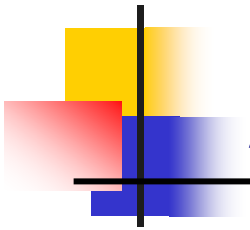
# Struts View Assembly and Validation

Gary D. Ashley Jr.

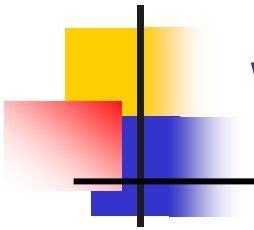
3rd Millennium Visions, Inc.

garyashley at toughguy dot net



# Agenda

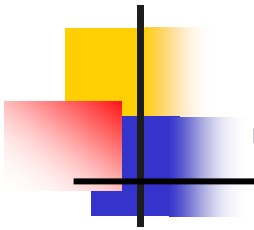
- What are common features of Struts?
- What are some new features in 1.1?
- What's in Struts Tag Libraries?
- Why use MessageResource bundles?
- What types of validation are provided?
- What can I do with Tiles?
- How do I …?



# What Is Struts?

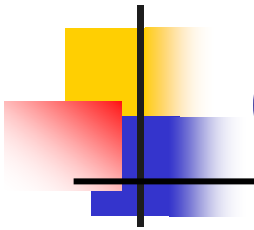
Struts – open source implementation of MVC design pattern for building web applications based on Servlet, JSP, XML, and Java.

- Controller – primary component is a servlet, ActionServlet.
- View – one of primary benefits of 1.0, and even 1.1 is custom tag library.
- Model – only general guidelines are provided by Struts.



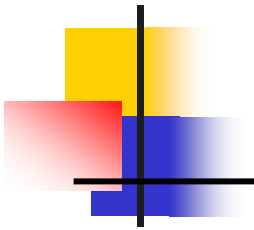
# Struts Facts

- Originally created by Craig McClanahan, and donated to Apache in May 2000. (also Tomcat and JSF)
- 27 packages (12 top level), and several hundred classes/interfaces.
- Built upon several Jakarta projects (most in Commons):
  - BeanUtils, Collections, Digester, Lang, Logging, Validator, FileUpload, and ORO.



# Common Features

- I18N Support (Internationalization)
- Powerful custom tag library
- Form Validation
- Supports various presentation layers (JSP, XML/XSLT, JSF, Velocity, …)
- Supports various model layers (JavaBeans, EJB, *etc.*)



# New Features to 1.1

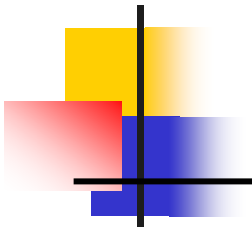
Main features:

- Module support
- Plugin Support
- Tiles Integration

Other important notes:

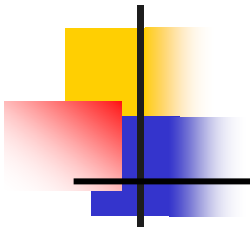
- JSTL and EL are supported.
- Early Access version of JSF is included.
- Scaffold

http://jakarta.apache.org/struts/userGuide/release-notes.html#Introduction



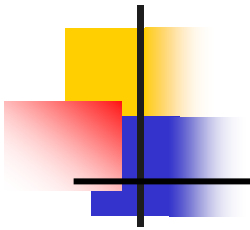
# New Features to 1.2.x

- At the writing of this presentation, 1.2 has not yet been released, but it is anticipated that one or more releases will be made prior to the summit in October.
- Some expected items will include removal of all deprecations since 1.0, and the integration of 1 or more new Jakarta Commons Projects.



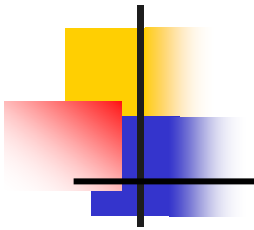
# Struts Tag Libraries

- Bean Tags
  - Tags useful in accessing/defining beans and their properties.
- HTML Tags
  - Tags useful in creating HTML output, but centers around input forms.
- Logic Tags
  - Tags useful in various logic operations, *i.e.* conditions, loops, and flow. If using JSTL, mostly obsolete.
- Nested Tags
  - Mostly an extension to other Struts libraries to work with tags in a nested context.
- Tiles Tags
  - Tags built to work with Tiles.



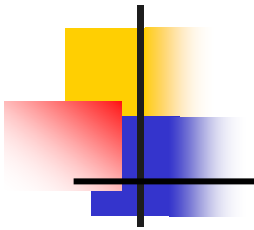
# Bean Tags

- The list of tags includes:
  - cookie, define, header, include, message, page, parameter, resource, size, struts, write
- I have found these the most widely used in my applications:
  - define – defines a new scripting variable
  - message – internationalized messages
  - write – render value of bean to JspWriter



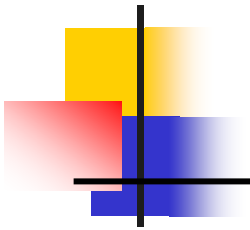
# HTML Tags

- General:
  - base, frame, html, img, link, rewrite, xhtml
- Forms:
  - button, cancel, checkbox, file, form, hidden, image, multibox, option, options, optionsCollection, password, radio, reset, select, submit, text, textarea
- Special:
  - errors, javascript, messages



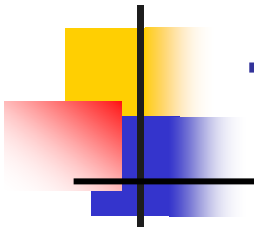
# Logic Tags

- It is recommended you use JSTL if available.
- The list includes:
  - empty, equal, forward, greaterEqual, greaterThan, iterate, lessEqual, lessThan, match, messagesNotPresent, messagesPresent, notEmpty, notEqual, notMatch, notPresent, present, redirect
- I find these most useful:
  - empty, equal, iterate, present



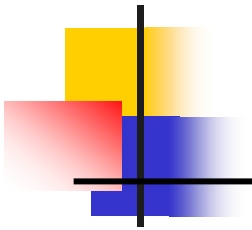
# Nested Tags

- Since I do not use the Nested Tags extension, I will leave this as an exercise to the reader.
  - I will admit the monkey Struts example is a very neat example, although I haven't met with any success at using nested tags on any applications.



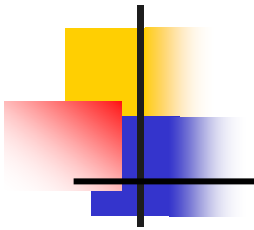
# Tiles Tags

- The list of tags include:
  - add, definition, get, getAsString, importAttribute, initComponentDefinitions, insert, put, putList, useAttribute
- I have found these most useful:
  - getAsString, insert, and useAttribute



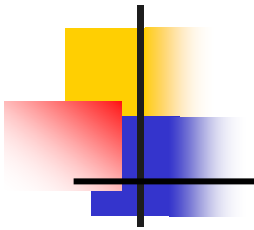
# MessageResources

- 4 base Struts classes
  - MessageResourcesFactory
  - MessageResources
  - PropertyMessageResources
  - PropertyMessageResourcesFactory
- Will be able use as common framework component outside of Struts once Jakarta Commons Sandbox project, called Resources, is moved out of sandbox.



# Message Resources

- The biggest benefit is ability to provide internationalization of text in your web application.
- Basic concept is to create a set of property files, *i.e.:*
  - messages-default.properties
  - messages-module1.properties
  - messages-default_ES.properties
  - messages-module1_FR.properties



# Messages and i18n

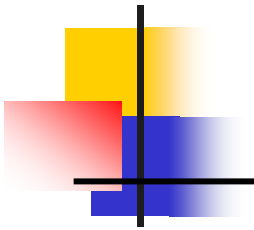
- in messages-default.properties:
  - label.hello=Hello
- in messages-default_ES.properties:
  - label.hello=Hola
- or messages-default_FR.properties:
  - label.hello=Bonjour
- *etc.*

# Message Resources

- You can create different bundles, and define them in Struts config:

```
<message-resources
    parameter="biz.threemv.config.messages-common"
    key="commonBundle"
    null="false"/>
```

  - Note: Each module can have 'default' bundle, with no key.
  - Note: Assumes package biz.threemv.config and put in WEB-INF/classes.



# Message Resources JSP Snip:

## Default bundle example:

- `<bean:message key="label.hello"/>`

## Bundle example:

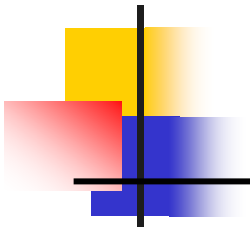
- `<bean:message key="label.hello"`
  `bundle="commonBundle"/>`

- Both will generate content based upon the User's Locale, most likely defined by browser settings.

# Message Resources, Other:

- Another great example of using MessageResources is provided in the Struts tag libs. Once this becomes a commons project, this will become even more useful.

```
/**
 * The message resources for this package.
 */
protected static MessageResources messages =
    MessageResources.getMessageResources(Globals.Package +
 ".LocalStrings");
```

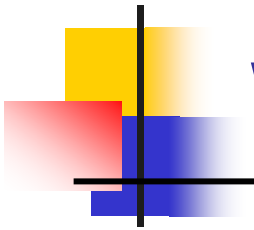


# Message Resources, Other

- Declaring this would allow:

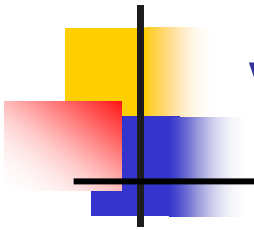
```
String errorMessage =
   messages.getMessage("getter.access", property, name);

throw new Exception(errorMessage);
```

- This is especially handy if you have plans to participate in writing open source frameworks, or if you work for an international corporation and many developers of different languages, *etc.*



# Why Use MessageResources?

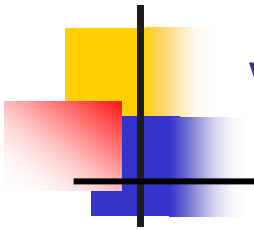
- Developing code in midwestern US, I've heard 'why use this'? Takes more time, and we don't have multi-lingual requirement.
  - Makes code easier to read. IMHO, I believe that having long plain text in middle of code (not comments) is bad. It just looks ugly. JSPs are bad enough, keep static text out.
  - Never know what the future holds. When I mentioned to my client potential for supporting Spanish, … they thought that could help with future federal mandates, *etc.*



# Validation

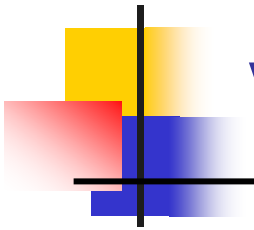
- Validation is a very powerful feature in Struts, that allows you to define validation rules for your forms.
  - Option to automatically validate input is provide in action mapping. (I've found that to be difficult to use with Tiles, though I expect that to change)
  - Programmatically decide when/where to validate, and how to handle it.
  - Works with i18n features. *i.e.*, different rules for a phone number based on country.

http://jakarta.apache.org/struts/userGuide/building_view.html#form_validation



# Validation Setup

- ValidatorForm –
  - It should be noted that ActionForm has a validate() method which allows you to provide your own validation implementation, or extend one. To use the validator, you must use ValidatorForm or a subclass in place of ActionForm.
- ValidatorPlugin –
  - You must include a link to the validator plugin in your struts-config.xml file. Module support is mostly available, but some issues exist. Check Struts documentation for your version to see published list.



# ValidatorPlugin

- The plugin setup contains essentially 2 files:

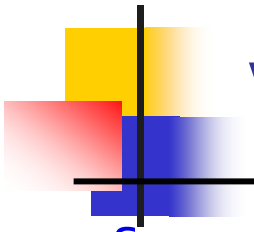
```
<plug-in className="org.apache.struts.validator.ValidatorPlugIn">

   <set-property property="pathnames"
value="/WEB-INF/my-validation.xml, /WEB-INF/validator-rules.xml" />

</plug-in>
```

  - my-validation.xml is your set of rules for your forms (declared as form-beans in struts config)
  - validator-rules is list of supported validation methods, and includes Javascript code used to generate client side validation (if used).

# validator-rules.xml example

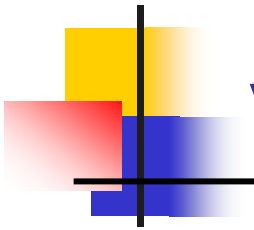
- This file can normally be left alone, but can be a useful example of how to plugin your own custom validation routines, if required.

```
<validator name="minlength"
        classname="org.apache.struts.validator.FieldChecks"
           method="validateMinLength"
     methodParams="java.lang.Object,
                   org.apache.commons.validator.ValidatorAction,
                   org.apache.commons.validator.Field,
                   org.apache.struts.action.ActionErrors,
                   javax.servlet.http.HttpServletRequest"
          depends=""
              msg="errors.minlength">

    <javascript><![CDATA[
        function validateMinLength(form) {
```



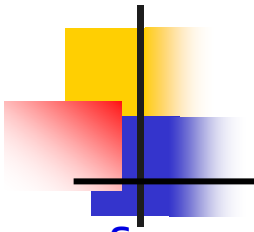
# Validation-config.xml

```
<form-validation>
   <global>
      <constant />
   </global>
   <formset>
      <constant />
      <form name="myForm">
        <field  property="firstName"
      depends="required,mask,minlength">
              <arg0 />
         </field>
      </form>
   </formset>
</form-validation>
```



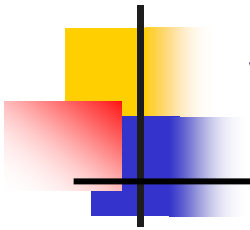
# validation depends

- The list of current validation routines includes:
  - required
  - requiredif
  - minlength and maxlength
  - mask
  - byte, short, integer, long float, and double
  - date
  - range, intRange, and floatRange
  - creditCard
  - email



# my-validation.xml

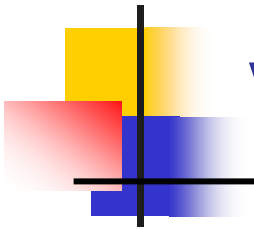
```
<formset>
   <form name="phoneEditorForm" >
       <field    property="areaCode"
            depends="required,integer,minlength,maxlength">

<arg0 key="${var:minlength}" name="minlength“
   resource="false"/>
<arg1 key="${var:maxlength}" name="maxlength"
   resource="false"/>

<var><var-name>maxlength</var-name>
     <var-value>3</var-value></var>
<var><var-name>minlength</var-name>
     <var-value>3</var-value></var>

       </field>
```



# Validation and i18n

- Validation will support i18n needs. Simply define multiple formsets. If a match against locale is not found, it will default to the formset with no locale settings. You can use language, country and variant settings that are mapped into same settings for Locale.

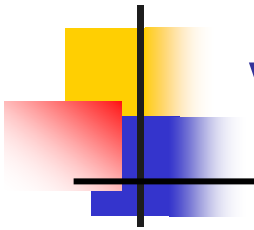
```
<formset language="fr”>
                or
<formset language="fr" country="CA">
```



# Validation Constants

- You can declare global constants to be used throughout:

```
<global>
   <constant> <constant-name>zip</constant-name>
                <constant-value>^\d{5}(-\d{4})?$</constant-value>
   </constant>
</global>

<field property="zip" depends="required,mask">
   <arg0 key="myForm.zippostal.label"/>

   <var> <var-name>mask</var-name>
        <var-value>${zip}</var-value> </var>
</field>
```



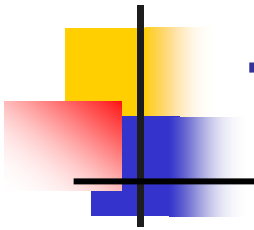
# Validation Example:

- Common jsp snip:

```
<logic:messagesPresent>
    <bean:message key="errors.header"/>
    <ul>
    <html:messages id="error">
        <li><bean:write name="error"/></li>
    </html:messages>
    </ul><hr>
</logic:messagesPresent>
```

- Common message-text.properties example:

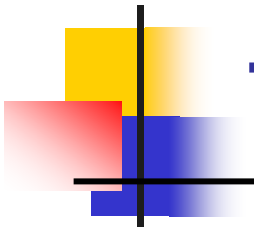
```
errors.minlength={0} can not be less than {1} characters.
errors.maxlength={0} can not be greater than {1} characters.
```



# Tiles

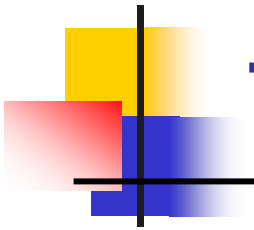
- Tiles is a very powerful templating engine that allows you to assemble presentation pages from component, called “Tiles”.

http://jakarta.apache.org/struts/userGuide/building_view.html#tiles



# Tiles Features

- Screen Definitions
- Layouts
- Dynamic Page Building
- Reuse of Tiles (components)
- Internationalization
- Multi-Channels



# Tile Screen Definitions

- Allow you to create a page of parts:
  - Header, Footer, TopNav, SideNav, and Body
- Definitions can be in:
  - One or more tiles-xxx.xml files
  - Directly in JSP pages
  - In Struts Actions
- Definitions have inheritance mechanism.

# Tiles Definition Examples

In tiles-default.xml file

A master template for use throughout the site.

```
<definition name="master" path="/main.jsp" >

  <put name="title"      value="Default Title"     />
  <put name="banner"     value="/banner.jsp"       />
  <put name="topnav"     value="/topNav.jsp"       />
  <put name="body"       value=""                  />
  <put name="formAction"  value="/nothing.do"      />
  <put name="footer"     value="/footer.jsp"       />

</definition>
```

# Tiles Definition Examples
## *(Continued)*

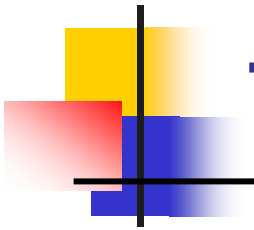
In tiles-xxx.xml file

Extend the master definition to provide page specific details.

```
<definition name="myPage" extends="master">

   <put name="title"       value="New page title" />
   <put name="body"        value="/myPage.jsp"    />
   <put name="formAction"  value="/myAction.do"   />

</definition>
```

# editorLayout.jsp Tiles Example

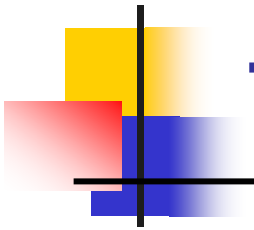
```
<%@ page language="java"%>
<%@ taglib uri="/tags/struts-tiles.tld“ prefix="tiles" %>

<tiles:useAttribute name="formAction" classname="java.lang.String" />
<html:html locale="true">
<head>
   <title><tiles:getAsString name="title" /></title>
</head>
<body>
   <tiles:insert attribute="banner“ />
   <tiles:insert attribute=“topnav" />

   <html:form  action="<%=formAction%>">
        <tiles:insert attribute=“body" flush="true"/>
   </html:form>

   <tiles:insert attribute="footer" />
</body>
</html:html>
```



# Tiles, editorLayout.jsp

## setup

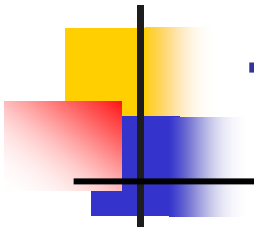
```
<%@ page language="java"%>

<%@ taglib uri="/tags/struts-tiles.tld“
           prefix="tiles" %>

<tiles:useAttribute
       name="formAction"
       classname="java.lang.String" />
```



# Tiles, editorLayout.jsp

Head section

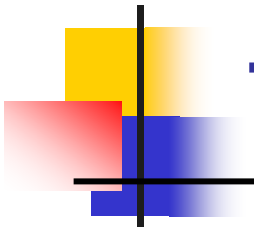
```
<html:html locale="true">
  <head>
  <title>

      <tiles:getAsString name="title" />

  </title>
…
</head>
```



# Tiles, editorLayout.jsp

Start body

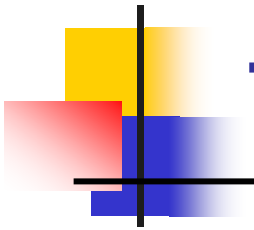
```
<body>
   <table>

        <tiles:insert attribute="banner“ />

        <tiles:insert attribute=“topnav" />
   …
```



# Tiles, editorLayout.jsp

Body cont.

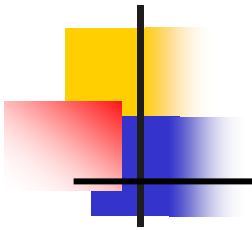
```
<html:form  action="<%=formAction%>">

  <tiles:insert attribute=“body" flush="true"/>

</html:form>
```



# Tiles, editorLayout.jsp

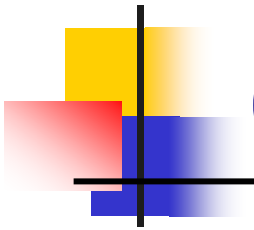
Finish body

```
    <tiles:insert attribute="footer" />

</body>

</html:html>
```



# How Do I …?

- Let's look at some code to show some solutions to common problems.
  - Checkbox Groups
  - Radio Button Groups
  - Tree Menus
  - HyperLinks
  - Drop Down Menus

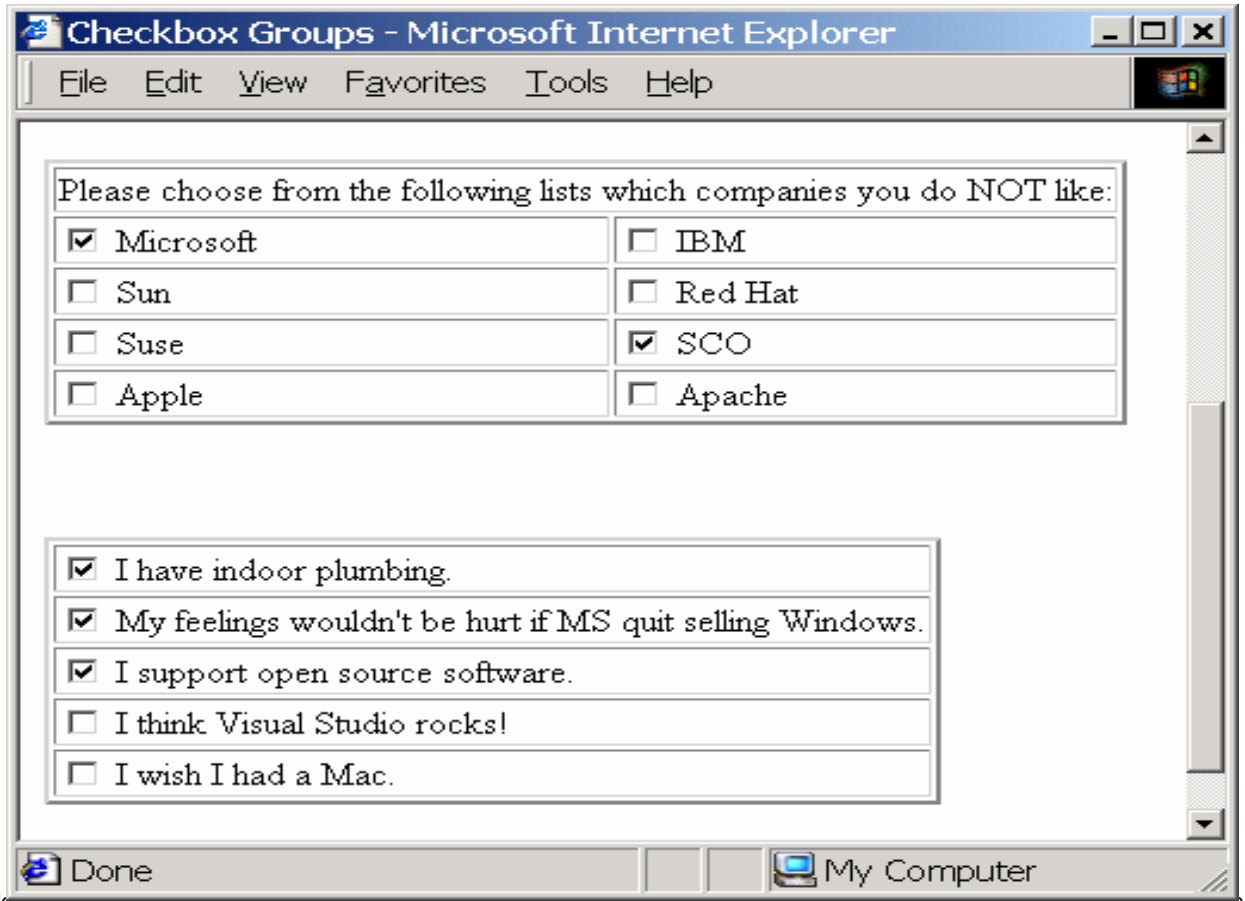


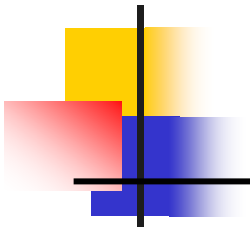
# Checkbox Groups

- There are typically 2 things you normally want to do with checkboxes.
  - Provide a list of choices to a single question. This results in a String[] concept for an answer.
  - Provide a list of questions, each having a single checkbox (yes/no) as an answer.

# Checkbox Groups Examples

- Let's take a look at how to build the following:



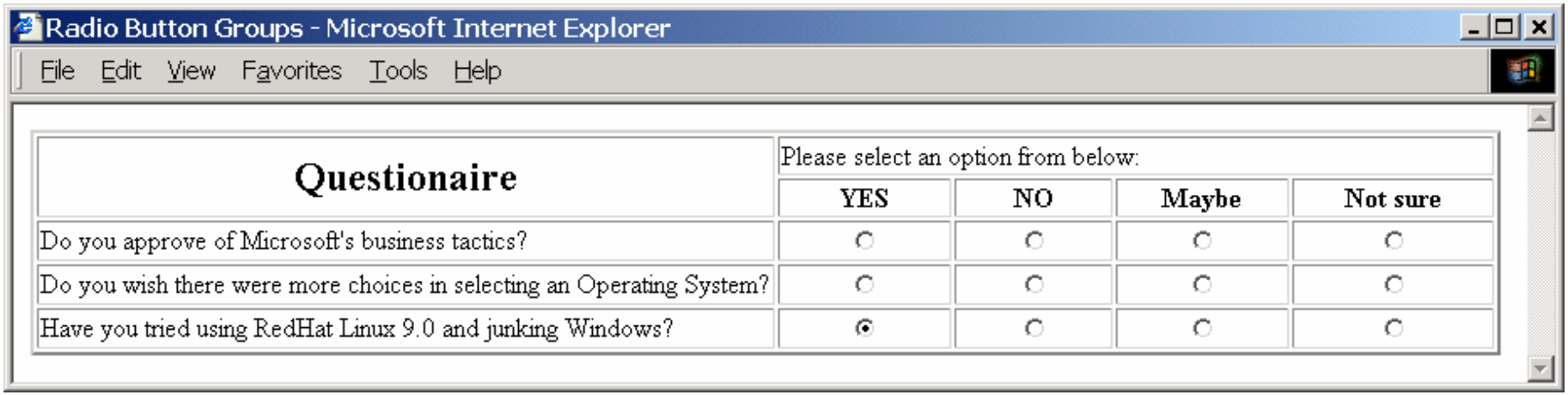


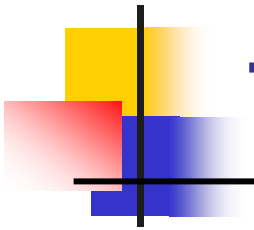
# Radio Button Groups

- Take a basic questionnaire. You have a list of questions that are database driven. You want to generate a list of questions based upon some business logic, present this list to the user, and allow them to choose an answer to each question from a standard set of radio buttons displayed in a table format.

# Radio Button Group Example

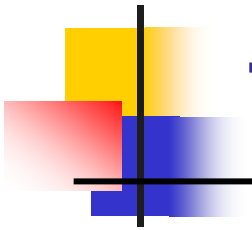
Let's take a look at how to build the following:





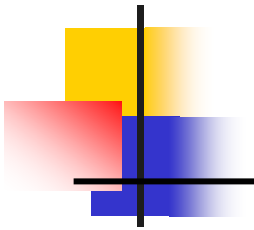
# Tab Menu

- You want to provide a custom tab menu menu for your site. You want this menu to be dynamic based upon a user's profile based on what roles they can play. You may want to provide other customizations. How can Struts help?
  - We will look at using a plugin and a custom XML configuration as a strategy.



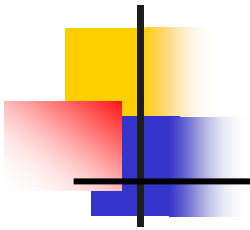
# Tab Menu Example

- Let's take a look at how to build the tab menu:



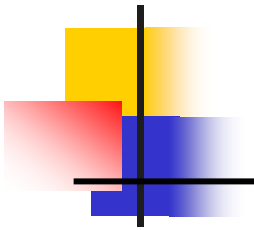
# Hyperlinks

- Provide cross module links
  - In the search module, you want to provide a [ view ] link to the person module.
- Hide a selection based on the users role.
  - You want to display a list of user profiles. If the current user is a 'manager', then you want to display a link to [ edit ] the profiles.
- Include multiple query string parameters
  - You want to build a link to edit a person's address. You need to include both the person oid and the address oid in the hyperlink.



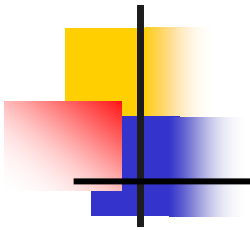
# Hyperlink Examples

- Let's take a look at how to build the hyperlinks:



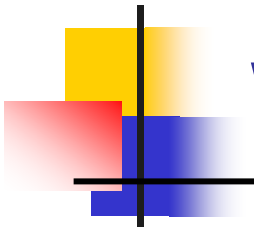
# Drop Down Menus

- You want to build a set of drop down menus / combo boxes to be used throughout your site. States, Address Types, *etc.* You want to load these *via* a set of database control tables or XML, but only once.
  - Discuss strategy of a 'MasterControl' plugin and xml solution.
- You want a Linked List box or parent/child relationship between 2 tables.
  - Discuss strategies.



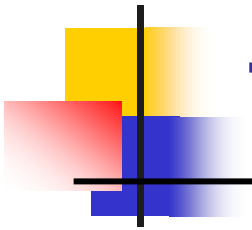
# Soccer Club Registration Demo

- Goals:
  - Use multi Struts tag library examples.
  - Provide validation.
  - Provide English and Spanish version.
- Demo:
  - Basic welcome, login, register pages.
  - Provide tabMenu over registration process.
  - Provide PDF 'receipt' of my registration.
  - Running on Jboss 3.0/Tomcat 4.1.24
  - Using Eclipse 2.1.1 as IDE



# What's Next?

- More tags and extended functionality
- JSTL and EL
- JSF
- DB support for MessageText
- Single Example App, using modules and latest features, plus best practices.



# The End.

Thank you for coming.

Gary D. Ashley Jr.

3rd Millennium Visions, Inc.

garyashley at toughguy dot net