



---

# Java EE Architecture with the Spring Framework

Peter Thomas

Satyam Computer Services Ltd.



# Overview

---

- Spring: quick intro
- JTrac – a real-life Spring web-app
- Architecture
  - DI / IoC
  - Spring DAO / Hibernate support
  - Spring AOP / Declarative TX
  - Spring MVC / Webflow
  - Acegi Security Framework
  - Spring Modules
- Spring implications
- Selected best practices

# Spring – History



Nov 2002

Feb 2003: SourceForge Project founded

Aug 2003: 1.0 M1

Mar 2004: 1.0



Jul 2004

# JTrac – http://jtrac.info

- JTrac: highly customizable issue tracking

The screenshot displays the JTrac web application interface. At the top, there are navigation tabs for 'MY TRACKERS', 'DEMO ISSUES', 'NEW', and 'QUERY'. Below this, there are filters for 'View Item by Id' and 'Results / page' (set to 25). A 'Logged By' dropdown menu is visible, listing users like Ashutosh Chetnagar, Guest User, Jupiter Admin, Madhusudan Rao, Peter Thomas, and Tanuj Mathur.

The main content area features a table of trackers with columns for Tracker, Role, Action, Logged By Me, Assigned To Me, Assigned, Fixed, Verified, Closed, Rejected, Rej-Closed, and Total. The table lists several trackers such as DEMO-ISSUES, DEMO, ADMS-TASKS, and DEMO DEFECTS.

Below the tracker table, there is a section for 'All Trackers' with a summary row: All Trackers | QUERY | 21 | 3 | 21 | 3 | 1 | 28 | 1 | 1 | 53.

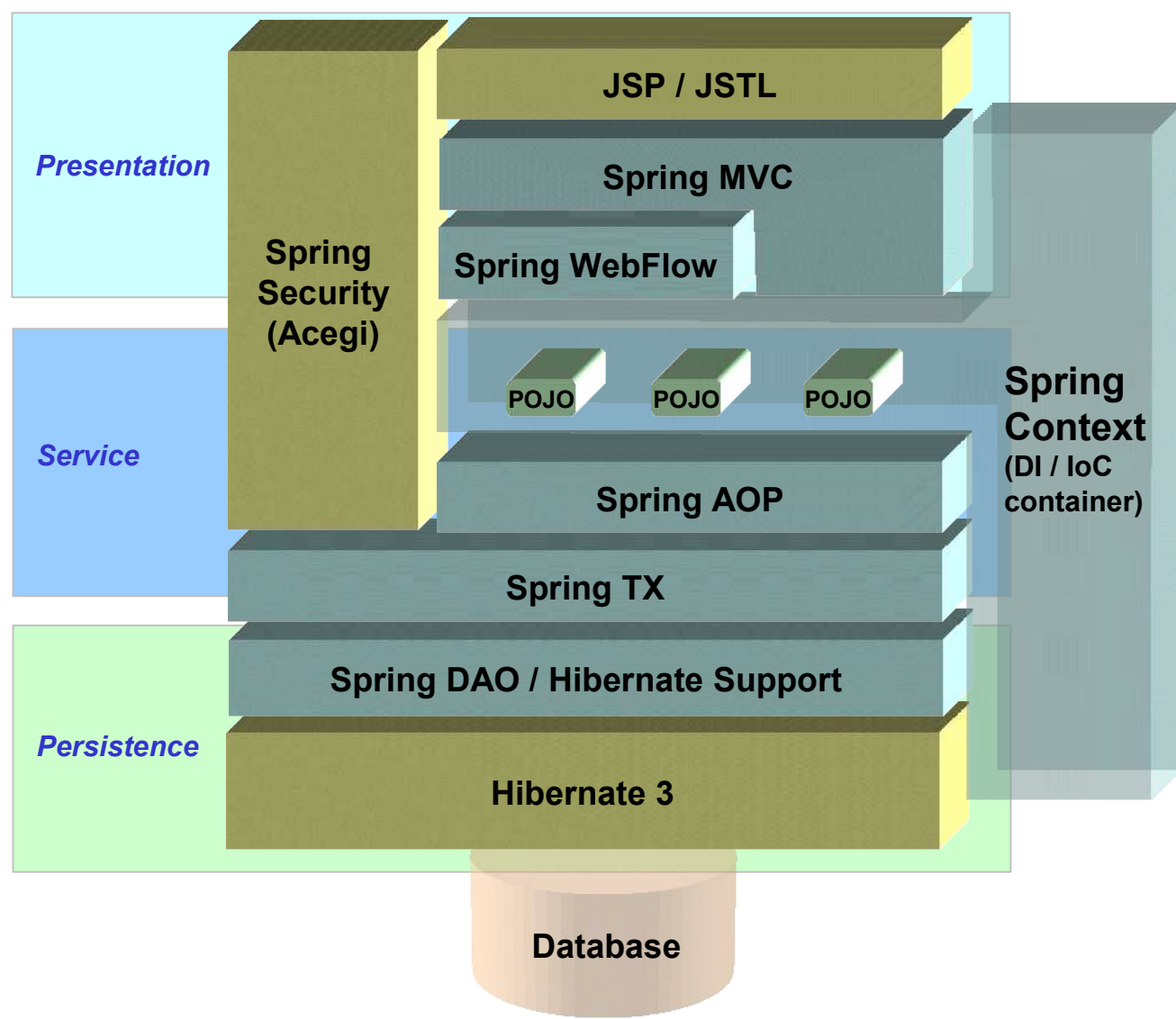
The bottom part of the screenshot shows a detailed view of an issue. It includes a 'Title' field, a 'Description' field, and a 'Severity' dropdown menu. A 'Notify By Email' section is also visible, listing users like Peter Thomas, Guest User, Jupiter Admin, Madhusudan Rao, and Tanuj Mathur.

On the right side, there is a 'Reject or put On Hold' section with a table of issue status transitions:

	Assigned	Fixed	Verified	Closed	Rejected	On Hold	Rej Closed
X					X	X	
X	X				X	X	
X	X						
X						X	
X				X		X	
X							

Below this table, there is a list of issues with columns for Title, Status, Logged By, Assigned To, Severity, Priority, and Date Logged. The list includes issues like 'to create separate...', 'my screens to be improved', 'Logon screen eq. ed', 'Java webtools should start with overview', 'ACK 3-TASKS', 'Type of user', 'need to be identified need to use Config wizard form', 'I have added themes as far as possible', 'Layout of screens to be improved', 'Default Controller Test to be improved', 'Misc report for bug form', and 'Feature request for default controller test'.

# JTrac Architecture



# Dependency Injection

```
<bean id="jtrac" class="info.jtrac.JtracImpl">  
  <property name="dao" ref="dao"/>  
</bean>
```

JtracImpl.java

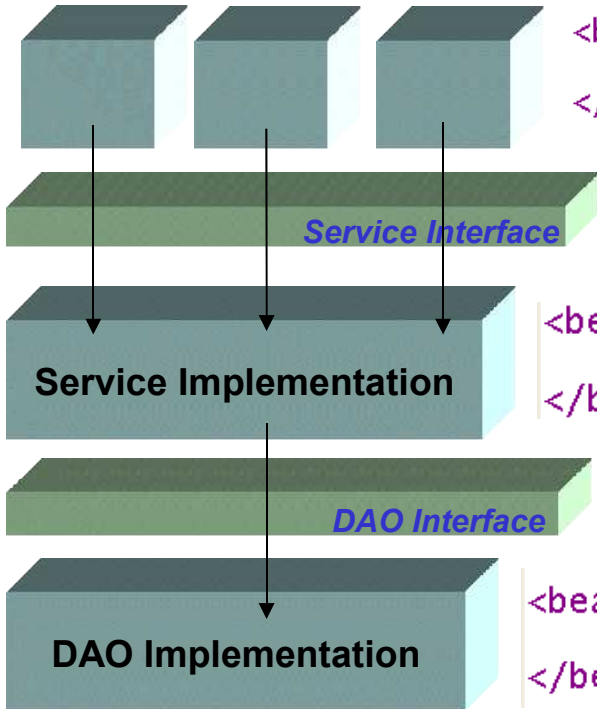
```
private JtracDao dao;  
  
public void setDao(JtracDao dao) {  
    this.dao = dao;  
}  
  
public void storeUser(User user) {  
    dao.storeUser(user);  
}
```

```
<bean id="dao" class="info.jtrac.hibernate.HibernateJtracDao">  
  <property name="sessionFactory" ref="sessionFactory"/>  
</bean>
```



# Dependency Injection (2)

## Controllers



```
<bean id="userFormAction" class="info.itrac.webflow.UserFormAction"
  <property name="jtrac" ref="jtrac" />
</bean>
```

```
<bean id="jtrac" class="info.itrac.JtracImpl">
  <property name="dao" ref="dao" />
</bean>
```

```
<bean id="dao" class="info.jtrac.hibernate.HibernateJtracDao">
  <property name="sessionFactory" ref="sessionFactory" />
</bean>
```



# Hibernate with Spring

---

- One liners
- Template pattern
- No need to deal with TX, Session etc.

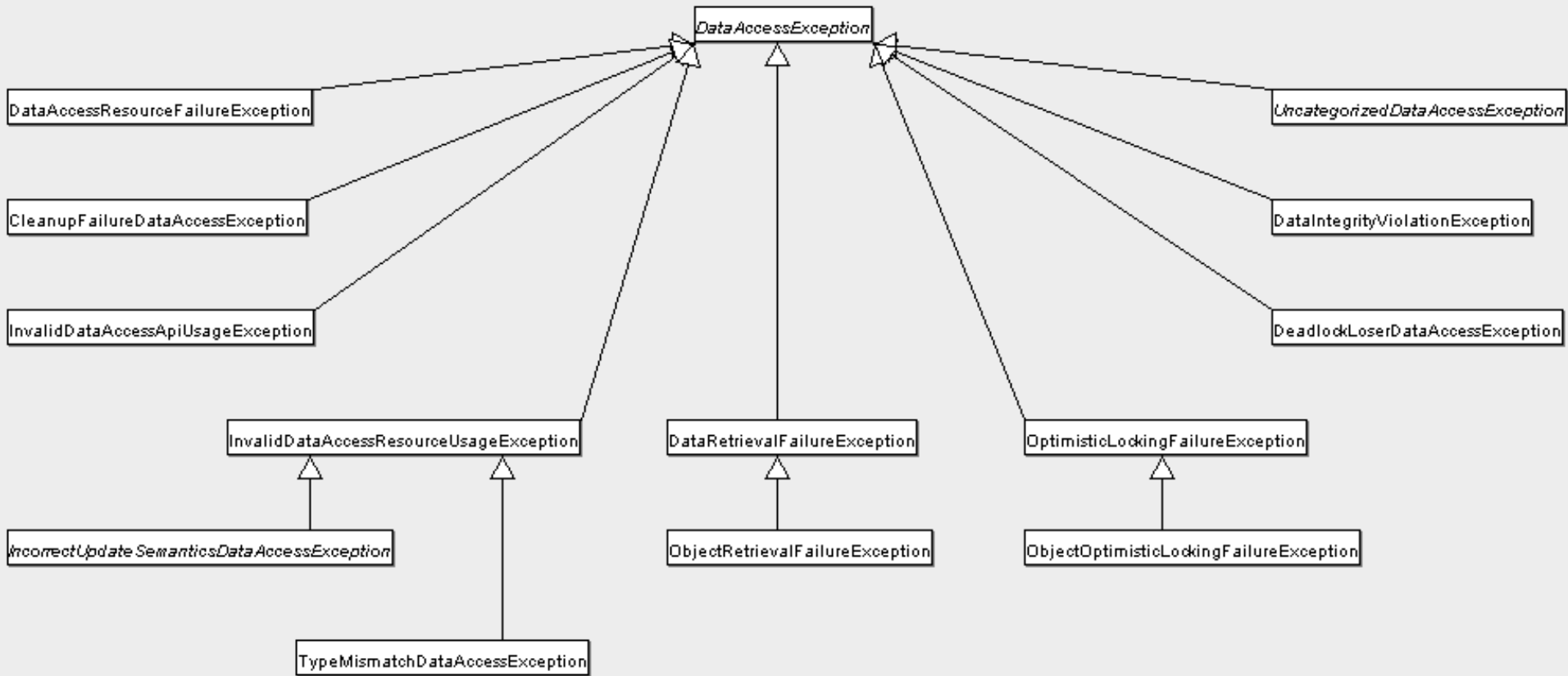
```
public void storeUser(User user) {
    getHibernateTemplate().merge(user);
}

public User loadUser(long id) {
    return (User) getHibernateTemplate().get(User.class, id);
}

public List<User> findAllUsers() {
    return getHibernateTemplate().find("from User user order by user.name");
}
```



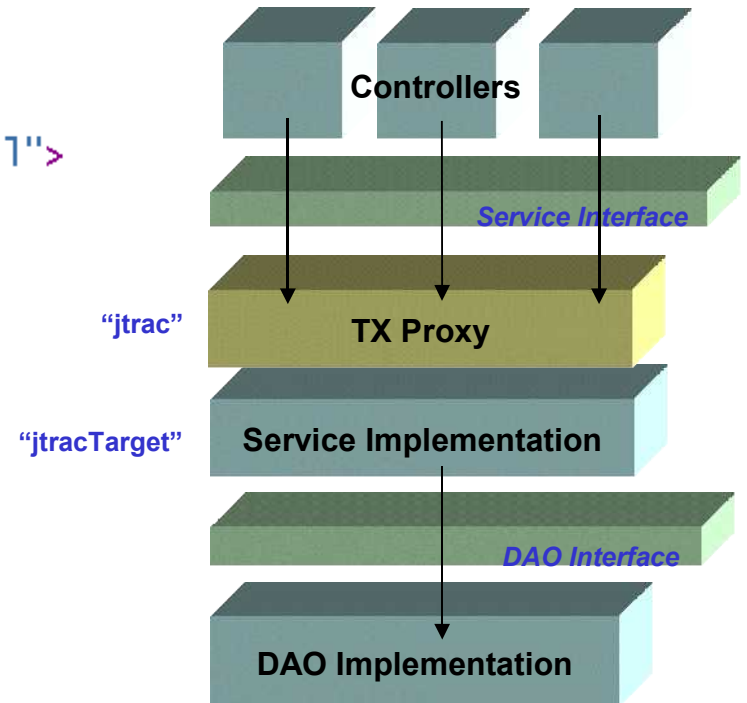
# Spring DAO – Exception Hierarchy



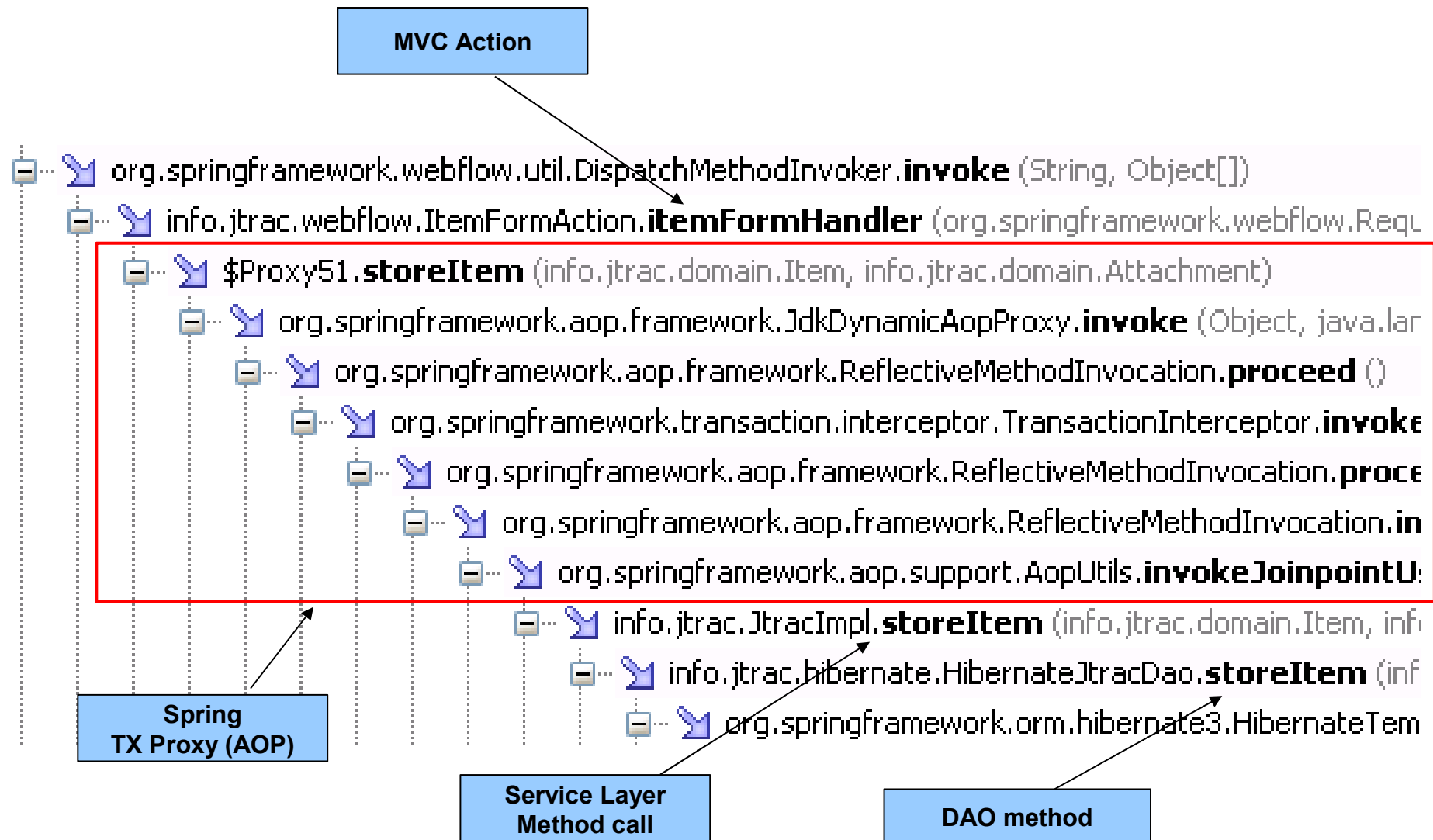
# Spring AOP / Declarative TX Mgmt.

```
<bean id="jtrac" class="org.[sf].transaction.interceptor.TransactionProxyFactoryBean">
  <property name="transactionManager" ref="transactionManager"/>
  <property name="target" ref="jtracTarget"/>
  <property name="transactionAttributes">
    <props>
      <prop key="store*">PROPAGATION_REQUIRED</prop>
      <prop key="remove*">PROPAGATION_REQUIRED</prop>
      <prop key="*">PROPAGATION_SUPPORTS,readOnly</prop>
    </props>
  </property>
</bean>

<bean id="jtracTarget" class="info.jtrac.JtracImpl">
  <property name="dao" ref="dao"/>
</bean>
```



# Spring AOP (contd...)



# Spring Web Flow

```
<flow start-state="userForm">
```

```
<input-mapper>  
  <mapping source="space" target="flowScope.space"/>  
</input-mapper>
```

```
<view-state id="userForm" view="user_form">  
  <entry-actions>  
    <action bean="userFormAction" method="setupForm"/>  
  </entry-actions>
```

```
<transition on="cancel" to="userListView"/>  
<transition on="submit" to="checkIfSpaceFlow">  
  <action bean="userFormAction" method="bindAndValidate"/>  
  <action bean="userFormAction" method="userFormHandler"/>  
</transition>
```

```
</view-state>
```

```
<decision-state id="checkIfSpaceFlow">  
  <if test="${flowScope.space == null}" then="userAllocateFlow" else="userListView"/>  
</decision-state>
```

```
<subflow-state id="userAllocateFlow" flow="userAllocate">  
  <attribute-mapper>  
    <input-mapper>  
      <mapping source="requestScope.userForm.user" target="user"/>  
    </input-mapper>  
  </attribute-mapper>  
  <transition on="userListView" to="userListView"/>  
</subflow-state>
```

```
<end-state id="userListView" view="userListView">  
  <output-mapper>  
    <mapping source="flowScope.space" target="requestScope.space"/>  
  </output-mapper>  
</end-state>
```

```
</flow>
```

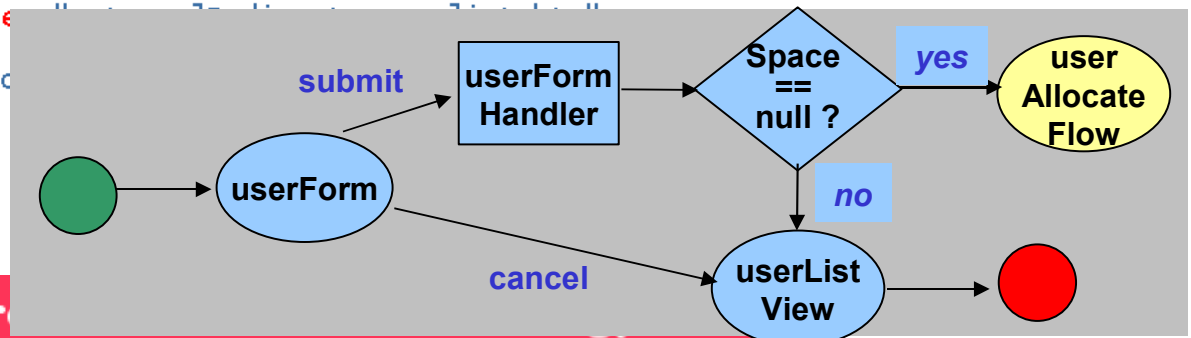
Logical View Name

State Transitions

backend logic call (normal Java method)

Flow Decision

Re-usable Flow invocation





# Security with Acegi

```
<bean name="authorizationFilter" class="org.acegisecurity.intercept.web.Fi
  <property name="authenticationManager" ref="authenticationManager"/>
  <property name="accessDecisionManager">
    <bean class="org.acegisecurity.vote.AffirmativeBased">
      <property name="allowIfAllAbstainDecisions" value="false"/>
      <property name="decisionVoters">
        <list>
          <bean class="org.acegisecurity.vote.RoleVoter"/>
        </list>
      </property>
    </bean>
  </property>
  <property name="objectDefinitionSource">
    <value>
      PATTERN_TYPE_APACHE_ANT
      /login.htm*=ROLE_ANONYMOUS,ROLE_USER
      /logout.htm=ROLE_ANONYMOUS,ROLE_USER
      /admin/**=ROLE_ADMIN
      /**=ROLE_USER
    </value>
  </property>
</bean>
```

any URL request within the "/admin/" directory or path can only be allowed for "ROLE\_ADMIN"

Any other request has to be allowed only for authenticated users



# Spring Modules example: Lucene

---

```
<bean id="indexDirectory" class="org.[sm].lucene.index.support.FSDirectoryFactoryBean">
  <property name="location" value="file:///${jtrac.home}/indexes"/>
</bean>

<bean id="indexFactory" class="org.[sm].lucene.index.support.SimpleIndexFactoryBean">
  <property name="directory" ref="indexDirectory"/>
  <property name="analyzer" ref="analyzer"/>
</bean>

<bean id="analyzer" class="org.apache.lucene.analysis.SimpleAnalyzer"/>

<bean id="indexer" class="info.jtrac.lucene.Indexer">
  <property name="indexFactory" ref="indexFactory"/>
</bean>
```

```
import org.springmodules.lucene.index.support.LuceneIndexSupport;

public class Indexer extends LuceneIndexSupport {

    public void index(AbstractItem item) {
        getTemplate().addDocument(item);
    }
}
```



# JTrac: Spring implications

---

- Application deployed as a WAR file
- Portable: Tomcat, Jetty, JBoss or any Java EE app server
- Easily unit-testable
- Faster build-deploy-test cycles
- Clean OO design, designed to interfaces
- No EJBs
- No Singletons
- No Service Locator / JNDI lookup
- No custom “Factory Pattern” implementation
- No DTOs / VOs
- No Annotations (optional)
- JSTL / JSP



# Why not RoR :)

---

- Spring
- Hibernate
  - Lazy Loading
- HSQLDB
  - Embedded database
- Jetty
  - Small footprint
  - Embedded app-server + web server !
- Libraries
  - JavaSVN, JFreeChart, Lucene, Apache POI
- i18n



# Notable in Spring 2.0

---

- Async task execution
- Portlet MVC
- Use Groovy, JRuby etc. for config
- MVC: Custom taglib for form controls (like Struts)
- Message driven POJOs
- JPA support
- Simplified XML config option
- AspectJ integration



# Agile Development with Spring + Hibernate

- **Spring**
  - No need of container
  - JUnit support classes
  - POJOs inherently easy to test
- **Hibernate**
  - Completely abstracts DB
  - HSQLDB can be used for testing
  - Unit tests can assume that DB exists, no mocks reqd.

*clean*

Delete all previous build artifacts / database files from file system

*compile*

Compile all code and unit tests

*db-start*

Boot a fresh instance of the HSQLDB database server

*db-create*

Connect to the database and forward-generate the schema from the mapping files using Hibernate.

*test*

Run unit-tests.

*db-stop*

Shutdown database.

*reports*

Generate Reports



# DEMO

---

- Unit + Integration Testing



# Watij [ <http://watij.com/> ]

---

- A new class of web test automation tools
  - Watir
    - Web Application Testing in Ruby
  - Selenium
  - Watij
    - Web Application Testing in Java
- Focus on using a real browser, not simulating one
- Especially important when lots of Javascript / AJAX



# Watij Features

---

- Write tests in pure Java
- Interactively script tests using BeanShell
- Possible to drive tests using JUnit
- No setup required on the Server Under Test
- Support for nested tables, frames
- Good “popup” and multi-window support
- And also “brute-force” keystroke support...



# Watij in conjunction with JUnit

The screenshot displays the NetBeans IDE interface. On the left, the 'JUnit Test Results' window shows a list of 8 tests, all of which passed. Three tests are highlighted with red boxes: 'testGetLoginPage', 'testSuccessfulLogin', and 'testCreateNewSpaceAndAllocateAdmin'. On the right, the 'AllTest.java' source file is open, showing the implementation of these tests. Red boxes highlight the test method signatures and their corresponding code blocks. Red arrows point from the test results in the left window to the corresponding code in the right window.

**JUnit Test Results:**

- info.jtrac.watij.AllTest passed
- testGetLoginPage passed (7.891 s)
- testSuccessfulLogin passed (1.793 s)
- testCreateNewSpaceAndAllocateAdmin passed
- testCreateNewItem passed (2.634 s)
- testSearchAllContainsItem passed (1.121 s)
- testUpdateHistoryForItem passed (0.471 s)
- testCreateNewUser passed (2.474 s)
- testLogout passed (4.546 s)

**Code Snippets:**

```
package info.jtrac.watij;

import ...

public class AllTest extends WatijTestCase {

    static {
        clazz = AllTest.class;
    }

    public AllTest(String name) {
        super(name);
    }

    public void testGetLoginPage() throws Exception {
        ie.start("http://localhost:8080/jtrac/auth/login.htm");
        assertTrue(ie.containsText("JTrac"));
    }

    public void testSuccessfulLogin() throws Exception {
        ie.textField(name, "j_username").set("admin");
        ie.textField(name, "j_password").set("admin");
        ie.button("Submit").click();
        assertTrue(ie.containsText("DASHBOARD"));
    }

    public void testCreateNewSpaceAndAllocateAdmin() throws Exception {
        ie.link(text, "OPTIONS").click();
        assertTrue(ie.containsText("Options Menu"));
    }
}
```



# DEMO

---

- Watij
- Driving functional tests with JUnit



---

# Thank you

<http://jtrac.info>