

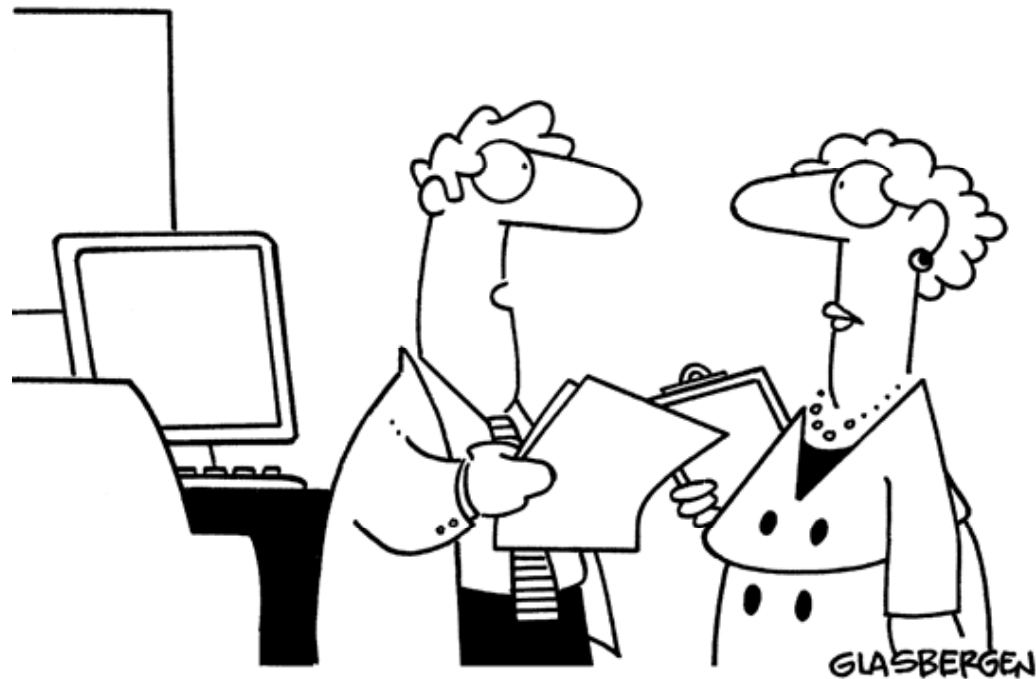
Web Applications

and

Struts 2

# Problem area

Copyright 2006 by Randy Glasbergen. [www.glasbergen.com](http://www.glasbergen.com)



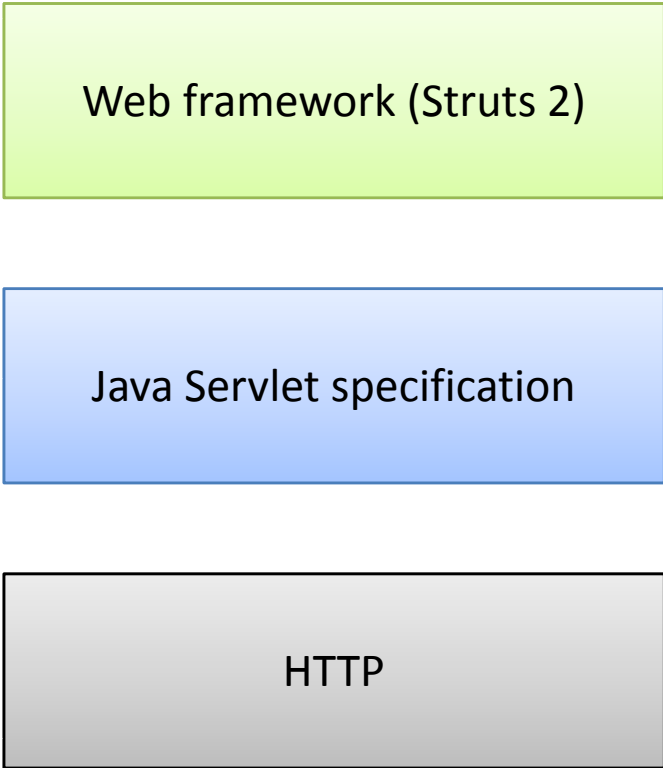
**“My team has created a very innovative solution,  
but we’re still looking for a problem to go with it.”**

# Problem area

- Separation of application logic and markup
  - Easier to change and maintain
  - Easier to re-use
  - Less error prone
- Access to functionality to solve routine needs
  - Data transfer between client (HTTP) and server (Java)
  - Validation
  - Internationalization
  - User interface components (tags)

# Web applications

- Struts 2 sits on top of two important technologies



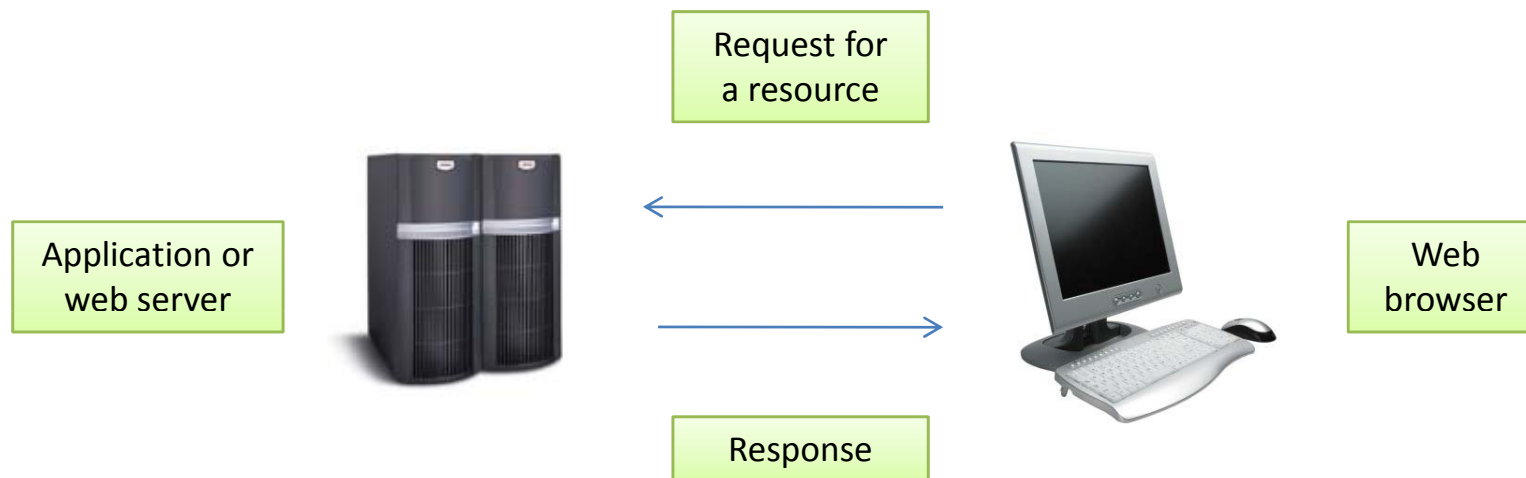
Web framework (Struts 2)

Java Servlet specification

HTTP

# Hypertext Transfer Protocol (HTTP)

- Series of client-server message exchanges
- Designed for static HTML rather than dynamic
  - Stateless (how to do things like authentication?)
  - Text based (how to map to and from Java types?)



# Java Servlet specification



- Provides intuitive object-oriented abstraction of HTTP:
- Servlet (HttpServlet)
  - Small Java program
  - Receive and respond to requests from Web clients
- Request (HttpRequest)
  - Object representing a client request
  - Access to parameters, request URL, input stream...
- Response (HttpResponse)
  - Object representing a server response
  - Access to content type, header, writer...

# Java Servlet specification



- Session
  - Provides a session between multiple requests
  - Usually corresponds to a user
  - Allows Servlets to bind objects and manipulate information
- Filter
  - Performs filtering on request sand/or responses
  - Configured in web.xml
  - Useful for authentication, logging...

# Web application archive



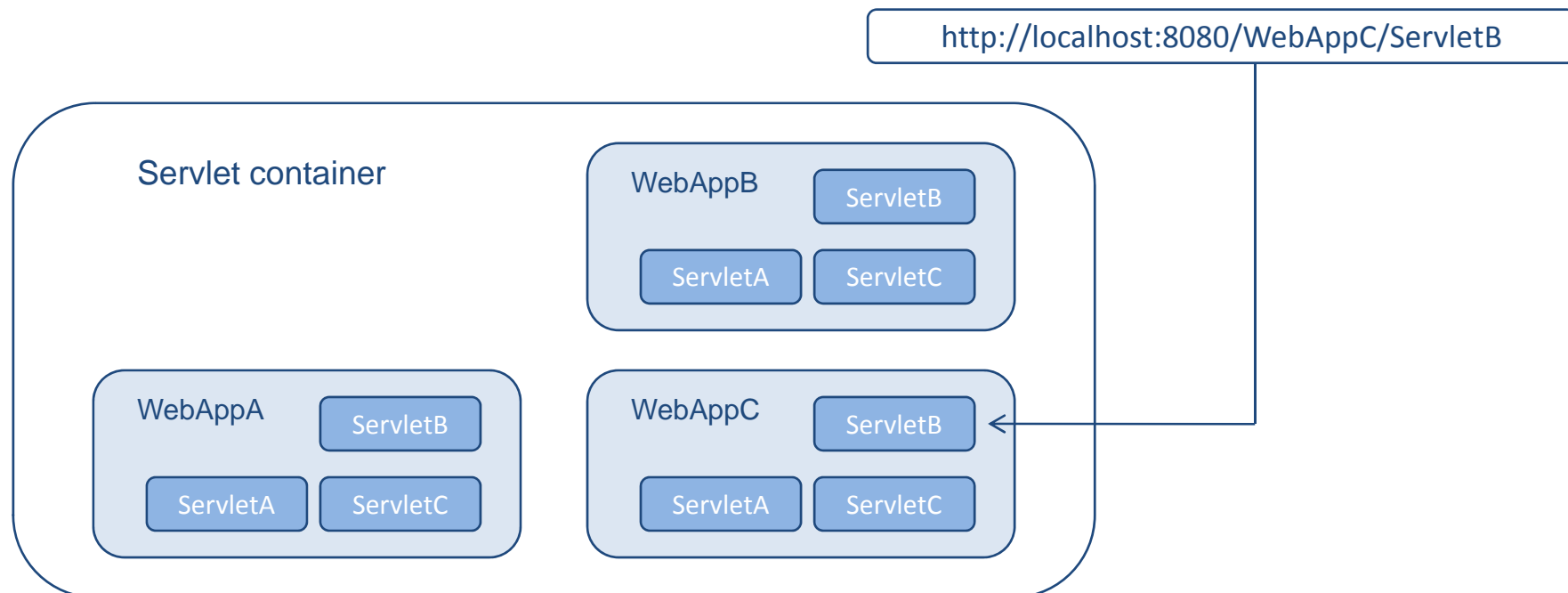
- Servlets are packaged into web applications as *WAR-files*
  - Zip file with a specific structure
- Contains:
  - Servlets
  - Java classes
  - Web resources (HTML, CSS, templates, images)
  - Dependent libraries (JARs)

/	Web resources (HTML, CSS, templates, images)
/WEB-INF	Private files (not returned by Web server)
/WEB-INF/web.xml	Deployment descriptor / configuration file
/WEB-INF/classes	Java classes
/WEB-INF/lib	Dependent libraries / JARs



# Servlet container

- Web applications are deployed in *Servlet containers*
  - Implementation of the Servlet specification
  - Web server able to run and manage the life-cycle of servlets
  - Popular open-source versions are *Tomcat* and *Jetty*



# Purpose of Web frameworks

- Solving application level concerns / routine work!
  - Binding request parameters to Java types
  - Validating data
  - Providing internationalization
  - Rendering presentation layer (HTML etc)
  - Making calls to business layer



# Action classes

- Purposes
  - Encapsulates the work to be done for a request
  - Serve as data carrier in transfer from request to view
  - Determine which result should render the view

```
public class InvertStringAction implements Action
{
    private String word;
    // set-method

    private String invertedWord;
    // get-method

    public String execute()
    {
        if ( word == null || word.trim().length() == 0 )
        {
            return INPUT;
        }

        invertedWord = getInvertedWord(); // implemented another place

        return SUCCESS;
    }
}
```

# Action classes

- ActionSupport: Abstract convenience class providing:
  - Validation
  - Internationalization
- Works together with interceptors in the default stack
- ActionContext: Container holding relevant data for a request:
  - ValueStack
  - Request parameters
  - Session variables
  - Application attributes

# ModelDriven Actions

- Transfers data directly onto model objects
  - Avoids tedious object creation code
- Provided by interface *ModelDriven*<T>
  - Exposes method *T getModel()*

# ModelDriven Actions

Class extending `ActionSupport` and implementing `ModelDriven`

Student instance created

`getModel` implemented

Student object populated with data and ready to use

Don't change object reference!

Data will be available in view

```
public class SaveStudent
    extends ActionSupport implements ModelDriven<Student>
{
    private Student student = new Student();

    public Student getModel()
    {
        return student;
    }

    private StudentService studentService = // retrieved somehow

    public String execute()
    {
        studentService.saveStudent( student );

        return SUCCESS;
    }
}
```

# Validation

- Basic validation accessible from `ActionContext`
- The *DefaultWorkflowInterceptor* works behind the scenes
- Separates validation logic from business logic



# Validation

Class extending `ActionSupport`

Java bean properties

Validate method containing logic for checking validity of data

Storing errors via methods provided by `ActionSupport`

Validation workflow separated from execution

Execute method containing logic

```
public class CalculateAction extends ActionSupport
{
    private Integer numerator;
    private Integer denominator;
    // set-methods

    private Double result;
    // get-method

    public void validate()
    {
        if ( denominator == null )
        {
            addFieldError( "denominator", "Please enter a numerator" );
        }
        elseif ( denominator == 0 )
        {
            addFieldError( "denominator", "Divison by zero not allowed" );
        }
    }

    public String execute()
    {
        result = // calculate somehow
    }
}
```

# Validation

Input result is used when validation errors exist

```
<action name="calculate" class="no.uio.inf5750.example.action.CalculateAction">  
  <result name="success" type="velocity">calculate.vm</result>  
  <result name="input" type="velocity">calculate.vm</result>  
</action>
```

Validation error messages printed for appropriate UI tag in view

```
<html>  
<body>  
  
#sform( "action=calculate" )  
  #stextfield( "label=Numerator" "name=**numerator**" )  
  #stextfield( "label=Denominator" "name=**denominator**" )  
  #ssubmit( "value=Calculate" )  
#end  
  
</body>  
</html>
```

# Internationalization

- Separation layer between source code and messages
- Built upon Java resource bundles, which will be found in:
  - Same package and with same name as Action class
  - Classpath after setting *struts.custom.i18n.resources* config property
- Messages accessible in Java code through *getText(..)*
- Locale accessible in Java code through *getLocale()*
- Messages accessible in template through *key* and *name* properties of UI tags



# Internationalization

struts.xml configuration

```
<constant name="struts.custom.i18n.resources" value="i18n.i18n" />
```

i18n.properties in i18n folder on classpath

```
salute=Bonjour!
```

Action class using `getText(..)` and `getLocale()`

```
Public class SaluteAction extends ActionSupport
{
    public String execute()
    {
        String salute = getText( "salute" );
        Locale locale = getLocale();
        return SUCCESS;
    }
}
```

Velocity template using text tag

```
<h3>#stext( "name=salute" )</h3>
```

# UI component tags

- High level tag API implemented in:
  - JSP
  - Velocity (used in this lecture)
  - Freemarker
- Data tags
  - Moves data between the ValueStack and the view
- Control tags
  - Alters the rendering flow of the view
- UI tags
  - Provides advanced HTML form controls

# Spring integration

- Useful to apply *dependency injection* to a Struts 2 application
  - *Struts 2 Spring plugin* is required (struts2-spring-plugin)
  - Spring must be started in web.xml with *ContextLoaderListener*
- Approach 1: Define dependencies and action classes in Spring
  - *Class* attribute in Struts 2 config will point at Spring bean identifiers
- Approach 2: Use auto-wiring by name
  - Struts 2 will instantiate Actions but let Spring inject beans afterwards
  - Bean identifiers are matched to set-methods in Actions

# Auto-wiring dependencies by name

Only the StudentService bean is defined in the configuration

```
<bean id="studentService" class="no.uio.inf5750.model.DefaultStudentService"/>
```

Action class provides StudentService property and set-method

Name of set-method is matched with Spring bean identifiers

```
public class SaveStudentAction extends ActionSupport
{
    private StudentService studentService;

    public void setStudentService( StudentService studentService )
    {
        this.studentService = studentService;
    }

    // Student properties and setters

    public String execute()
    {
        studentService.saveStudent( student );
    }
}
```

# Summary

- Struts 2 is built upon the Java Servlet specification which is built upon HTTP
- Struts 2 solves application level routine work not provided by the Servlet specification
  - Binding request params to Java types
  - Validating data
  - Providing internationalization
  - Rendering presentation layer
  - Making calls to business layer



# Resources

- Brown, Davis, Stanlick: *Struts 2 in Action*
- Velocity user guide:
  - <http://velocity.apache.org/engine/devel/user-guide.html>
- Struts home page:
  - <http://struts.apache.org>
- Example code on course homepage