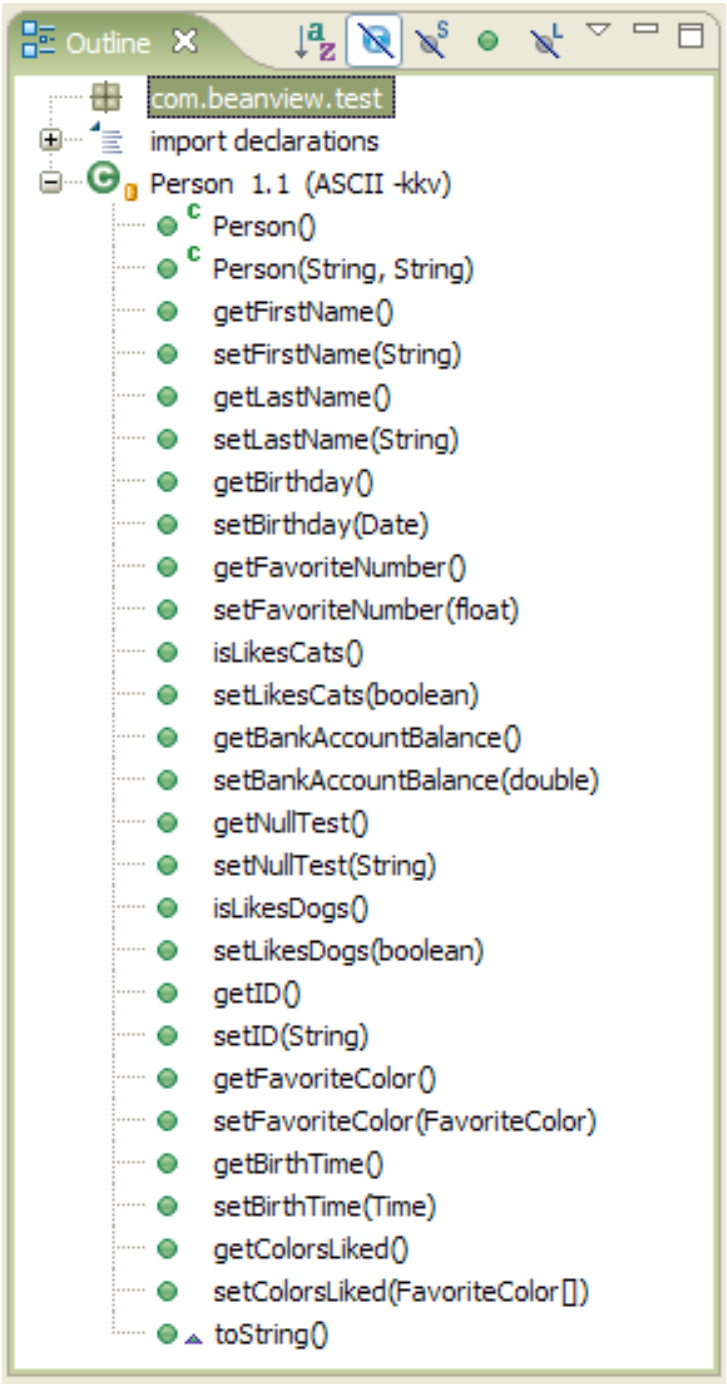
BeanView Screenshots

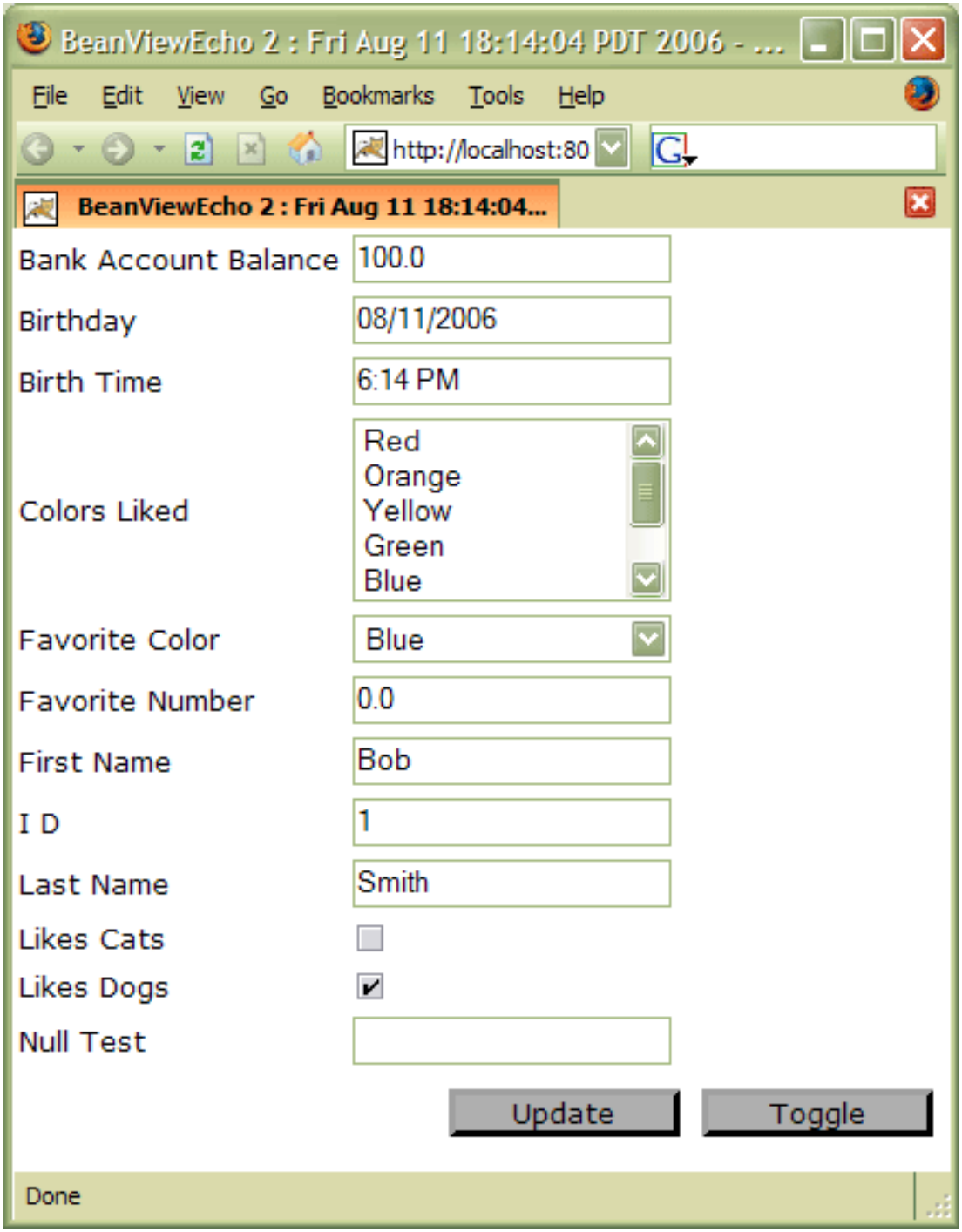
In the examples below, the form shown was generated via reflection via a simple Person object (an ordinary JavaBean). The reflection data is cached, which means that there is very little overhead for the actual display of the form, the validation, etc. The Person object shown is a completely unannotated JavaBean. BeanView supports the use of annotations for additional control over the displayed user interface.



- Echo 2 (Web) Implementation
- Swing (Fat Client) Implementation
- Echo 2 (Web) Validation Example
- Swing (Fat Client) Validation Example
- Complex Panel Example
- Simple Object Mapping Support
- Collection Support
- Naked Object Prototype
- Selenium Script Example

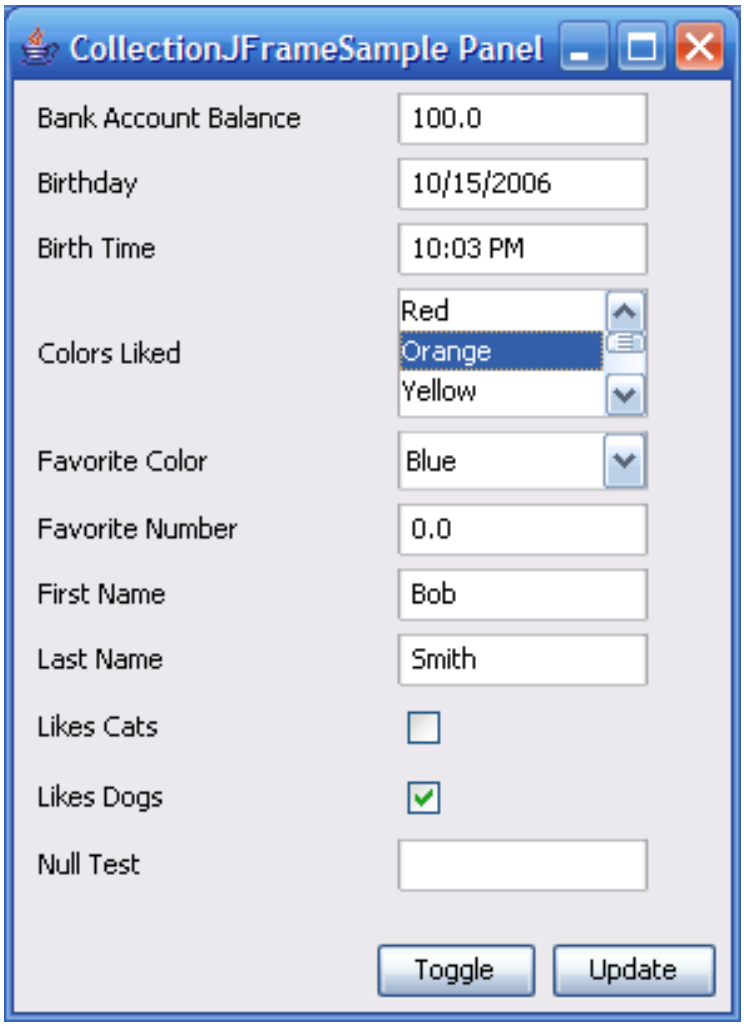
This Person JavaBean can be easily rendered into both a web user

interface (via Echo 2) and a fat client via Swing. In both cases, the form is simply a panel added to the relevant user interface. For the demonstrations below, the only application specific code needed is for laying out the form and adding the buttons for interactivity - all of the display and validation logic is generated automatically by BeanView.

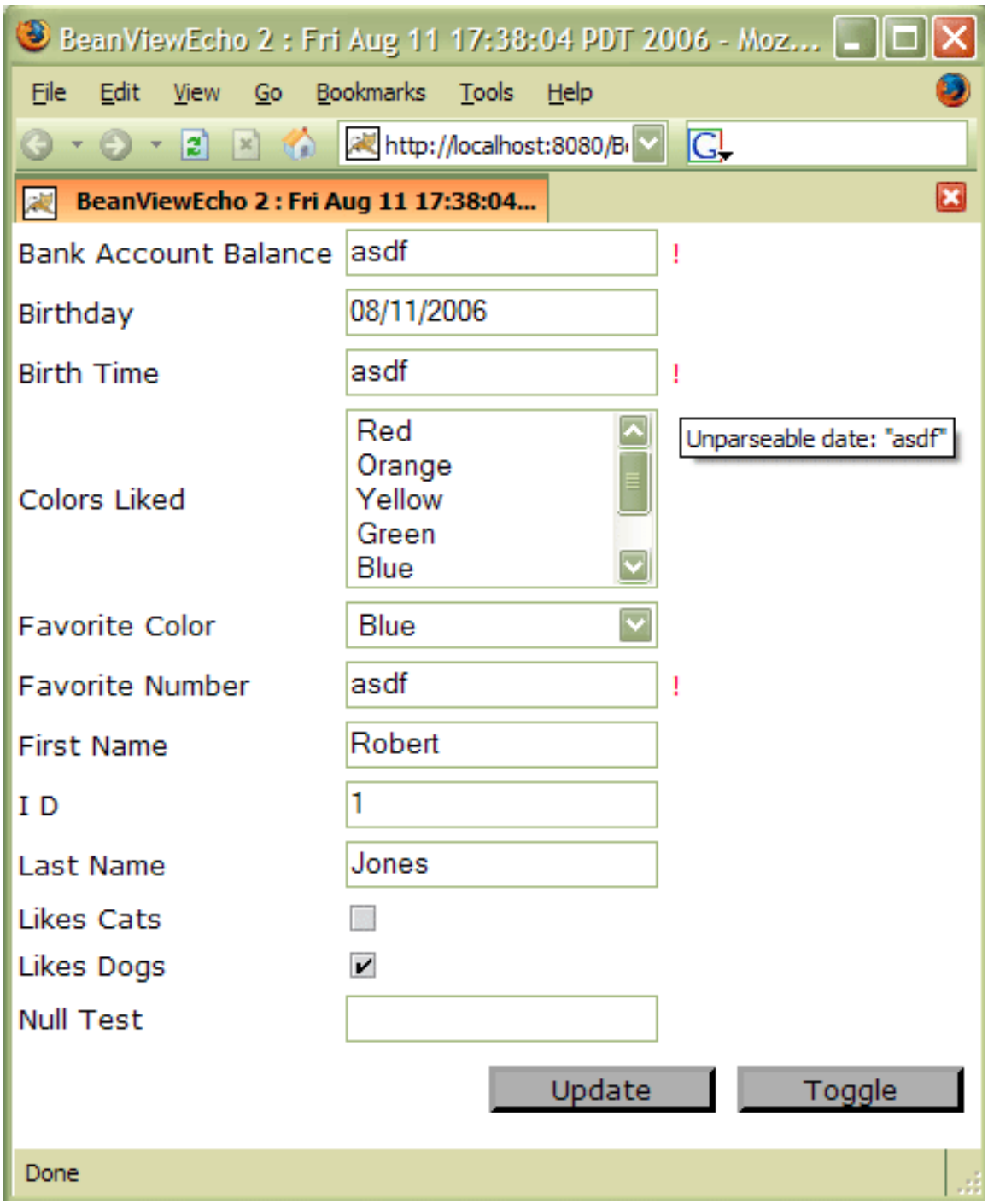


In this Echo (web) user interface, clicking the Update button tests the

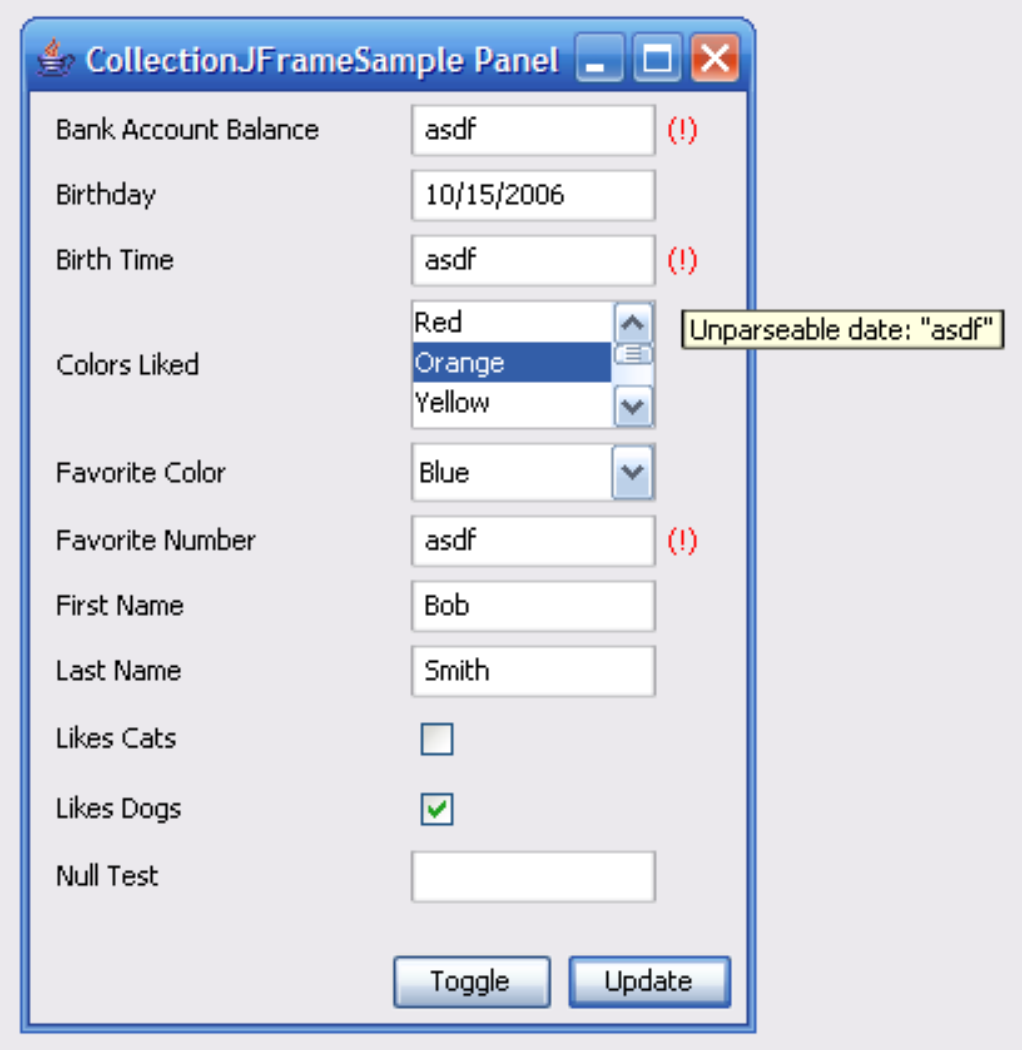
`updateObjectFromPanel()` method, and the Toggle button tests the `updatePanelFromObject()` method.



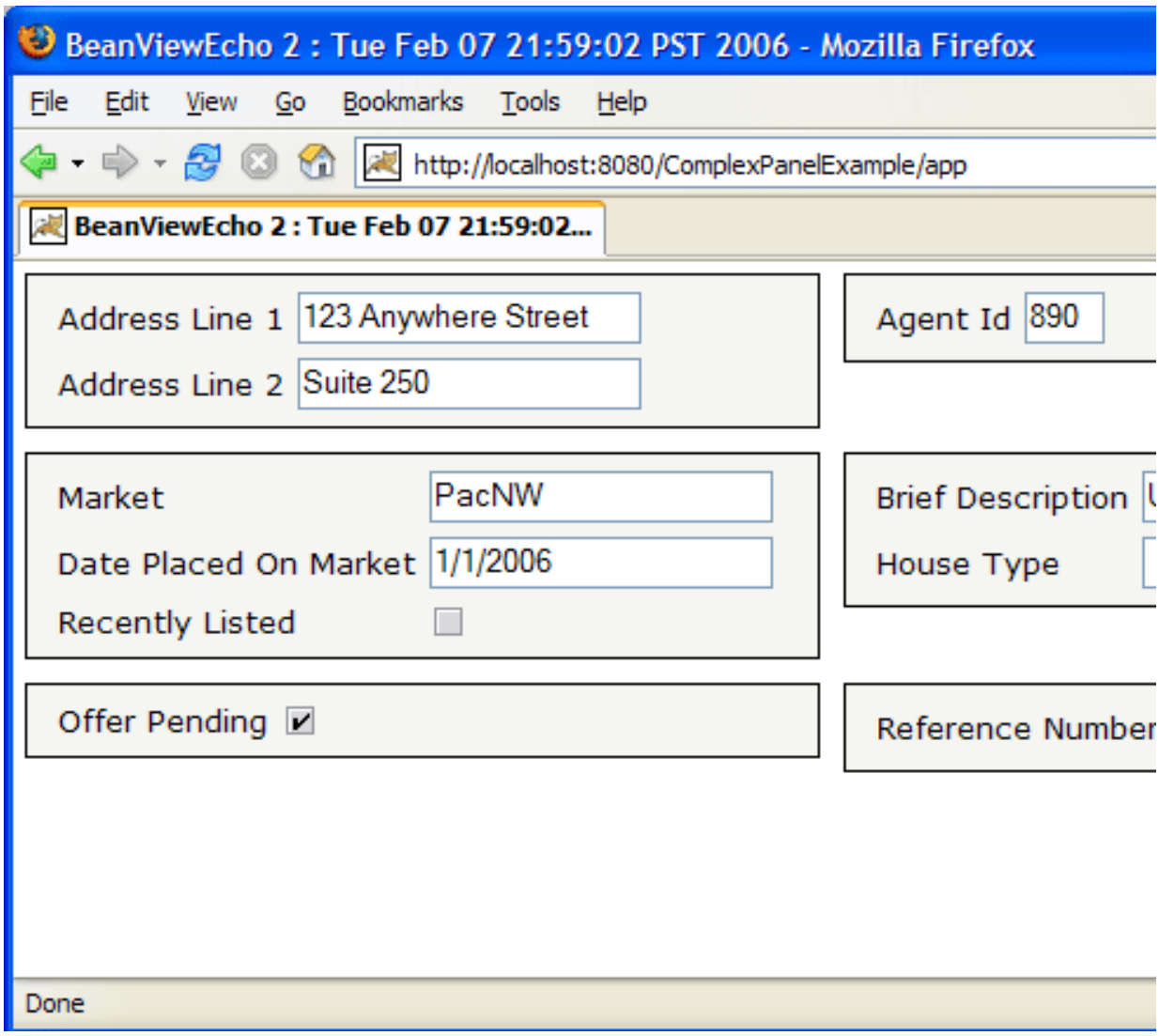
In this Swing (desktop) interface, clicking the Set Mary and Set Bob buttons tests the `updatePanelFromObject()` method, and the Update Mary button tests the `updateObjectFromPanel()` method.



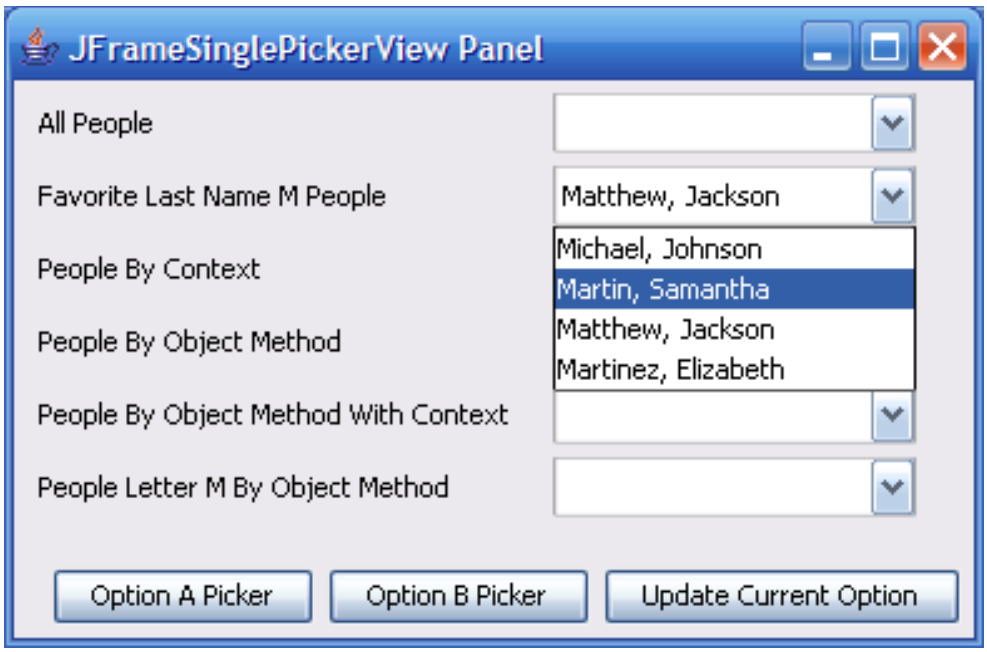
This example shows how errors in the user input are automatically generated by a call to `updateObjectFromPanel()` in a web user interface..



This example shows how errors are displayed in a Swing user interface from a call to `updateObjectFromPanel()`. Note that in the Swing user interface, the error message is displayed as a red exclamation point and a tooltip. It would be very easy to subclass the Swing panel to override this behavior.



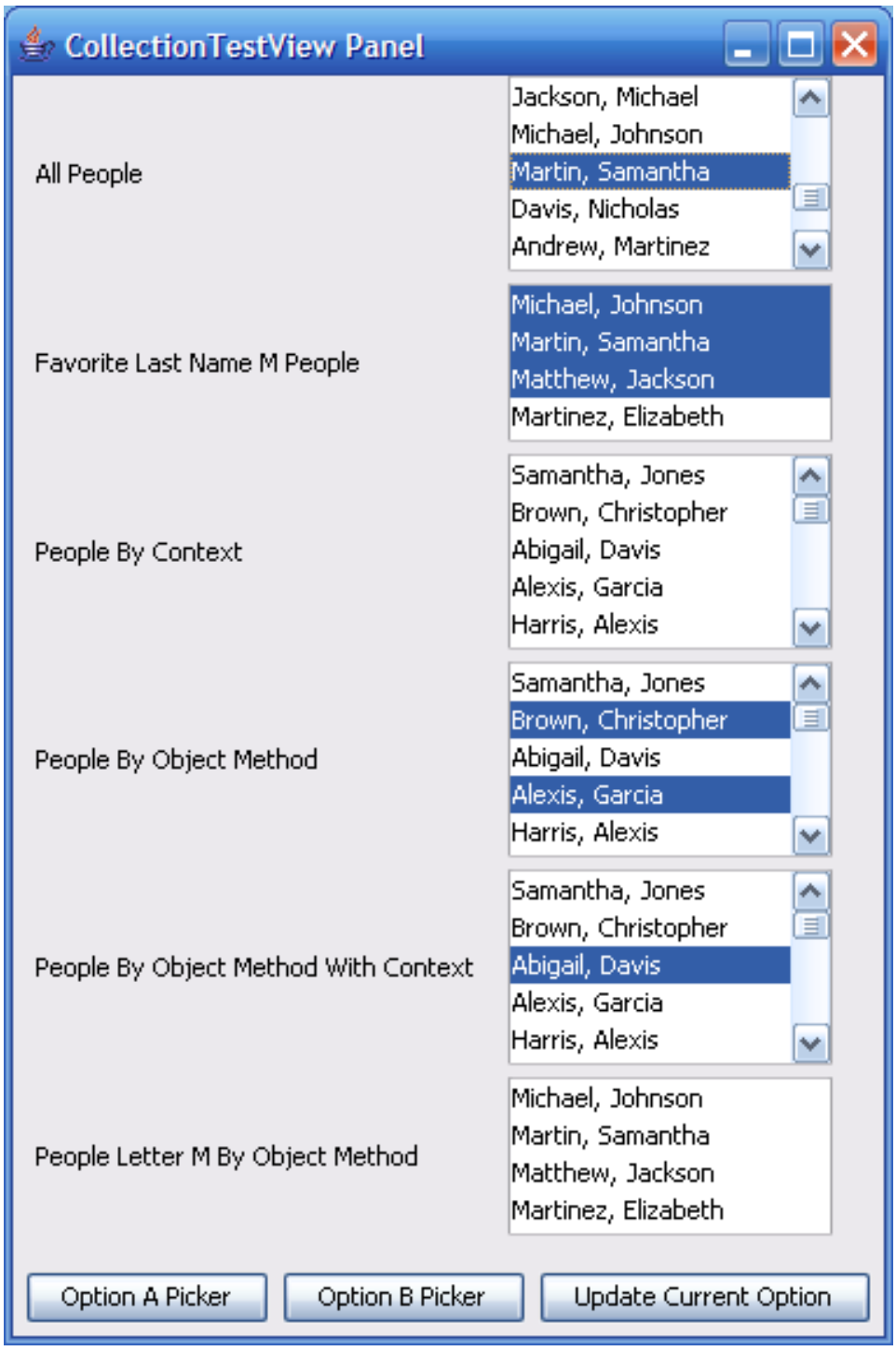
This example shows a more complex UI, in which the designer has taken advantage of the BeanView subview capability to exercise more control over the layout. All of these subpanels are generated from a single bean.



This screenshot shows how object relationships can be handled. In this case, the "Favorite Last Name M People" property is represented by the following property and annotation:

```
@PropertyOptions(options =
"SimpleObjectFactory.getLastNameStartsWithM")
public SimpleObject getFavoriteLastNameMPeople()
{
    return favoriteLastNameMPeople;
}
```

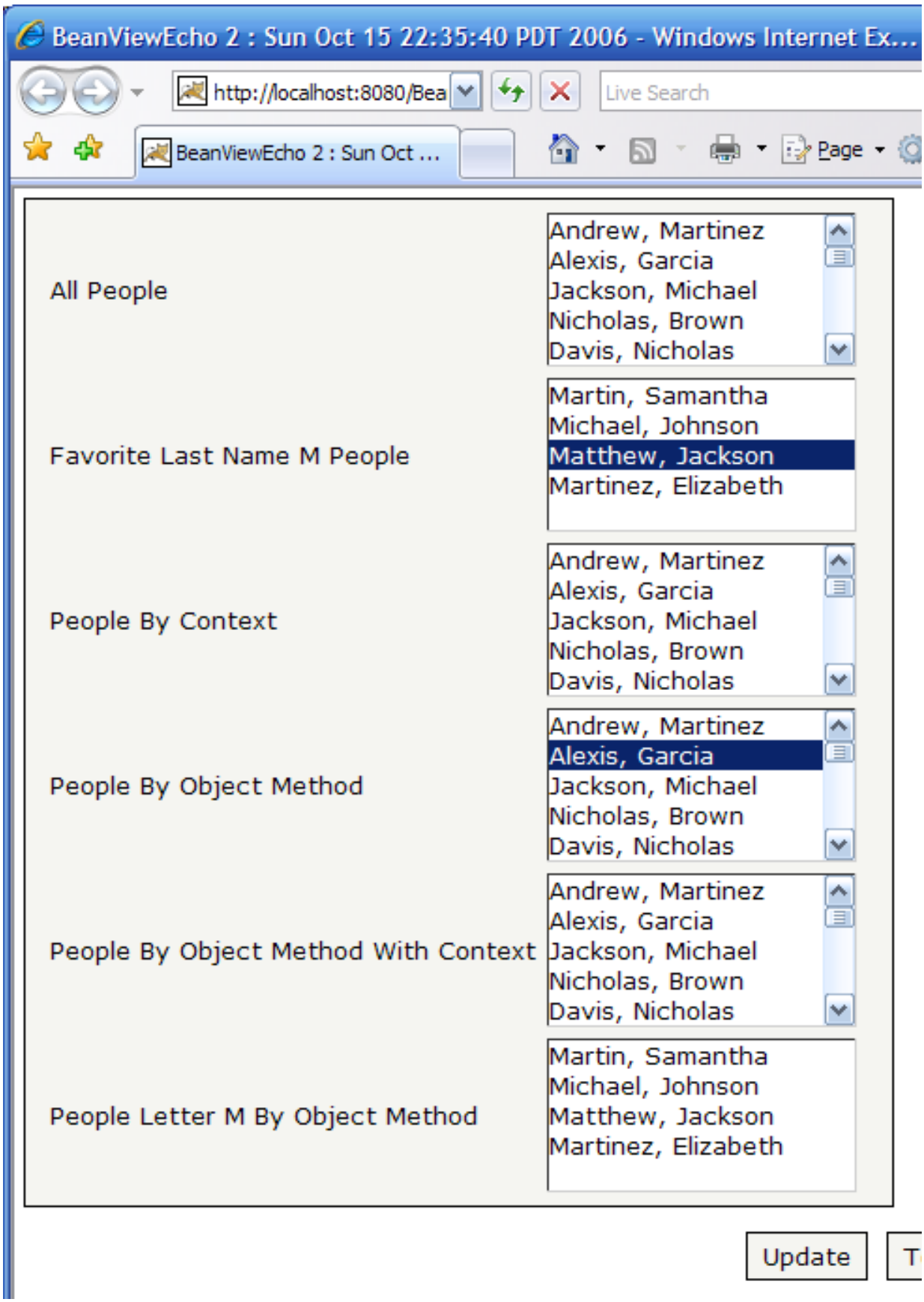
Note that the same factory methods are called by both the Swing and Echo 2 implementations.



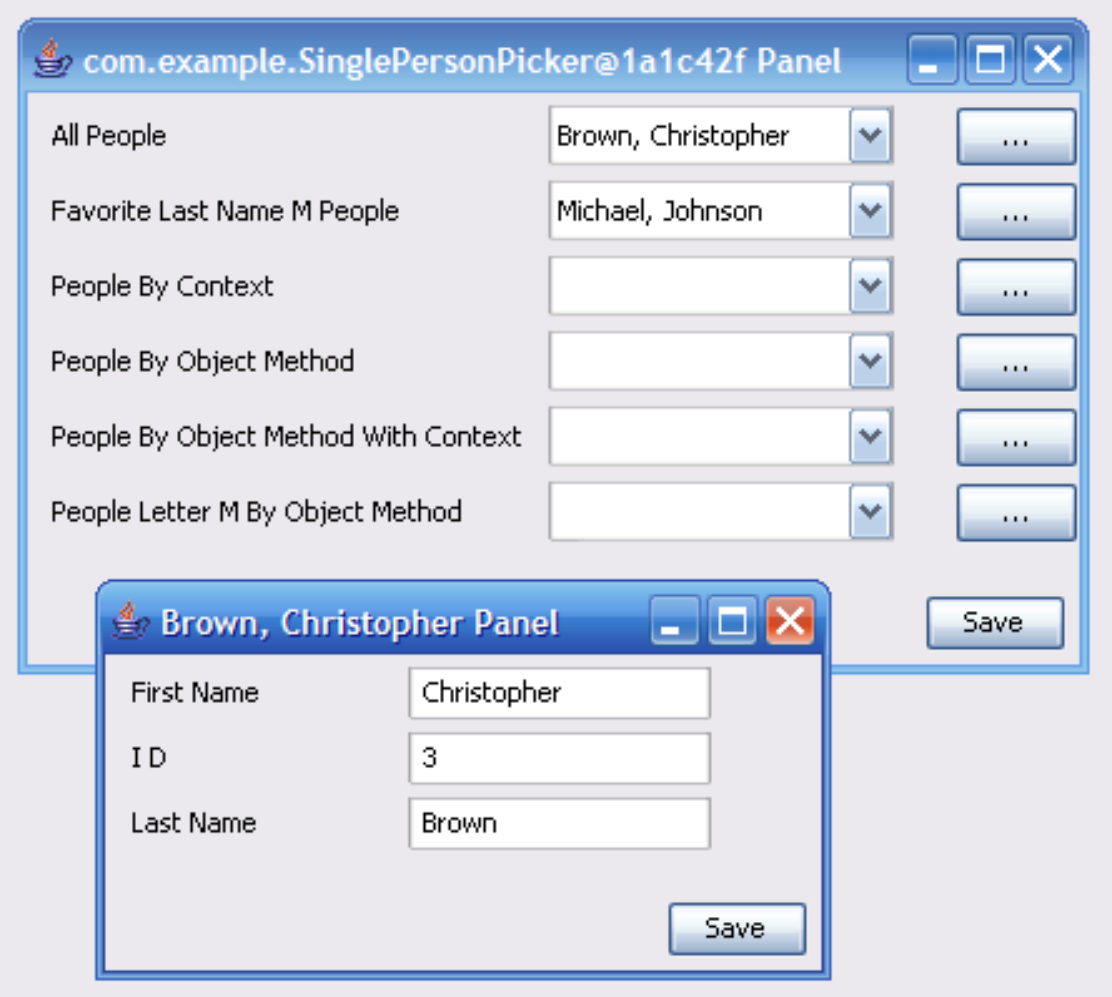
This screenshot shows how simple annotations and a variety of methods can be used to generate the underlying options. Each of these options is represented by a simple Collection<Person> property. The All People option points to a static method that returns all possible People. The People By Context method returns a subset of people based on the

current object. The Object method version allows you to put the logic directly in the JavaBean.

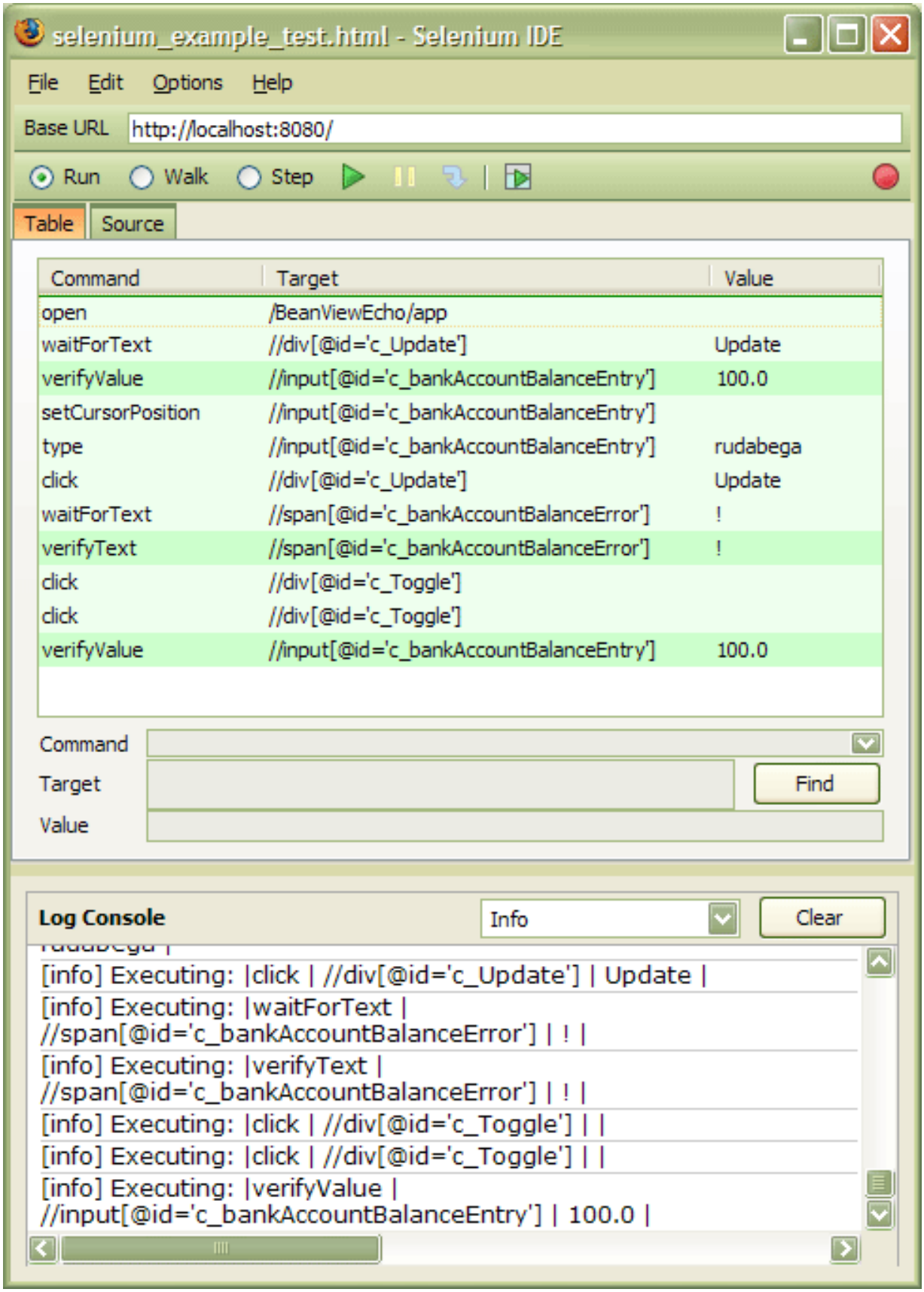
A simple example of the use of annotations in this fashion can be found at the BeanView blog.


BeanViewEcho 2 : Sun Oct 15 22:35:40 PDT 2006 - Windows Internet Ex...
http://localhost:8080/Bea
Live Search
BeanViewEcho 2 : Sun Oct ...
Page
All People
Andrew, Martinez
Alexis, Garcia
Jackson, Michael
Nicholas, Brown
Davis, Nicholas
Favorite Last Name M People
Martin, Samantha
Michael, Johnson
Matthew, Jackson
Martinez, Elizabeth
People By Context
Andrew, Martinez
Alexis, Garcia
Jackson, Michael
Nicholas, Brown
Davis, Nicholas
People By Object Method
Andrew, Martinez
Alexis, Garcia
Jackson, Michael
Nicholas, Brown
Davis, Nicholas
People By Object Method With Context
Andrew, Martinez
Alexis, Garcia
Jackson, Michael
Nicholas, Brown
Davis, Nicholas
People Letter M By Object Method
Martin, Samantha
Michael, Johnson
Matthew, Jackson
Martinez, Elizabeth
Update

Here is an example of the Echo 2 version of the previous collection demo.



This diagram shows a simple prototype of a Swing-based "Naked Objects" implementation using BeanView. While this implementation is not complete, it shows how BeanView can be customized to meet a variety of needs. See \beanview_example_swing\src\main\java\com \example\naked\NakedObjectDemo.java for more information.


selenium_example_test.html - Selenium IDE
File Edit Options Help
Base URL http://localhost:8080/
Run Walk Step
Table Source
Command Target Value
open /BeanViewEcho/app
waitForText //div[@id='c_Update'] Update
verifyValue //input[@id='c_bankAccountBalanceEntry'] 100.0
setCursorPosition //input[@id='c_bankAccountBalanceEntry']
type //input[@id='c_bankAccountBalanceEntry'] rudabega
click //div[@id='c_Update'] Update
waitForText //span[@id='c_bankAccountBalanceError'] !
verifyText //span[@id='c_bankAccountBalanceError'] !
click //div[@id='c_Toggle']
click //div[@id='c_Toggle']
verifyValue //input[@id='c_bankAccountBalanceEntry'] 100.0
Command
Target
Find
Value
Log Console
Info
Clear
[info] Executing: |click | //div[@id='c_Update'] | Update |
[info] Executing: |waitForText | //span[@id='c_bankAccountBalanceError'] | ! |
[info] Executing: |verifyText | //span[@id='c_bankAccountBalanceError'] | ! |
[info] Executing: |click | //div[@id='c_Toggle'] | |
[info] Executing: |click | //div[@id='c_Toggle'] | |
[info] Executing: |verifyValue | //input[@id='c_bankAccountBalanceEntry'] | 100.0 |

The latest release of Echo 2 supports setting Render IDs, which means that BeanView now works quite elegantly with Selenium. The image above shows the Selenium IDE view of a simple script validating the example Echo 2 view. The script shows a test which views the page, sets an incorrect value, gets a validation error, and then clicks the Toggle button twice to reset the page.

Return to http://www.beanview.com/

@author $Author: wiverson $
@version $Revision: 1.2 $, $Date: 2006/10/16 05:39:41 $