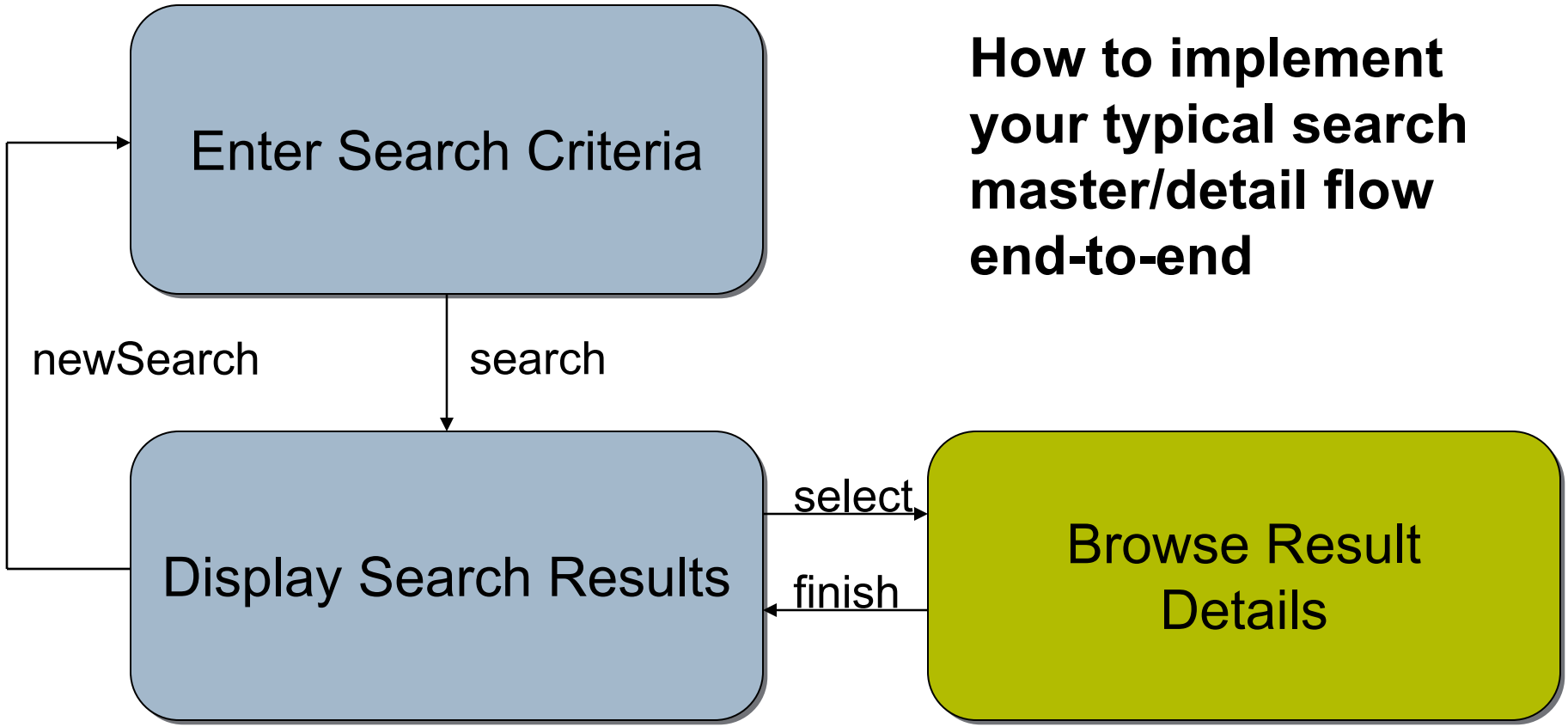
Topics in This Session

Web Flow Quick Start

Key Benefits

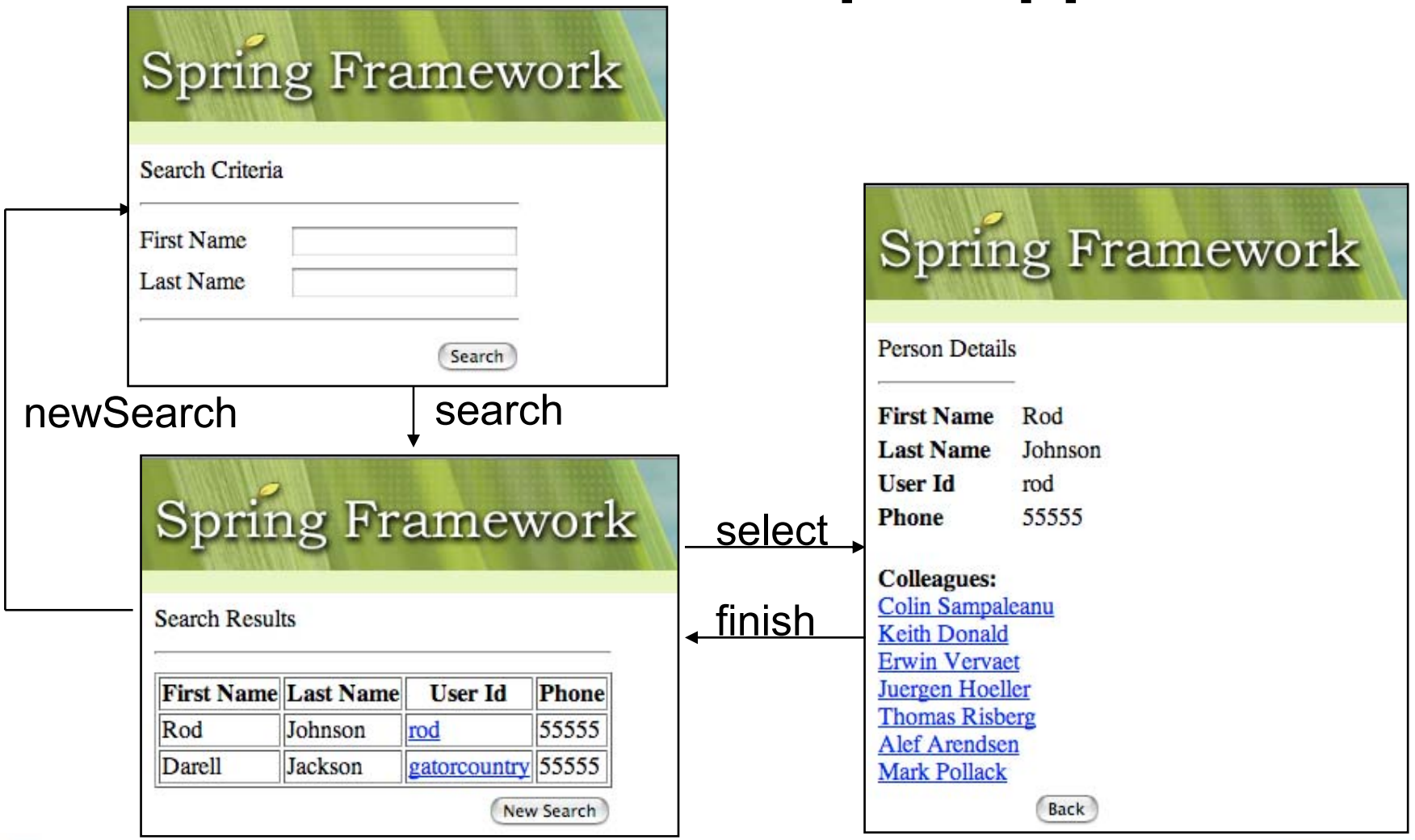
What's Next

Web Flow Quick Start



How to implement your typical search master/detail flow end-to-end

The Contacts Sample Application



Quick Start Assumptions

- The application logic to perform a contact search is already written
- The JavaServerFaces (JSF™) views are already written
- New web application project shell created

Quick Start Steps

- Bootstrap the Contacts Application
- Configure the Web Flow Engine
- Implement the Search Flow

Quick Start Steps

- **Bootstrap the Contacts Application**
- Configure the Web Flow Engine
- Implement the Search Flow

Bootstrap the Contacts Application

- Inside <webapp/> within web.xml

```
<context-param>
   <param-name>contextConfigLocation</param-name>
   <param-value>
      /WEB-INF/contacts-webapp-config.xml
   </param-value>
</context-param>

<listener>
   <listener-class>
      org.springframework.web.context.ContextLoaderListener
   </listener-class>
</listener>
```

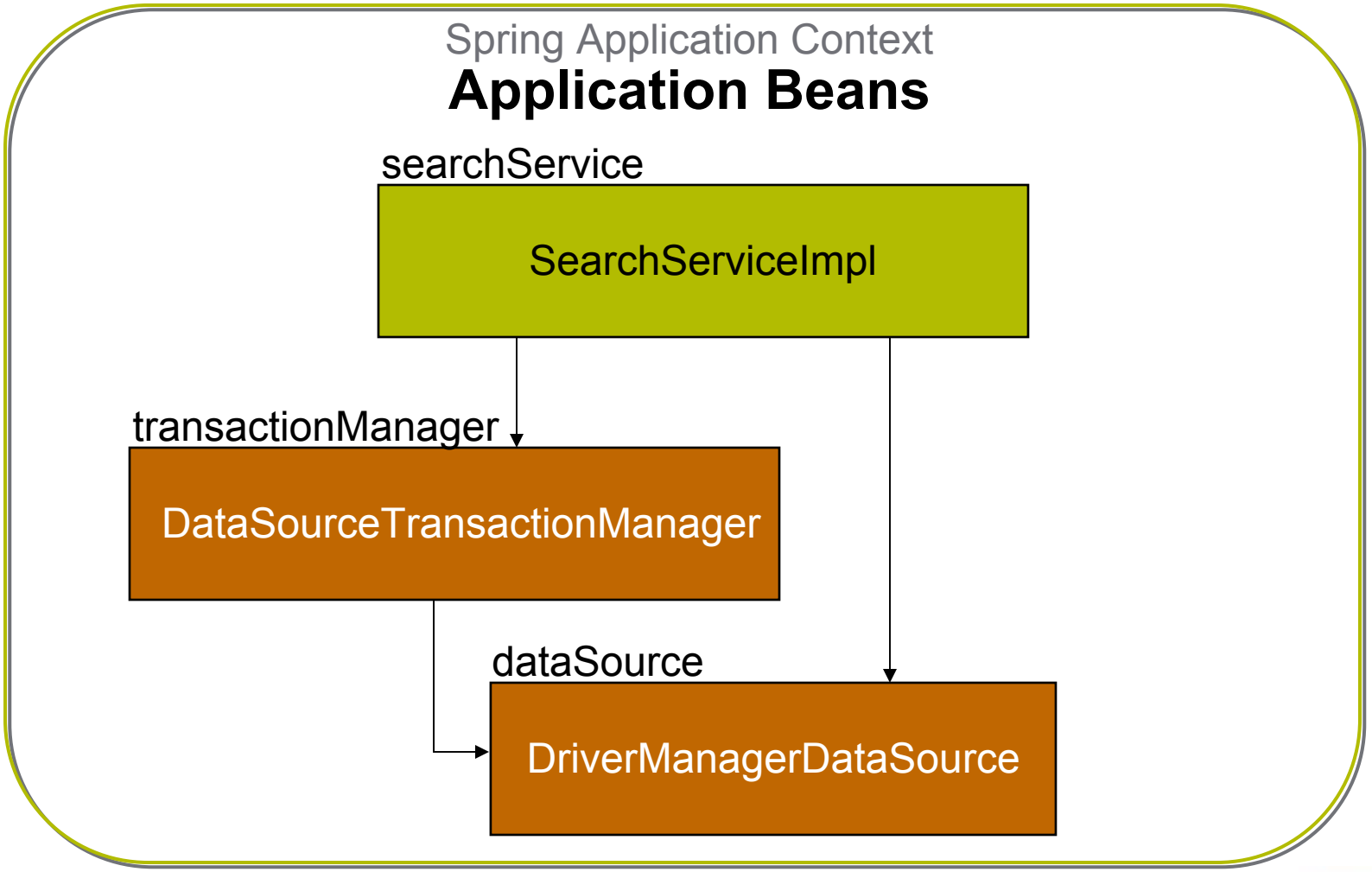
Where application configuration resides

Inside webapp-config.xml

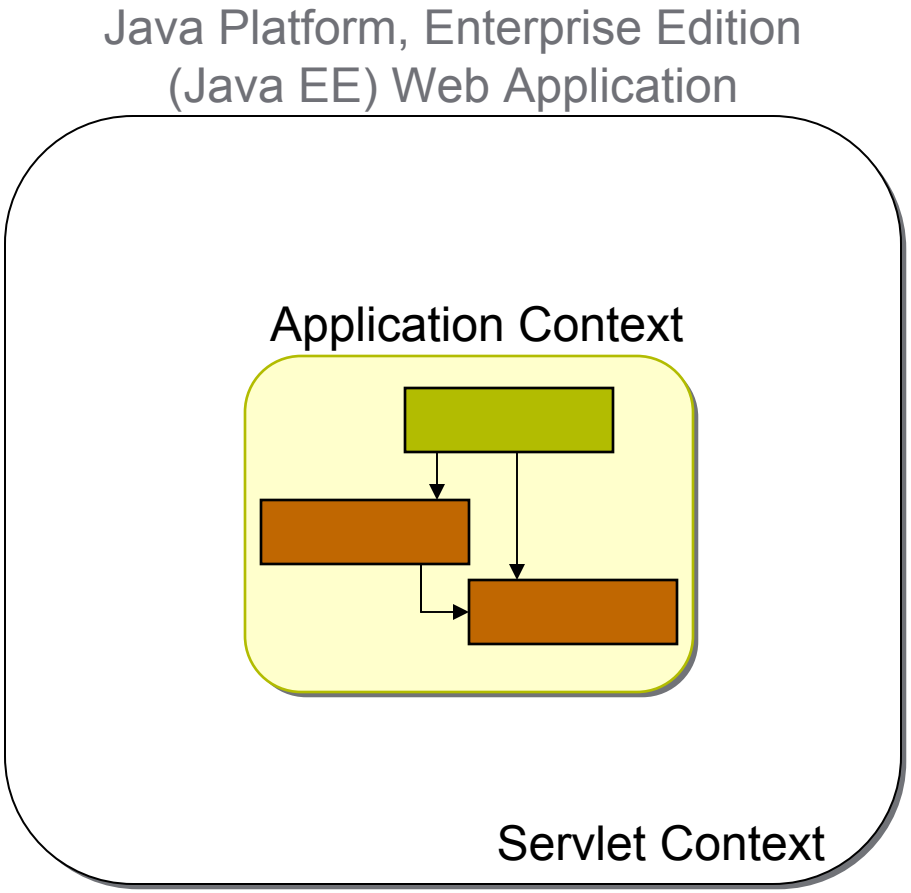
```
<bean id="searchService" class="phonebook.impl.JdbcSearchService">
    <constructor-arg ref="dataSource"/>
</bean>

<bean id="dataSource" class="phonebook.TestDataSourceFactory"/>

<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>

<tx:annotation-driven transactionManager="transactionManger/>
```

```
public class JdbcSearchService implements SearchService {
    …
    @Transactional
    public List<Contact> searchDirectory(SearchCriteria c) {
}
```

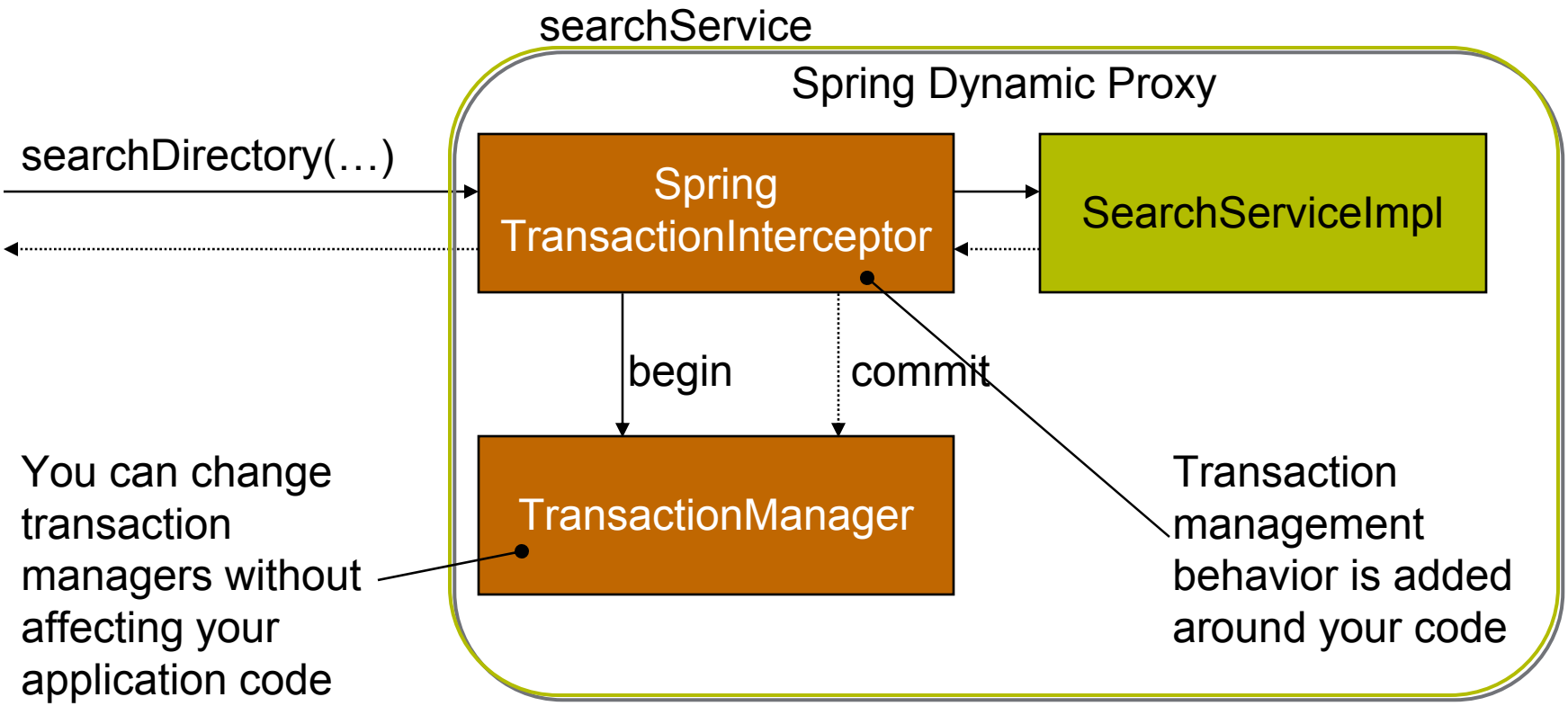
Web Application Context Initialization



After Web Application Initialization



If You Invoked a Search…



Quick Start Steps

- Bootstrap the Contacts Application
- **Configure the Web Flow Engine**
- Implement the Search Flow

Inside contacts-webflow-config.xml

```
<flow:executor id="flowExecutor" registry-ref="flowDefinitionRegistry"/>

<flow:registry id="flowDefinitionRegistry">
    <flow:location path="/WEB-INF/flows/*-flow.xml"/>
</flow:registry>
```

Executes flows in registry on behalf of clients

Contains flow definitions eligible for execution

Inside faces-config.xml

```
<faces-config>
   <application>
      <navigation-handler>
         org.springframework.webflow.executor.jsf.FlowNavigationHandler
      </navigation-handler>
      <variable-resolver>
         org.springframework.webflow.executor.jsf.DelegatingFlowVariableResolver
      </variable-resolver>
   </application>
   <lifecycle>
      <phase-listener>
         org.springframework.webflow.executor.jsf.FlowPhaseListener
      </phase-listener>
   </lifecycle>
</faces-config>
```

Drives flow navigation

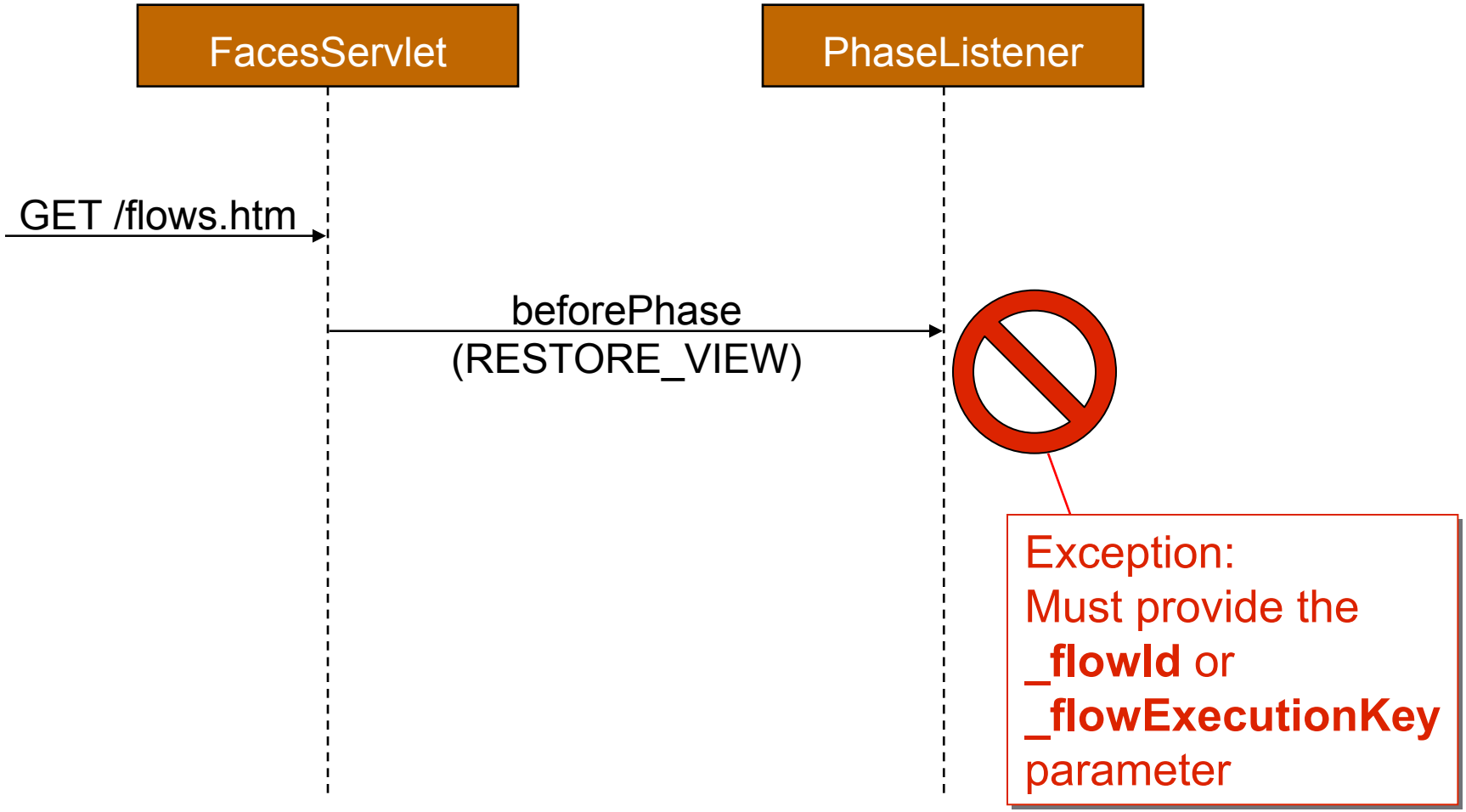
Resolves flow variables

Manages flow lifecycle

Check Point

- Bootstrap the Contacts Application ➨
- Configure the Web Flow Engine ➨
- Implement the Search Flow

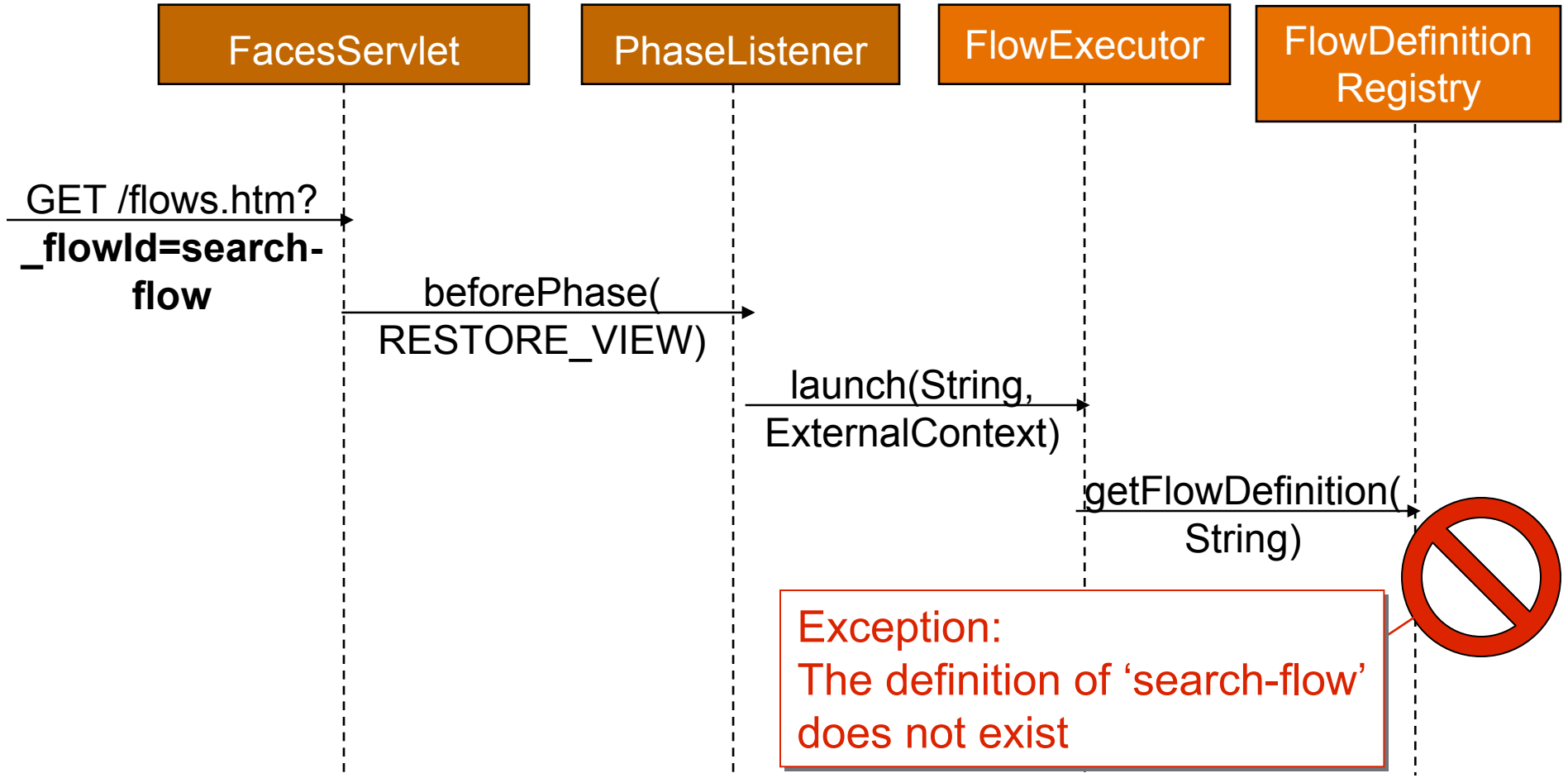
Testing Invoking flows.htm Directly



Important Flow Executor Arguments

- **flowId**
  - Needed to launch a new flow execution
- **flowExecutionKey**
  - Needed to resume an existing flow execution
    - Tracked for you by JSF™

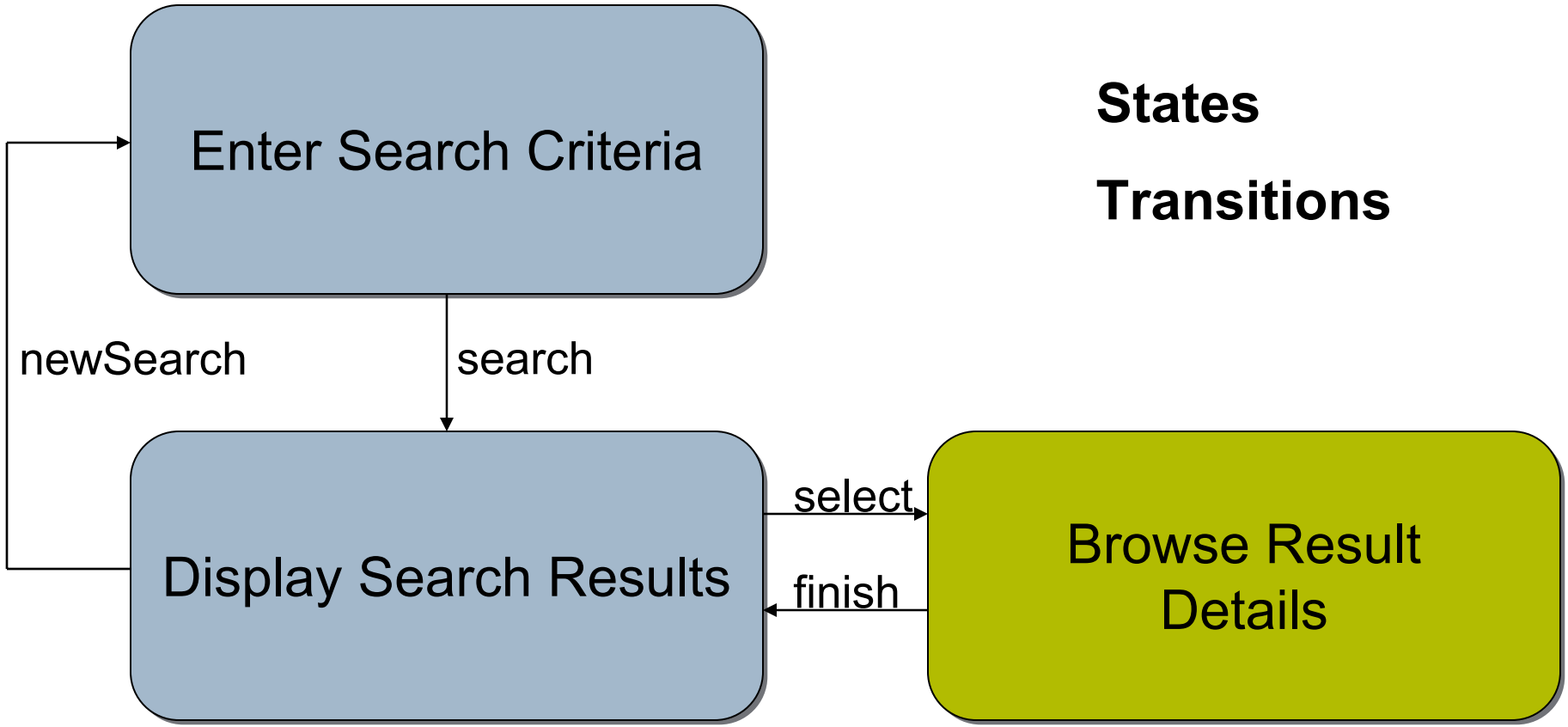
flows.htm?_flowId=search-flow



Quick Start Steps

- Bootstrap the Contacts Application
- Configure the Web Flow Engine
- **Implement the Search Flow**

Recall—Contact Search Flow



States

Transitions

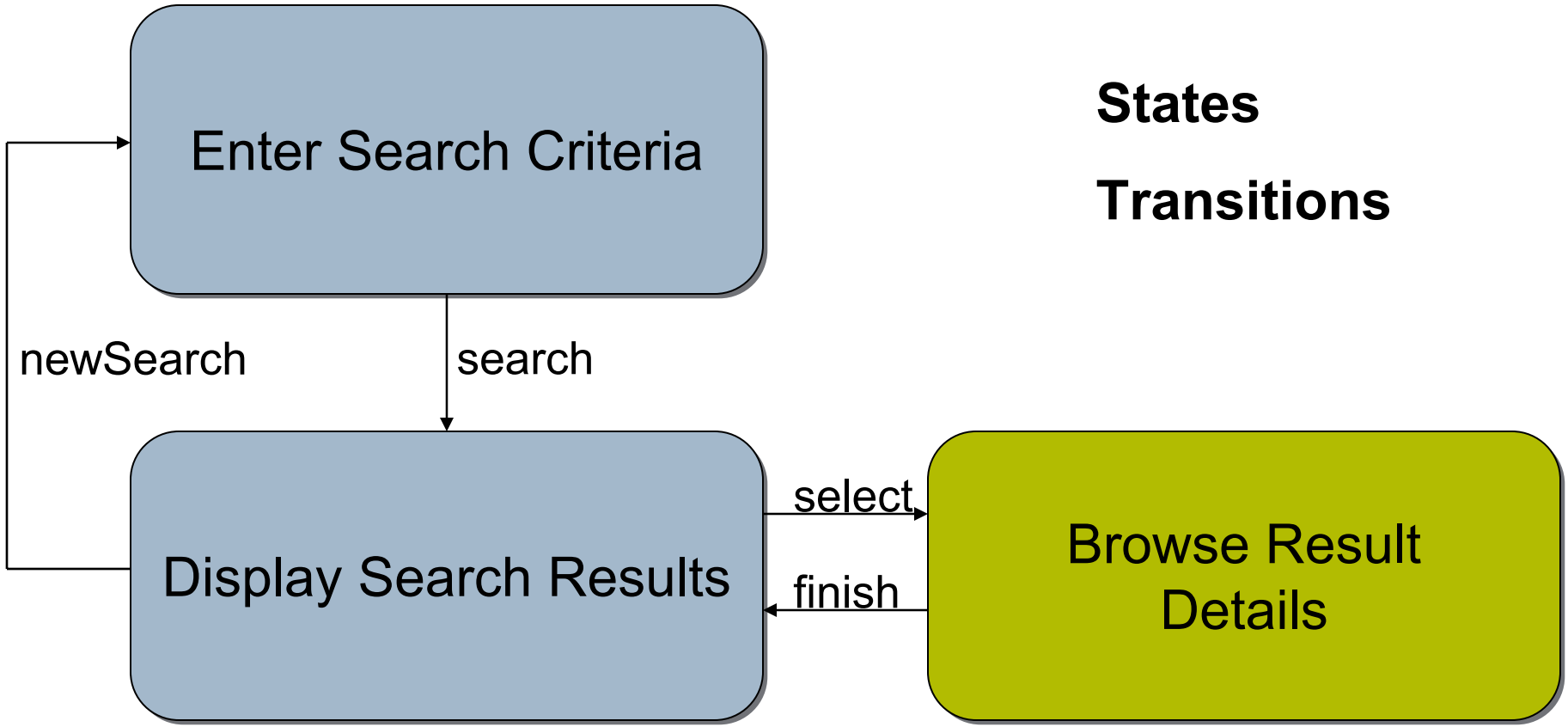
XML Flow Definition Template

```
<?xml version="1.0" encoding="UTF-8"?>
<flow xmlns=http://www.springframework.org/webflow
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
                http://www.springframework.org/schema/webflow
                http://www.springframework.org/schema/webflow/spring-webflow-
1.0.xsd">

        <!-- Implement your flow here -->

</flow>
```

DEMO

Implementing the Search Flow

Putting It All Together—Graphical



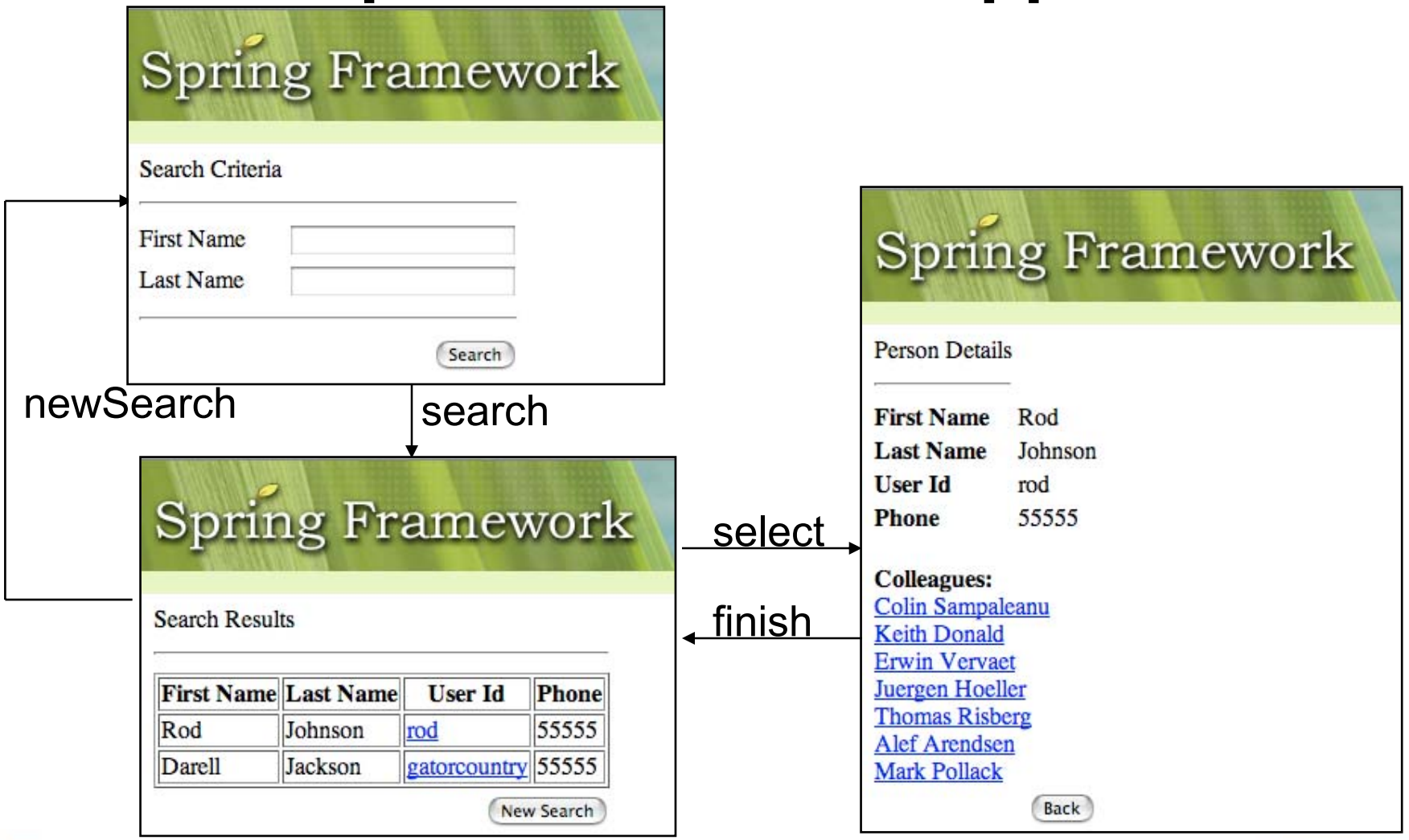
Putting It All Together—Search Flow

```
<flow …>
   <start-state idref="enterSearchCriteria"/>

   <view-state id="enterSearchCriteria" view="searchCriteriaForm">
     <transition on="search" to="displaySearchResults"/>
   </view-state>

   <view-state id="displaySearchResults" view="searchResults">
      <render-actions>
         <bean-action bean="searchService" method="search">
            <method-arguments>
               <argument expression="flowScope.searchCriteria"/>
            </method-arguments>
            <method-result name="results"/>
         </bean-action>
      </render-actions>
      <transition on="select" to="browseDetails"/>
   </view-state>
   …
```

Putting It All Together—Search Flow (2)

```
…
<subflow-state id="browseDetails" flow="details-flow">
    <attribute-mapper>
        <input-mapper>
            <mapping source="requestParameters.id" target="id" from="string" to="long"/>
        </input-mapper>
    </attribute-mapper>
    <transition on="finish" to="displayResults"/>
</subflow-state>

</flow>
```

Completed Contacts Application



Topics in This Session

Web Flow Quick Start

Key Benefits

What's Next

Key Benefits

- Enforcement of navigation rules
- Automatic state management
- Modularity
- Abstraction
- Dynamicity
- Web best practices
- +1 with strong integration
- Tooling

Enforcement of Navigation Rules

- With Web Flow, it is simply not possible to short-circuit navigation rules
  - Go ahead and try it :-)
- You get enforcement of navigation rules for free
- And you never have to worry about the flow resuming back in the right state

Automatic State Management

- You simply allocate local “flow scope” variables and the flow manages them for you
- When a flow ends or expires, everything in flow scope is cleaned up automatically

Modularity

- Flows are modules that encapsulate and centralize the rules to carry out a dialog with the user
- The sub-flow concept allows flows to be partitioned into manageable chunks describing high-level application functionality
  - You can change one flow without effecting another

Abstraction

- Web Flow definitions are high-level, self-contained application modules
  - They are not coupled with a particular environment like HTTP
- The same flow can be run inside a Spring MVC app, a Struts App, a JavaServer™ Faces application, a Portlet, or a JUnit test **without change**

Dynamicity

- Flows are externalized modules that can be recompiled on-the-fly
- Recompilation allows you to implement and test your navigation rules without container restart

Web Best Practices

- Back button just works
- Post + Redirect + Get pattern implemented out-of-the-box

+1 With Strong Integration

- Very easy to introduce into a project
  - Self-contained library with few extra dependencies (only need ognl.jar and spring-binding.jar extra)
  - Strong documentation
  - Wide-array of samples and other resources
  - Minimal setup to get going
- Compliments existing controller technologies
  - Fills a gap by solving the problem of linear wizards very well
  - Struts, Spring MVC, JavaServer Faces platform, and JUnit are all supported environments

Tooling

- Spring IDE 2.0 provides
  - A web flow graphical editor/visualizer
  - A web flow XML editor
- Visualizing flows improves developer and end-user communication
- Makes the tool accessible to new audiences

What's Next

- Web Flow 1.0 set a strong foundation
- Spring Web Flow will keep evolving with 1.1

On the Road to Web Flow 1.1…

- Flow Inheritance and Composition
  - The ability to compose flows together to form a graph/hierarchy
- Conversation-scoped persistence contexts
  - Automatic management of persistence state
- Enhanced Java-based flows
  - Engineering flows from @Annotation metadata
- More third-party integration
  - Securing flows (Spring Security/Acegi)
  - Ajax integration with ICEFaces and Ajax4JSF

Q&A

Thank You

Spring Web Flow 1.0 A Next-Generation Web Application Controller Framework

Keith Donald

Interface21
www.interface21.com

TS-6821