Struts Tutorial

by Stephan Wiesner

Struts Tutorial

by Stephan Wiesner
Copyright © 2002 Stephan Wiesner

This is a tutorial for the Struts framework. It consists of a small application which is developed step by step. It can display books in the browser, new books can be created and their state saved/loaded in xml files. I try to do the whole thing step by step, with lots of screenshots to show that it really does work and to help you along. Many of the screenhots actually show error messages. This will help you avoid some of the many pitfalls along the way. This work should enable you to work with Struts in a short time, but you will not become a Struts guru with it. You should have the official documentation at hand to read the theoretical background. This work is *closed*, I don't think I will continue it's development.

Table of Contents

List of Figures

Chapter 1. Introduction

Abstract

Why I wrote the tutorial, who is the target group and an overview of the contents.

To get an idea what this tutorial is about and whether you should be able to learn something new.
Annotations [*add annotation*]

1.1. Introduction

During this tutorial I will develop a little application, step by step, that implements Struts. You should have some experience with JavaServerpages, JSP and XML and you should have a server implemented that can run those things. I use *Struts Version 1.0.2* and *Tomcat 4.1.12*; see my homepage for a tutorial about installing Tomcat 3.x under MS Windows, if you don't have it yet. Please, I keep getting mails (from India mostly), asking me to explain the setup in *websphere*. I don't have that server and don't intend to get one, so I can't give any specific help for it.

The tutorial will start with the installation of the Struts framework itself, will implement some minimalistic features in pure HTML and will then add Struts funtionality, giving hints and thoughts about how to organize your code. Finally I'll even show you some tricks with AspectJ to make your life as a programmer somewhat easier.

About the annotations: From time to time I decide to publish mails send to me about this tutorial. They are marked as 'Annotation' or 'Remark' (currently DocBook has no elements for annotations, though this feature is under discussion) and are placed at the bottom of the chapter they best fit into. I started this after I already had lots of mails, so many of them are not included (I'm just too lazy, I guess :-). I will continue to do this, so if you don't want that (I just post the name, not the mail address), please state it in your mail. And, please let me know from which country you come. I'm curious about that . . . Text marked with '[]' is not part of the original mail but an addition from me (typical [...]).

Also take a look at the Chapter 15, *Epilogue*. chapter about technical backgrounds. It might interest you.

Important for *Tomcat 4.x users: Some code will not work!*. Tomcat 4.x doesn't allow you to work without a package. At the moment, I am going through it all and update it, but it I might have overlooked somethin.
Annotations [*add annotation*]

Chapter 2. Installation of Struts

This one is quite easy. I take it, you have a running Tomcat 3.2 (or comparable server) and know how to handle it? I will not go into detail but all steps necessary to make my examples work are explained.
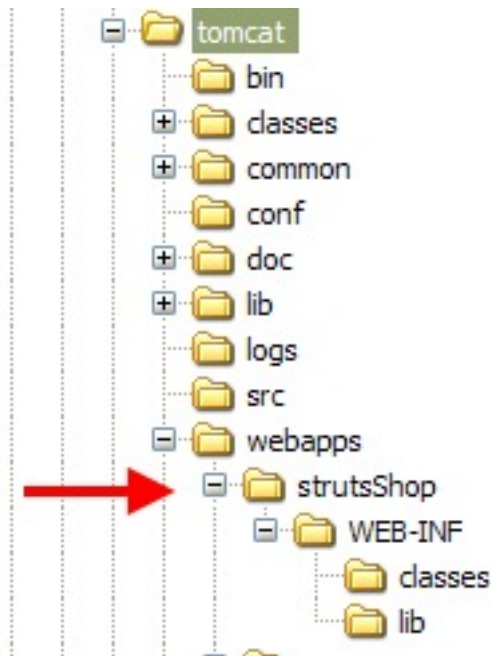
Figure 2.1. The folder structure

Generate a directory structure like in the screenshot (adapt if you don't use Tomcat)
Download the newest version of Struts (I use 1.0.2)[1].
Extract the download.
Copy the Struts.jar
Paste it to the lib directory of your application (see above). DON'T, I repeat *DON'T just add it to your CLASSPATH!* I did that and it worked fine. Until I restartet the Tomcat, that is. Took me quite a while to figure the connection out. At least with my system there was a big problem, so I recommend to add it just to this application.
That is all, for the moment. Don't delete the rest of the Struts download, though. We will need more later on.
Annotations [*add annotation*]

---

[1]Struts is developing fast, so you if at the time of your reading of this text there is a new version, take my. You can switch to the new one after you worked through the examples. This might save you lots of pain.

Chapter 3. The First Try: Simple JSP

We will start with a simple JSP page, just to ensure that we did everything right until now. Create a file called BookView.jsp in the 'strutsShop'-directory.

```
First JSP

    <%@ page language="java" %>

    <html>
    <head>
    <title>Struts Tutorial: BookView</title>
    </head>

    <body bgcolor="white">
    <h2>BookView</h2>

    </body>
    </html>
```

Adresse | http://127.0.0.1/strutsShop/BookView.jsp

BookView

Figure 3.1. Our First Page

Not very impressive, I admit, but we will start improving it right away.
Annotations [*add annotation*]

Chapter 4. Second Try: Struts For Internationalisation

We will use Struts to display predefined texts in the default language of the user. Well, not for every user, I suppose... This will require some closer look at the functionality of Struts.

Display the title of the page in different languages.

We starts with some basics about Struts: Go to the directory you extracted the Struts download to and copy some files to your WEB-INF directory like in the screenshot (you should browse through a few of the examples that come with Struts, while searching for them).
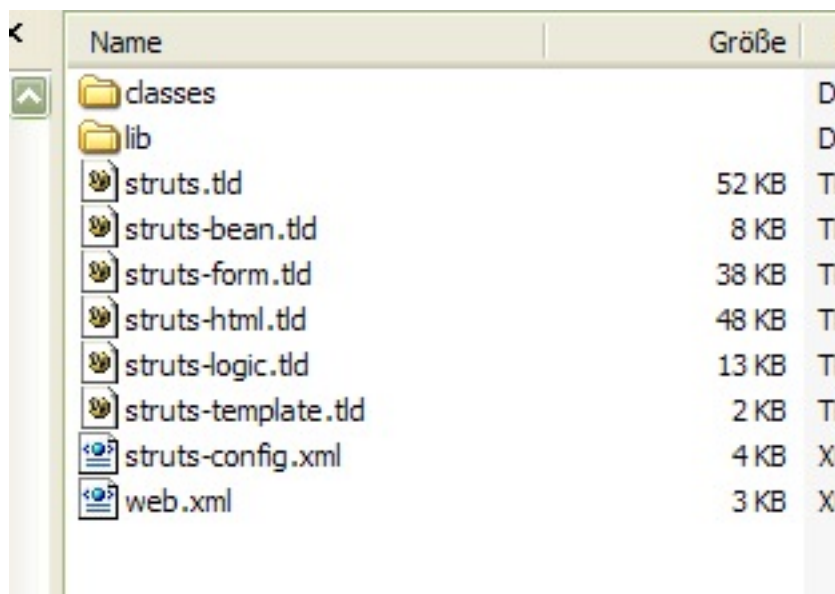
Figure 4.1. Needed LTD files

Take the time to look into the struts-html.tld file. Don't panic, though, we will not adapt those files! They are there to help us and define the functionality of Struts. You can find the parameters you need for using a Strut tag in there. You could look into the official docu and would probably be better off, though.

Our web.xml must define an action Servlet and the links to the ltd-files. Just copy one web.xml from the Struts sources and remove everything else.

```
Empty struts-config.xml

<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">
<struts-config></struts-config>
```

```
web.xml

<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
  PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
  "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

<web-app>

<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
  <init-param>
    <param-name>application</param-name>
    <param-value>ApplicationResources</param-value>
  </init-param>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
      <param-name>debug</param-name>
      <param-value>2</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>2</param-value>
    </init-param>
    <init-param>
      <param-name>validate</param-name>
      <param-value>true</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
</servlet>


  <!-- Standard Action Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
     <servlet-name>invoker</servlet-name>
     <url-pattern>/servlet/*</url-pattern>
  </servlet-mapping>

  <!-- The Welcome File List -->
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>

  <!-- Struts Tag Library Descriptor -->
<taglib>
    <taglib-uri>
        /WEB-INF/struts-bean.tld
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-bean.tld
    </taglib-location>
</taglib>
<taglib>
    <taglib-uri>
        /WEB-INF/struts-html.tld
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-html.tld
    </taglib-location>
</taglib>
<taglib>
    <taglib-uri>
        /WEB-INF/struts-logic.tld
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-logic.tld
    </taglib-location>
</taglib>
<taglib>
    <taglib-uri>
        /WEB-INF/struts-template.tld
    </taglib-uri>
    <taglib-location>
        /WEB-INF/struts-template.tld
```

```
        </taglib-location>
    </taglib>

    </web-app>
```

Okay, lets start with something really easy. We will display the title of the page in different languages, depending on the preference of the user.

Create a file called 'ApplicationResources.properties' in the classes dirctory.
Open it and enter the line: 'index.title=Struts Tutorial' into it
Create another file called 'ApplicationResources_de.properties' (de for Deutschland (Germany)) and enter 'index.title=Struts Einführung'
Edit the BookView.jsp and change the head to:

```
    BookView.jsp: Introducing Internationalization

<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld"  prefix="bean" %> ❶

<%@ taglib uri="/WEB-INF/struts-html.tld"  prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html:html locale="true">
<head>
<html:base/>
<title>
  <bean:message key="index.title"/> ❷

</title>
</head>
<body>
    <h2>BookView</h2>
</body>
</html:html>❸
```
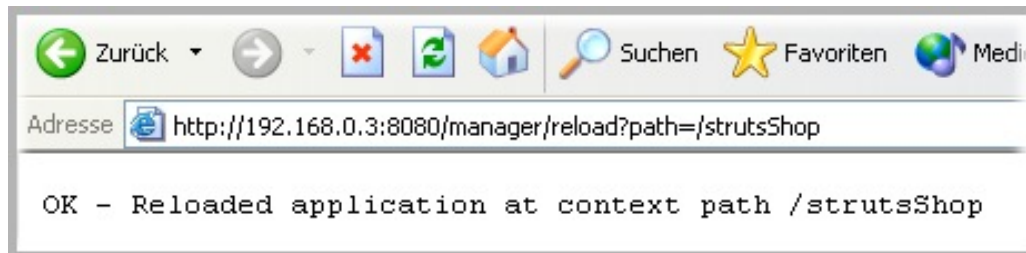
❶   Import the needed TLD files.
❷   Here is our language dependend value.
❸   Don't forget to close this tag in the right format.

Figure 4.2. Displaying the German title element

Now you have to reload the strutsShop context, otherwise the propertie files

will not be evaluated.

Figure 4.3. Reloading a Context With Tomcat 4.x

You have to do this everytime you change something in the language files.

Note

For Tomcat 3.x, you have to restart the server, as it doesn't have the manager functionallity.

What exactly you see depends on the default language on your coumputer. I see the German title 'Struts Einführung'. Change your default language and relaod the page. Set it first to english, then to German and the title should adapt accordingly.

Note

As you just saw it is very easy to display texts in different languages. What you will realize very quickly, though, is how tedious it can get to enter/check/change everything in different languages, but that has nothing to do with the way it is implemented in Struts. I have made the experience that it is necessary to implement every Text using the variables and that it is best to use just one language for the development. Then, before making a release, I adapt the files for the other languages. This way I save a lot of double work.

Annotations [*add annotation*]

Annotations

Jochen Tuchbreiter gave me the hint that instead of restarting the server, you can also call 'http://127.0.0.1/manager/reload?path=strutsShop'. This works in Tomcat 4.x only!

Ed Trembicki-Guy: The first tip from the end of chapter 4 about reloading a web app doesn't always work. I found that when I modify web.xml I must restart Tomcat. Just reloading the web app does not apply my changes to web.xml

Stephan Wiesner: With Tomcat 4.1.12 I didn`t even have to reload the context. The web.xml got parsed right away. The struts-config.xml, however still needs a restart of the server (at least often, not always .-().

Vijay Khanna from India discovered that the empty Struts-Config.xml must contain at least one root element (like any XML file). This is fixed in this version.

Thanks Vijay.

Neal had the following problem [see next remark for solution]: When I tried "Struts For Internationalisation" got this error on Tomcat console... `=====error on Tomcat console...========= Starting service Internal Services Java Web Services Developer Pack/1.0-ea2 Starting service Java Web Services Developer Pack Java Web Services Developer Pack/1.0-ea2 org.xml.sax.SAXParseException: The processing instruction target matching "[xX][mM][lL]" is not allowed. at org.apache.xerces.util.ErrorHandlerWrapper.createSAXParseEx ception(Er rorHandlerWrapper.java:232) . . . . . org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:2 03) [ERROR] Digester - -Parse Fatal Error at line 2 column 6: The processing instruction target matching "[xX][mM][lL]" is not allowed. <org.xml.sax.SAXParseException: The processing instruction target matching "[xX][mM][lL]" is not allowed.>*

Neal: Hi Stephan, web.xml had a parse error per the log file. So I had to delete all empty linesin web.xml...and then it worked.. I copied-and pasted web.xml directly from yr siteto my PFE editor(maybe the editor is the culprit..dunno..maybe it put invisible characters for "next line") Perhaps a warning msg alongside web.xml at the site would be appropriate(a la suggestion:). So the error I mailed you originally happens when there is a problem with an xml file.

Chopin Yen: "I have tried both tomcat 3 and 4 and have gotten a different error messages. I am getting MESSAGE not found under Tomcat 3 and the following from tomcat 4 which is my primary interest:[...] No getter method for property title of bean org.apache.struts.taglib.html.BEAN at org.apache.jasper.runtime.PageContextImpl.handlePageException(PageContextImpl.java:463)" The solution came a few mails later: "I think I know what happened: I was using struts 1.1 beta version... I am using 1.0.2 now. It seems to work now."

Ty Connel: Answered to the remark above: "I had a similar / same problem. Using the 4.1.7 version of tomcat and Struts 1.1b. To get it to work for me, I had to copy the following files to my applications WEB-INF/lib directory: commons-beanutils.jar commons-collections.jar commons-dbcp.jar commons-digester.jar commons-logging.jar commons-pool.jar commons-services.jar commons-validator.jar struts.jar xerces.jar"

Vivien: "javax.servlet.ServletException: Missing message for key index.title" The index.title is defined in the ApplicationResources.properties files. This error occures when you are missing the entry 'index.title' in the file or when you are missiong the propertie file itself (maybe none for your language, if you are using neither German nor English as your system language).

Chapter 5. Struts for Forms

Abstract

This chapter will finally allow us to do some real programming (about time, I know). We will create a simple Bean (Book.java) and two JSP pages. One to create a new book and a second to display it. For that we will use Struts. Further, we will take a first look at the struts-config.xml to configure our application.

Goal: To understand how Struts can help us with standard behaviour concerning forms.
Annotations [*add annotation*]

5.1. First Try at Forms

Let's start with the Bean. Create a file Book.java [2] in your classes directory and enter the following:

```
    Book.java

package books;
import java.util.Vector;

/*
    A simple book.
    @author stephan@stephanwiesner.de
*/
public class Book
{
    /** The title */
    private String title = "";
    /** We can have more than one author */
    private Vector authors = new Vector();
    /** The number of pages the book has */
    private int pages = 0;


    /** Standard constructor. */
    public Book()
    { }

    /** @param title The new Title   */
    public void setTitle(String title)
    {   this.title = title;    }

    /** @return The title. */
    public String getTitle()
    { return this.title; }


    /** @param pages The new number of pages. */
    public void setPages(int pages)
    { this.pages = pages; }


    /** @return The number of pages. */
    public int getPages()
    { return this.pages; }

    /**
        We don't want to work with the Vector here, as it is
        only a reference we would get!
        @param author Add another author
    */
    public void addAuthor(String author)
    { this.authors.add(author); }
```

[2] This class, as all classes (in this tutorial/ my work/ the world), is not perfect. I keep getting mails telling me that I should use an Array or a List instead of the Vector or that I'm lazy because I use * for the imports. Well, the internet is a free world and you can try to lecture me and I will even think about it. Most of the time we are talking oppinions, though, not hard, proved facts. So, don't expect me to heed all your advices ;-)

```
    /**
        Pay attention not to use the wrong number.
        @param position The number of the author to remove.
    */
    public void removeAuthor(int position)
    { this.authors.remove(position); }

    /** @return The number of authors the book has. */
    public int getNumberOfAuthors()
    { return this.authors.size(); }
}
```

This is just a tiny Bean that shouldn't need any explanations.

Now, on to the form. We need a simple HTML form that can create a book. We will use three text fields for title, author (only one for the start) and number of pages. We'll need some preparations for that. You have to pay close attention for this, as it is quite complicated for a beginner (my personal opinion).

Edit your BookView.jsp to include the following (the import belongs to the top of the JSP):

```
<%@ page import="java.util.*, books.*" %>

<html:form action="createBook" method="GET">
    Title:<html:text property="title" /> <br/>
    <html:submit property="submit"/>
</html:form>
```

Note

The usage of a package and the corresponding import statement are new for Tomcat 4.x. This was not included in earlier versions of this tutorial and lead to quite some discussions and tons of mails. Well, as you can see, I finally included it .-)

Then try it out. You will get something like in the following Screenshot. What happened? We said something like "Struts, use this form to set the value of the property named 'title' of the Bean."

One Moment! Of what Bean? We wanted to use our Book.java, of course, but Struts doesn't know that! We have to define an Action and that has to know which Bean to work with.

dresse 🔊 http://127.0.0.1/strutsShop/BookView.jsp

Error: 500

Location: /strutsShop/BookView.jsp

Internal Servlet Error:

```
javax.servlet.ServletException: Cannot find ActionMappings or ActionFormBeans collection
        at org.apache.jasper.runtime.PageContextImpl.handlePageException(PageContextImpl
        at _0002fBookView_0002ejspBookView_jsp_6._jspService(_0002fBookView_0002ejspBook'
        at org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:119)
```

Figure 5.1. Nice Errormessages are always close at hand

We need a second JSP now. Create *CreateBook.jsp*

```
    CreateBook.jsp

<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld"  prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld"  prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html:html locale="true">
<head>
    <html:base/>
    <title><bean:message key="index.title"/></title>
</head>

<body bgcolor="white">
<h2>Create a book</h2>


<html:errors/>

<html:form action="createBook.do" method="GET">
    Title:<html:text property="title" /> <br/>
    <html:submit property="submit"/>
</html:form>

</body>
</html:html>
```

The Action-Class: Create a file called BookAction.java in your classes direc-
tory and enter:

```
    BookAction.java
import javax.servlet.http.*;
import org.apache.struts.action.*;


/*
  The action for the creation of a book.
  @author stephan@stephanwiesner.de
*/
public final class BookAction extends Action
{
  /**
    @param mapping The ActionMapping used to select this instance
    @param form The optional ActionForm bean for this request (if any)
    @param req The non-HTTP request we are processing
    @param res The non-HTTP response we are creating
    @return Return an ActionForward instance describing where and how
        control should be forwarded, or null if the response has already
        been completed.
  */
  public ActionForward perform(ActionMapping mapping,
      ActionForm form,  HttpServletRequest req,
      HttpServletResponse res)
  {
      System.out.println("Start perform(" + form + ") . . ." );
      String title = req.getParameter("title");
      Book book = new Book();
      book.setTitle( title );
      System.out.println("After creation of book: " + book.getTitle() );

      req.setAttribute("BOOK", book); ❶
      return mapping.findForward("bookCreated");
  }
}
```

❶   Insert the book into the request for further processing. It can later be ac-
cessed with request.get("BOOK").

Note

This class is NOT using Struts like it is meant to be (yet)! What it does is to create a book and give it the title submitted from the form. The code uses HttpServletRequest-getParameter(). That is not magic, you can do that more easily without Struts. A better way is explained in Section 5.3, "A better way to seperate Book and BookForm"

Edit you struts-config.xml to the following:

```
struts-config.xml including forward and mapping

<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">


<struts-config>
  <form-beans>
    <form-bean name="bookForm" type="books.Book"/>❶

  </form-beans>

  <global-forwards>
    <forward   name="bookCreated"  path="/BookView.jsp"/>

  </global-forwards>

  <action-mappings>
    <action    path="/createBook"❷

               type="books.BookAction"
               name="bookForm"❸

               scope="request"❹

               input="/CreateBook.jsp">❺

    </action>
  </action-mappings>

</struts-config>
```

❶   Map the name 'bookCreated to the class 'Book(.class)'.
❸   'bookForm' is mapped to 'book.class' above (❶).
❹   How long should the values be saved. Another typical value would be 'session'.
❺   In case of an error, jump tp this page.
❷   Don't forget the package name.

We told the Struts framework to establish a connection between the name *'book-Form'* and the class *'Book'*. Further, we created the forwarding shortcut called *'bookCreated'*, which will point to BookView.jsp. And, finally, we defined what our form with the attribute *'action="createBook.do"'* is about to do: Take the Bean associated with 'bookForm' which will get its input from 'CreateBook.jsp' and create that Bean on the command 'createBook'.

Try it out! Compile your classes. Depending on what you did with the Struts.jar, you might get tons of errors. I take a very simple approach for that. I have a local copy of the *JAR file in my classes directory* and include it in the javac command.

Figure 5.2. Using Struts.jar locally
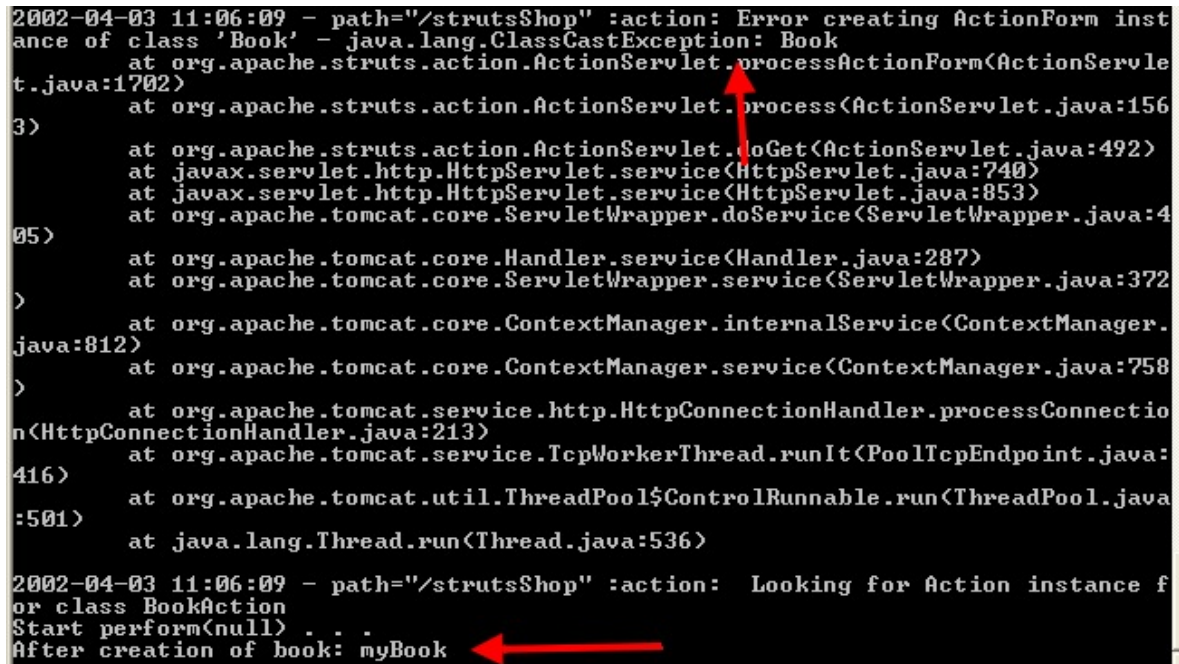
Note

Mind that this is just for compilation. It has nothing to do with the use of the JAR file in Tomcat itself! It must reside somewhere the server can find it.

Restart your context (you have to restart it every time you change one of the config files, which can become very annoying). Now load the CreateBook.jsp into your Browser. It should display a single text field, asking you to enter a title. Well, don't hesitate, do it! The result is still very dissapointing, I'm afraid. We get the same page again. No books seems to be created.

Figure 5.3. Where is the book?

```
2002-04-03 11:06:09 - path="/strutsShop" :action: Error creating ActionForm inst
ance of class 'Book' - java.lang.ClassCastException: Book
        at org.apache.struts.action.ActionServlet.processActionForm(ActionServle
t.java:1702)
        at org.apache.struts.action.ActionServlet.process(ActionServlet.java:156
3)
        at org.apache.struts.action.ActionServlet.doGet(ActionServlet.java:492)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:740)
        at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
        at org.apache.tomcat.core.ServletWrapper.doService(ServletWrapper.java:4
05)
        at org.apache.tomcat.core.Handler.service(Handler.java:287)
        at org.apache.tomcat.core.ServletWrapper.service(ServletWrapper.java:372
)
        at org.apache.tomcat.core.ContextManager.internalService(ContextManager.
java:812)
        at org.apache.tomcat.core.ContextManager.service(ContextManager.java:758
)
        at org.apache.tomcat.service.http.HttpConnectionHandler.processConnectio
n(HttpConnectionHandler.java:213)
        at org.apache.tomcat.service.TcpWorkerThread.runIt(PoolTcpEndpoint.java:
416)
        at org.apache.tomcat.util.ThreadPool$ControlRunnable.run(ThreadPool.java
:501)
        at java.lang.Thread.run(Thread.java:536)

2002-04-03 11:06:09 - path="/strutsShop" :action:  Looking for Action instance f
or class BookAction
Start perform(null) . . .
After creation of book: myBook
```

Figure 5.4. . . . here is one.

The output from Tomcat (probably depends on your logging level defined in your server.xml) proves that we did create a book. But what about the long error message above? It says that there was a 'Error creating ActionForm instance of class 'Book' - java.lang.ClassCastException: Book'. We will come to that in the next chapter.

> **Question: Looked at the API?**
>
> **Exercise: If you have not looked at the API from Struts yet, do it. Take a look at the classes Action (which we already used) and ActionFormular.**

Annotations [*add annotation*]

What did we learn?

We created a bean for storing the properties of a book. Next we defined an Action (we still don't really know what that is about, only that we seem to need one) and used a formular (with it's attributes defined in struts-config.xml) to create a book. Only, we don't seem to have it done quite right.

5.2. Struts: Introducing the ActionForm

Abstract

This chapter will present some reasoning about MVC and will show how to implement the ActionForm.

Goal: Our example from the prevoius chapter wasn't satisfying, yet. We will introduce an ActionForm in order to use the might of Struts in it's complete glory.

The idea of using forms with Struts is to enable them to handle the creation/update of Beans. This means for our example that we want a form that enables us to enter all needed information for a valid book. Then we want some checking to happen (like not creating a book without a title) and give the user information (error-, successmessages). Further we want to be able to display the properties of our books and be able to change them.

The ActionForm is supposed to represent a simple container, without application logic and with just two methods of importance: validate() and reset(). This is supposed to be compatible with the MVC2 paradigm. So, can we just let our Book extend that thing and hack away? Sure, we could. But think again.

Question: Why not extend the ActionForm?

Exercise: Why is it not a good idea to let our Book.java extend ActionForm? Take a look at the API and what we would have to do.

Answer: We would have to import classes from the Struts framework. This means that our model would depend on it. Think what you would have to do, to write a little GUI to create books, using our Book.java model class. It would depend on Struts too, though it has nothing to do with a web server. That is certainly not a crime, but in the way I understand MVC it is not recommended.

We will now introduce another class: BookForm.java. It will include an instance of our book and have some getter and setter methods to enable us to access the book.

Note

This new class will not create an advantage in every case. It keeps the real book Bean independant of Struts. However this is of no interest at all until you really do use the book for something else, like a GUI. If you do not however, you will waste time and resources! I asked around on the Struts mailing list and got no better solution, so here we are...

Here is the new class:

```
    BookForm.java

package books;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.*;

/*
   <b>ActionForm</b>
```

```
*/
public class BookForm extends ActionForm
{
   private Book book =new Book();
   String title = "Ye old Book";

   public void setTitle(String title)
   {
       book.setTitle(title);
   }
   public String getTitle()
   {
       return book.getTitle();
   }


   public void setBook(Book book) { this.book = book; }
   public Book getBook() { return this.book; }

   /**
    * Reset all properties to their default values.
    *
    * @param mapping The mapping used to select this instance
    * @param request The servlet request we are processing
    */
   public void reset(ActionMapping mapping, HttpServletRequest request)
   {
       this.book = new Book();
   }



   public ActionErrors validate(ActionMapping mapping,
       HttpServletRequest request)
   {
       ActionErrors errors = new ActionErrors();
       if ((book.getTitle() == null) || (book.getTitle().length() < 3))
           { errors.add("Title", new ActionError("error.book.title")); }
       return errors;
   }
}
```

We include a book and the getTitle() and setTitle() methods are accessed by our form. Nothing magic about that. What is more interesting is the last method: validate(). This is a method of ActionForm which I overwrote to enable checking on the attributes of the book. The *errors* variable is something like a Hashtable. One can put all sorts of errors in there and later display them to the user. Struts checks those and if there is even a single one it will assume the creation of the book a fault.

Before we can retry our example we will have to do some more work. It is back to struts-config.xml time. We need to associate the form with our new class instead of the Book.java. So we have to change the form-beans tag to *<form-bean name="bookForm" type="BookForm"/>*. Further we have to define the error message we want to display in case of a wrong entry. Enter something like 'error.book.title=Error with title of book' into your properties file. Now compile your classes, restart the Tomcat and reload the CreateBook.jsp. [3]

---

[3]I had to delete my 'work' directory again :-(

```
2002-04-04 14:10:28 - path="/strutsShop" :jsp: init
2002-04-04 14:10:39 - path="/strutsShop" :action: Processing a GET for /createBo
ok
2002-04-04 14:10:39 - path="/strutsShop" :action:  Looking for ActionForm bean u
nder attribute 'bookForm'
2002-04-04 14:10:39 - path="/strutsShop" :action:  Creating new ActionForm insta
nce of class 'BookForm'
2002-04-04 14:10:39 - path="/strutsShop" :action:  Storing instance under attrib
ute 'bookForm' in scope 'request'
2002-04-04 14:10:39 - path="/strutsShop" :action:  Populating bean properties fr
om this request
2002-04-04 14:10:39 - path="/strutsShop" :action:  Validating input form propert
ies
2002-04-04 14:10:39 - path="/strutsShop" :action:  No errors detected, acceptin
g input
2002-04-04 14:10:39 - path="/strutsShop" :action:  Looking for Action instance f
or class BookAction
2002-04-04 14:10:39 - path="/strutsShop" :action:  Double checking for Action i
nstance already there
2002-04-04 14:10:39 - path="/strutsShop" :action:  Creating new Action instance

Start perform(BookForm@c80b01) . . .
After creation of book: testBook
2002-04-04 14:10:39 - path="/strutsShop" :jsp: init
```

Figure 5.5. Book Creation and Validation

Question: Changing the title

Exercise: Take a closer look at the output. Now enter another title for the book. What happens if you enter no title at all?

What did we learn? We talked about the pros and cons of using Struts for the automatic setting of the state of our Book.java Bean. We introduced another class, the ActionForm and we saw what happens if an invalid entry is made. We still don't have a real book created, yet, though.
Annotations [*add annotation*]

Annotations

Viktor Todorov: [. . .] My (first?!)question is: - how does the acton name "createBook" changes to "createBook.do" when submiting a request? Who added ".do" to "createBook" and when it is done??? I think it is the "action" servlet but how...?

Stephan: The web.xml defines what to do with requests ending with '.do': <url-pattern>.do</url-pattern>. They are forwarded to the org.apache.struts.action.ActionServlet. The '.do' is stripped and the struts-config.xml is parsed for a fitting target (CreateBook.jsp in our example).*

Vince from Toronto: Vince from Toronto in Canada mentioned the following: In chapter 6, you discuss about the pros and cons of using the Book class versus a new BookForm class. Another reason is to maintain the contents of the original information in the Book class. As you develop more complex applications that interact with a database, you want the validation to complete successfully

before updating a record in a database. The BookForm class is like a tempo-rary class until all validation is successful.

5.3. A better way to seperate Book and Book-Form

In the previous chapter we used a new class BookForm.java to access our Book.java without implementing Struts directly into it. I will now show you a way to do it without implementing all the getter and setter methods twice.

Struts allows us to access the methods of instance variables directly. It took me some time to understand it, I have to admit, and I'm not sure I like the concept. It is quite easy, however.

Here is the changed CreateBook.jsp:

```
        <html:form action="createBook.do" method="GET">
            Title:<html:text property="book.title" /> <br/>
            Pages:<html:text property="book.pages" /> <br/>
            <html:submit property="submit"/>
        </html:form>
```

As you can see, I use *'book.title'* instead of 'title'. I didn't make any other changes! Of course we can remove some methods from BookForm.java now, but we don't have to! This works only because we have a getter and setter for the instance variable 'book' in BookForm.java. Don`t forget to adapt the *BookView.jsp* as well, or you will get an error saying that the getter for title can't be found.

So, we can reduce BookForm.java to:

```
    BookForm.java, reduced

package books;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.*;

/*
  <b>ActionForm</b>
*/
public class BookForm extends ActionForm
{
  private Book book = new Book();

  public void setBook(Book book) { this.book = book; }
  public Book getBook() { return this.book; }

  /**
   * Reset all properties to their default values.
   *
   * @param mapping The mapping used to select this instance
   * @param request The servlet request we are processing
   */
  public void reset(ActionMapping mapping, HttpServletRequest request)
  {
      this.book = new Book();
  }

  public ActionErrors validate(ActionMapping mapping,
      HttpServletRequest request)
  {
      ActionErrors errors = new ActionErrors();
      if ((book.getTitle() == null) || (book.getTitle().length() < 3))
```

```
        { errors.add("Title", new ActionError("error.book.title")); }

    if (book.getPages() < 1)
        { errors.add("Page", new ActionError("error.book.page")); }
    return errors;
  }
}
```

Annotations [*add annotation*]

Annotations

[. . .] Stephan, Great tutorial, spotted one small omission. In 7. after doing the book.title and book.pages changes, the title was not being displayed - it was null. Small change also required in BookAction [displayed on the console]: System.out.println("Start perform(" + form + ") . . ."); String title = req.getParameter("book.title"); Book book = new Book(); book.setTitle(title); System.out.println("After creation of book: " + book.getTitle()); "title" has changed to "book.title". Regards, Tom Smith

Edward Brode pointed out that we can use the following in the BookAction.java: "BookForm bf = (BookForm)req.getAttribute("bookForm");" This safes us the overhead of initiating a new Book and setting the attributes for that one.

5.4. Handling Invalid Entries

Abstract

I will show you how Struts enables you to react to invalid user entries. Error messages suppport different languages.

I will continue directly where the exercise in the previous chapter ended. There you saw a message in the output from Tomcat that said something about an error occuring and a redirection.

```
    action: Populating bean properties from this request
    action: Validating input form properties
    action: Validation error(s), redirecting to: /CreateBook.jsp
```

The error occured, when trying to create a book with an empty title. If you take another look at the BookForm.java source, you will see that I defined a check in the validate method. With that I told Struts what an error (for my book) is. The validate method will be called by the BookAction class. However, I didn't tell it what to do in case of an error and I never commanded no one to print an errormessage. So the default behaviour kicked in and we ended where we started. Or did we? What happens if you enter a Title with just a single letter? It should lead to another error. And so it does. The CreateBook.jsp is displayed again and the textfield is set. This is very important! You might not have realized it but you *loaded a new page* and the form got filled out as best as Struts could!

Question: Default value for number of pages

Exercise: Modify the application in a way that the number of pages defaults to one and that there is an entry field for it. Try to make it work and pay attention to the values of you book before and after creation. By the way, what happened to the default value of BookForm.title?

Answer: We would have to import classes from the Struts framework. This means that our model would depend on it. Think what you would have to do to write a little GUI to create books, using our Book.java model class. It would depend on Struts too, though it has nothing to do with a web server. That is certainly not a crime, but in the way I understand MVC it is not recommended.

```
The changed CreateBook.jsp

   <h2>Create a book</h2>

   <html:form action="createBook.do" method="GET">
       Title:<html:text property="title" /> <br/>
       Pages:<html:text property="pages" /> <br/>
       <html:submit property="submit"/>
   </html:form>
```

We don't need to adapt the struts-config.xml or the BookAction.java. What you will (hopefully) see is that the form is always filled with the actual values of our book. In case of the pages it is filled with the default value even before we entered anything into the form. This means that a book is created the first time the page is displayed. If you are watchfull, you discovered something else. When entering correct data we were automatically transfered to BookView.jsp! That is not easy to see, as we had the same contents on the page, but now with the change it becomes apparent.

Having learned all this it is now finally time to do something about the output. We certainly want an errormessage displayed, if one exists and we want our book properly displayed if it got created sucessfully.

I will start with the error messages. They should be displayed on the CreateBook.jsp page, as we get redirected here in case of an error. This one is extremely easy. Just enter the following line into the CreateBook.jsp and enter a book with a faulty title.

```
<html:errors/>
```
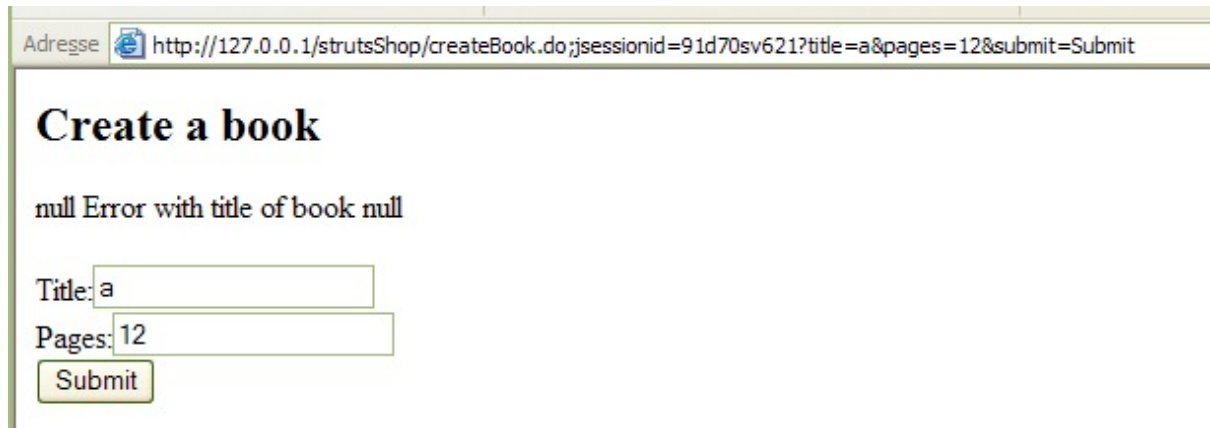
Figure 5.6. Our first Error Message

As the display is pure HTML you can put some color attributes around it to make it easier to spot. What if we want not all, but just one message to be displayed? We just can say which to use: <html:errors property="title"/> Go, try it out.

Hmh, that didn't work? Okay, take a moment to think. Where does the error come from? We defined it in the BookForm.java. Taking a close look will reveal our mistake. We gave the error the name 'Title' with a capital T. These kind of mistakes are very common and hard to debug. There is not really an error, we just don't get no message at all.

Now, what about those ugly 'null' words around our error message? First modify your code so that the number of pages can generate an error, too. If you let it fire you will see that we still have one null at start and end each. So it must be something global for the ActionErrors class. A quick survey of the docu which comes with Struts will enlighten us. It is indeed something global. It enables us to define a wrapper around our error messages. Change your properties file to contain the following, restart the Tomcat and reload your page.

```
index.title=Struts Tutorial
error.book.title=<member>Error with title of book</member>
error.book.page=<member>A book needs at least one page</member>
errors.header=<h3><font color="red">Error: <UL>
errors.footer=</ul></font><hr></h3>
```
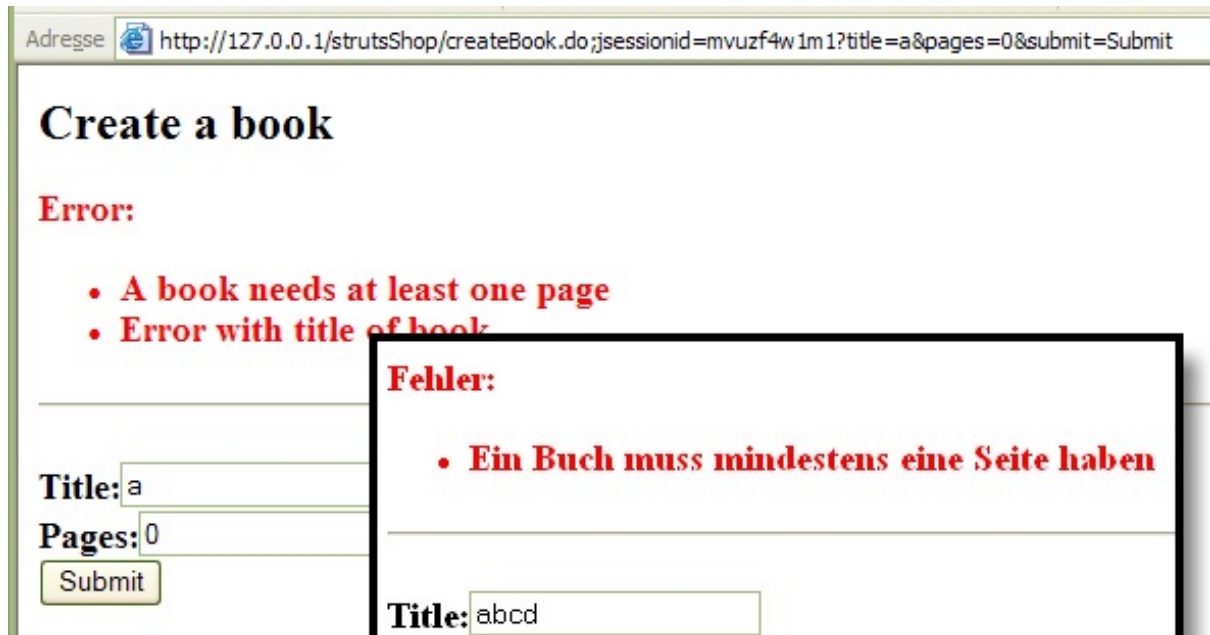
Figure 5.7. Correct Error Messages

Annotations [*add annotation*]

What did we learn?

We finally managed to create a book and to display correct errormessages. In case you didn't realize it: They are sortet alphabetically.

Annotations

Anita Carpenter had difficulties displaying the error message about the wrong number of pages, when creating a book with zero pages. This is my reply: [...] "You only call the error of the title. Just insert <html:errors/> to display all errors or <html:errors property="Pages"[page] /> for the number of pages."

5.5. Recapitualation: Where we are

We have reached quite an understanding of Struts by now and it is time to survey our knowledge before we get to some new ideas.

By now we can display text using internationalisation features from Struts to support different languages. We created a book Bean and it took quite some time and effort to use Struts for automatic setting of its values, including errorhandling. We discovered lots of nice advantages the use of Struts gives us, but we painfully discovered how much work it can become. Don't even think about all those little classes we will have to write for our applications: XBean, XForm, XAction, maybe special error handlers, etc. And, of course, there is

Tomcat. We have to restart our Server for most changes to take effect (I work for a company were we had a project with really huge amounts of data. We developed a content management system which took half a day to restart the server. Not Tomcat, though...).

So, where does that leave us? Do we even like Struts? Want to work with them? I honestly can't say that for myself, yet. I have to use them for work at my university, so I don't have a decission to make. You probably can, though. My advise is, of course, to follow me through the next chapters and find out what else there is. Then you can decide for yourself.

Question: Extend your classes: Author and CD

Exercise: Adapt your classes so that you can set a single author for the book, too. Let the error messages be displayed right next to the corressponding input field. Try to define a second bean, complete with input form and error handling. Take a CD or DVD for example.

Answer: By now you should be able to do it on your own, using the chapters above. See Section B.1, "CDs" for some code for this.

Question: Edit struts-config.xml

Exercise: Edit your struts-config.xml. Change the action-mapping from input="/CreateBook.jsp" to input="/BookView.jsp". What happens when you enter a new correct/incorrect book?

Answer: Just try it out :-)

Annotations [*add annotation*]

Chapter 6. Using Logic

Abstract

This chapter consists of two parts. A simple introduction, that shows how it works, but concludes that it is not worth using it, and a more advanced example, deomonstrating that , maybe, it is worth using, after all.

Goal: Understand how Logic Tags can make our lives easier and if they are even worth the effort to learn about them (my personal first impression was quite in the contrary). **Annotations** [*add annotation*]

6.1. First Try at Logic

Lets start with an iterator. We will do it the old way first to see how that works, change to Struts afterwards and compare the two ways. Modify your BookView.jsp to include the following:

```
    First Iterator, no Struts

<%
  Book books[] = new Book[3];
  books[0] = new Book();
  books[0].setTitle("Book 0");
  books[1] = new Book();
  books[1].setTitle("Book 1");
  books[2] = new Book();
  books[2].setTitle("Book 2");
  for(int i = 0; i < 3; i++)
  {
      out.println("<member>" + i + ") " + books[i].getTitle()
      + "</member>");
  }
%>
```
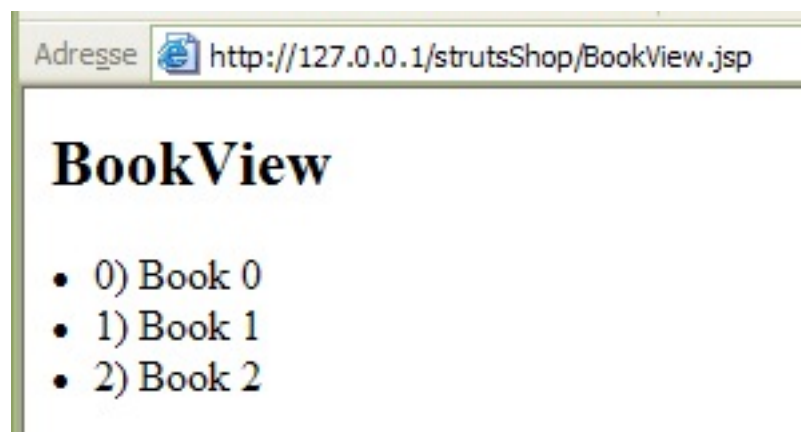

Figure 6.1. Iterate, using for-loop

That one was standard stuff, so lets turn to Struts and see how it can help us.

```
    Iteration, using Struts
```

```
   <ol>
   <logic:iterate id="myBooks" collection="<%= books %>">
     <member>
       Title = <bean:write name="myBooks" property="title"/>
     </member>
   </logic:iterate>
   </ol>
```

The result is the same. So, you see, it is quite easy to use Struts for iteration. But, you might ask (I certainly did), where is the benefit? We exchanged only one command and there we removed the good old for-loop, which we happen to know very well, and replaced it with the Iterate tag. I admit quite frankly that I don't like this. Not at all. The Struts-code does look cleaner and I don't have to use JSP tags and 'real Java code', but in my humble opinion not every thing that is new is good. So, my personal advice is to try it out and to decide for yourself what to use. I will do exactly that and maybe, someday, I will update this tutorial and think how stupid I was not to see the real benefit. You are welcome to teach it to me, by the way...

Annotations [*add annotation*]

Annotations

Didier Dubois: "I try it [the tutorial] under WSAD and it works very well (Websphere does not need to be restarde if I only change JSP or a Bean.) However, I think I have a good answer for your text in chapter 10 about the benefit of using logic vc. plain JSP. It is because of the MVC stuff. The rule is "Do not NEVER, EVER use Java inside your JSP. NEVER!" But you're true: if we look @ the 2 pieces of code we do not see the real benefit of the <logic:> tags. This is as reason why JSP is a very weak technical solution (but a very good political one: your boss will be happy with that :-) If you want some more detailed information about that, just go to http://jakarta.apache.org/turbine/common/further-reading.html. It is very interesting. The link to the Jason's article will give you some deeper sight about that.

Tony: Hi Stephan, [...] I was reading through several tutorials that were pretty theoretical but I have to have some guide that shows me how to get a hand on the practial stuff. That's when I found yours. Maybe it's a personal thing but I prefer learning things the practical way. I'm currently on chapter 10 now and you're asking why logic should be any better than doing it the "old" implementation style... well. you're right. from a programmer's point of view, implementing it directly might be the more transparent choice. If you work in a team (as I do) and really have to abstract a business model and hand control, web design and layout over to a designer then I think you're better off doing it the struts <logic> style. just my 5 cents, cheers tony

6.2. Advanced Logic

Abstract

This section describes a more advanced scenario: We have a class BookCatalog, which contains a number of books and we want to present those books. The

user should be able to sort them by title and by pages.

Note

Please note, that the following code is not intended for anything else but demonstration. I included two different approaches in the BookKatalog.java file, just to demonstrate different ways and certainly not because I consider this good style.

Consider Implementing Comparable

By implementing Comparable, you allow your class to interoperate with all of the many generic algorithms and collection implementations that depend on this interface. You gain a tremendous amount of power for a small amount of effort. Virtuall all of the value classes in the Java platform libraries implement Comparable.

—Joshua Bloch, Effective Java, Item 11

Let's start with the Book.java. We need some mechanism for the sorting. I decided to use the interface comparable. The interface and the last methods are new, the rest is the same as in the examples above.

```
    Book.java

package books;
import java.util.Vector;

/*
    A simple book.
    @author stephan@stephanwiesner.de
*/
public class Book implements Comparable
{
    /** The title */
    private String title = "";
    /** We can have more than one author */
    private Vector authors = new Vector();
    /** The number of pages the book has */
    private int pages = 0;


    /** Standard constructor. */
    public Book()
    { }

    /** @param title The new Title    */
    public void setTitle(String title)
    {   this.title = title;    }

    /** @return The title. */
    public String getTitle()
    { return this.title; }


    /** @param pages The new number of pages. */
    public void setPages(int pages)
    { this.pages = pages; }


    /** @return The number of pages. */
    public int getPages()
    { return this.pages; }

    /**
        We don't want to work with the Vector here, as it is
        only a reference we would get!
        @param author Add another author
    */
    public void addAuthor(String author)
    { this.authors.add(author); }
```

```
    /**
        Pay attention not to use the wrong number.
        @param position The number of the author to remove.
    */
    public void removeAuthor(int position)
    { this.authors.remove(position); }

    /** @return The number of authors the book has. */
    public int getNumberOfAuthors()
    { return this.authors.size(); }


    /**
      @param otherBook The book with which to compare
      @return a negative integer, zero, or a positive integer as the the
      title of the specified Book is greater than, equal to, or less than the
      title of this Book (String.cmpareToIngnoreCase()).
    */
    public int compareTo(Object otherBook)
    {
        if (! (otherBook instanceof Book) )
        { return 0; } // we don't know how to handle this

        return this.title.compareToIgnoreCase(((Book)otherBook).getTitle());
    }
}
```

```
    BookKatalog.java

package books;

import java.util.*;

public class BookKatalog
{
    private SortedSet books = Collections.synchronizedSortedSet(new TreeSet());
    private List list = Collections.synchronizedList(new ArrayList());

    public BookKatalog()    {    }

    public void addBook(Book book)
    {
        books.add(book);
        list.add(book);
    }

    public Iterator getBooks(){ return books.iterator();  }

     /**
      Sorts the quotes. Uses shellSort time complexity of n^(6/5))
     */
    public void sortByPages()
    {
        int[] schrittweiten = { 7,3,1};
        for (int i = 0; i < schrittweiten.length; i++)
        {
          int schrittweite = schrittweiten[i];
          for (int z = 0; z < list.size(); z++)
          {
            Book temp = (Book)list.get(z);
            int stelle = z;
            while ((stelle - schrittweite >= 0)
                && (temp.getPages() < ((Book)list.get(stelle - schrittweite)).getPages()) )
            {
                list.set(stelle, list.get(stelle - schrittweite));
                stelle -= schrittweite;
            }// while
            list.set(stelle, temp);
          }// for z
        } // for schrittweiten
    }


    public Book getBook(int i) {  return (Book)list.get(i);    }


    public int numberOfBooks()
    { return list.size(); }
}
```

```
    BookKatalog.jsp

<%@ page import="java.util.*, books.*" %>

<%@ page language="java" %>
 <%@ taglib uri="/WEB-INF/struts-bean.tld"  prefix="bean" %>
 <%@ taglib uri="/WEB-INF/struts-html.tld"  prefix="html" %>
 <%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

 <html:html locale="true">
 <head>
 <html:base/>
 <title>
    <bean:message key="index.title"/>
 </title>
 </head>
 <body>
    <h2>BookKatalogJSP</h2>

<%
   BookKatalog bk = new BookKatalog();

   Book b1 = new Book();
      b1.setTitle("Book 1");
      b1.setPages(100);
      bk.addBook(b1);
   Book b3 = new Book();
      b3.setTitle("Book 3");
      b3.setPages(300);
      bk.addBook(b3);
   Book b2 = new Book();
      b2.setTitle("Book 2");
      b2.setPages(200);
      bk.addBook(b2);
   Book b4 = new Book();
      b4.setTitle("Book 4");
      b4.setPages(20);
      bk.addBook(b4);

      Iterator i = bk.getBooks();
      while (i.hasNext())
      {
         System.out.println("iterating...");
         Book book = (Book)i.next();
         System.out.println("Have book");
         out.println("<li>" +  book.getTitle() + "=" + book.getPages() + "</li>");
      }

      out.println("<hr>");
      bk.sortByPages();

      int z = 0;
      while (z < bk.numberOfBooks())
      {
         out.println("<li>" + bk.getBook(z).getTitle() + "="
           + bk.getBook(z).getPages() + "</li>");
         z++;
      }
%>

 </body>
 </html:html>
```

As you can see, the JSP is quite primitive. It creates a BookKatalog, adds some books and then prints them to the screen. Take a moment to consider what the output will be. See Figure B.1 for the screenshot.

The JSP above did not use Struts for the iteration yet. This can be done like this:

```
    BookKatalogJSP.jsp

 <ol>
 <logic:iterate id="myBooks" collection="<%= bk.getBooks() %>">
   <member>
    <li> Title = <bean:write name="myBooks" property="title"/></li>
   </member>
 </logic:iterate>
 </ol>
```

```
<hr>

<ol>
<%
   bk.sortByPages();
%>
<logic:iterate id="myBooks" collection="<%= bk.toArray() %>">
   <member>
    <li> Title = <bean:write name="myBooks" property="title"/></li>
   </member>
</logic:iterate>
</ol>
```

This looks somewhat cleaner, but still not like too much of an advantage. Now, consider the next part.

First add the following two lines to the language properties file:

```
   BookKatalog.jsp

book.pages=Number of Pages
book.title=Booktitle

book.pages=Anzahl Seiten
book.title=Buchtitel
```

```
   BookKatalogJSP.jsp, 2nd Edition

<table border="1">
<%
     Iterator i = bk.getBooks();
     while (i.hasNext())
     {
         System.out.println("iterating...");
         Book book = (Book)i.next();

         out.println("<tr><td>");
         %>
         <bean:message key="book.title"/>
         <%
         out.println("\"" + book.getTitle() + "\"");
         %>
         </td><td>
         <bean:message key="book.pages"/>
         <%
         out.println(book.getPages() + "</td></tr>");
     }
%>
</table>


<br />   <br /> <br />

 <table border="1">
 <logic:iterate id="myBooks" collection="<%= bk.getBooks() %>">
   <member>
    <tr><td> <bean:message key="book.title"/>
      "<bean:write name="myBooks" property="title"/>"
    </td><td>
       <bean:message key="book.pages"/>
       <bean:write name="myBooks" property="pages"/>
    </td></tr>
   </member>
 </logic:iterate>
 </table>



<hr>
<form action="BookKatalogJSP.jsp" method="GET">
 <select name="selectedBook">
 <logic:iterate id="myBooks" collection="<%= bk.getBooks() %>">
   <member>
    <option> <bean:message key="book.title"/>
      "<bean:write name="myBooks" property="title"/>"
       <bean:message key="book.pages"/>
       <bean:write name="myBooks" property="pages"/>
```

```
    </option>
   </member>
 </logic:iterate>
 <input type="submit">
 </select>
</form>
```
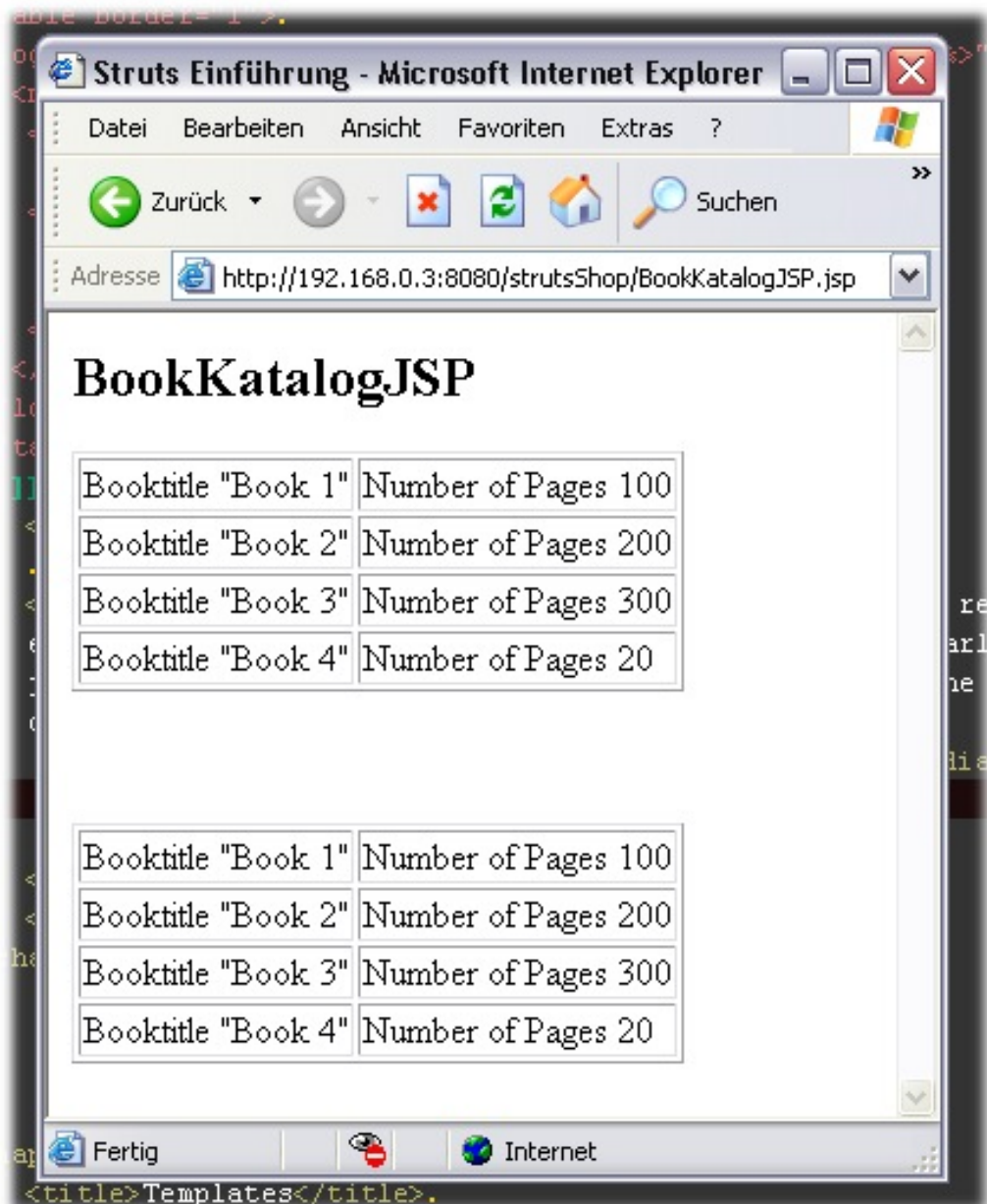
The code snippet above consists of three parts. The results of which are the same, but the code of the second one is clearly 'cleaner' than that of the first. This was just a small example, the larger the application gets the stronger the difference, of course.

Figure 6.2. Our First Template displayed

Annotations [*add annotation*]

Chapter 7. Templates

Using Template Tags

> 'The "struts-template" tag library contains tags that are useful in creating dynamic JSP templates for pages which share a common format. These templates are best used when it is likely that a layout shared by several pages in your application will change. The functionality provided by these tags is similar to what can be achieved using standard JSP include directive, but are dynamic rather than static. '
>
> —Struts Manual

I will start with the 'old fashioned' way JSPs offer for the use of templates. Consider the scenario that every page of your side has to contain the same text, say a copyright at the bottom. We certainly don't want to copy the text into all those pages and what if we have to change it? JSP offers us the possibility to include pages, so we can do something like:

```
Using JSP Include

   <html>
    <body>
     <h2>Hello World</h2>
     <jsp:include page="/copyright.jsp" />
    </body>
   </html>
```

I admit, until now the aproach above worked perfectly for me. It is simple, straight forward and easy to adapt. There is a major drawback, though. What happens if you change your layout? If the copyright notice from above is suddenly supposed to appear at the top of the page, you will have to adapt every page...

Note

In my personal oppinion this is a minor drawback. A complete rework of the layout is always a very serious decision and will never be accomplished by changing a few lines of code. I have developed a page or two and I have used templates in Perl, ServerSide JS and PHP as well as with JSP and it did help me a lot. But, I never had a single template for all pages. So, if you read that Struts will enable you to do exactly that, don't bet your career on it...

Now, let us take a closer look at the way Struts uses templates. Create a directory 'templates' right in your root directory and enter the following files:

```
Template: Head.jsp

<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld"  prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld"  prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html>
<head>
<html:base/>
<title>
<bean:message key="index.title"/> - Template
```

```
    </title>
    </head>
```

```
    Template: Navi.jsp

    <b>My Links:</b>
    <ul>
      <member><a href="http://jakarta.apache.org/struts">Struts</a></member>
      <member><a href="http://java.apache.org">Java @ Apache</a></member>
      <member><a href="http://java.sun.com">Download JDK</a></member>
      <member><a href="http://www.stephanwiesner.de">Stephan Wiesner</a></member>
    </ul>
```

```
    Template: Foot.jsp

    <hr/>
    <center><%=new java.util.Date()%>, by Stephan Wiesner</center>
    </body></html>
```

Now we need to define our master template.

```
    Template: BookTemplate.jsp

    <%@ page language="java" %>
    <%@ taglib uri='/WEB-INF/struts-template.tld' prefix='template' %>

    <template:get name='head'/>

    <body>
    <table border="1" height="400" cellpadding="10" cellspacing="10">
      <tr valign='top'>
        <td>
          <template:get name='navi'/>
        </td>
        <td>
          <template:get name='content'/>
        </td>
      </tr>
    </table>

    <template:get name='foot'/>
```

This is the layout for all our pages. We define that the head element will consist of a template called 'head'. What the head contains is not clear at this moment (we could enforce it, of course).

```
    Template: CreateBookTemplate.jsp

<%@ taglib uri='/WEB-INF/struts-template.tld' prefix='template' %>
<template:insert template='/templates/BookTemplate.jsp'>

    <template:put name='head' content='/templates/Head.jsp' />
    <template:put name='foot' content='/templates/Foot.jsp' />
    <template:put name='navi' content='/templates/Navi.jsp' />
    <template:put name='content' content='/BookView.jsp' />
<!--jsp:include page="/BookView.jsp" flush="true" /-->
</template:insert>
```

We have a complete template now, so go ahead and try it out.

Figure 7.1. Our First Template displayed

Note

I included a few pages, just to show how it works. My personal experience is, that most layouts will be more static. Head, Foot and Navigation are always the same (layout, not the contents), while the actual content of the pages will vary. So you would end with a structure that consists of JSP include tags and don't really use Templates. The idea itself is very fascinating though and I would like to get some mails with links of working, real world examples. Though I probably will not be allowed to look at the actual code, so I could only guess what really happens . . . ?

What did we learn? We are now able to define the layout of many web pages in a very consistent way. We can change the layout very fast and with hardly any work at all.

Annotations [*add annotation*]

Annotations

Daniel Novy: Experienced a StackOverflowError with the CreateBookTemplate.jsp. This occured, because the template included itself, which led to it in-

cluding itself, which led to . . . well, an StackOverflow :-(This is fixed by now.

Chapter 8. Beans to XML

Use XML files for serialization of created books. JDK1.4 needed!

This chapter has nothing to do with Struts. So, if you are only after knowledge about that, you might savely skip it. If you do not have a lot of experience with Servlets/JSP you might learn a trick or two anyway, though. I certainly did... *You will need JDK1.4 or above to make the examples work!*

In this chapter I will present the classes XMLEncoder and XMLDecoder, which ship with JDK1.4 and enable us to use XML for serialization. Why is that good? Well, if you are in doubt and have used the old way to serialize objects before, skip ahead and look at the output generated. Instead of binary data you get clean, easy to understand XML. You can use this XML for data exchange or process it in any way you want. And, most important, you can *change your class and still read the XML!*

First we will have to let our Book.java Bean use Serialization. Add 'implements java.io.Serializable' to the class definition. That is all we need to change! Well, now we have to find a way to actually save our book, but we don't have to adapt it any other way.

Now create a new JSP page. Call it BookSerialize.jsp and enter the following (for simplicity we will start without Struts):

```
BookSerialize.jsp

<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld"  prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld"  prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>
<%@page import="java.beans.XMLEncoder, java.beans.XMLDecoder, java.io.*"%>

<html:html locale="true">
<head>
<html:base/>
<title>
<bean:message key="index.title"/>
</title>
</head>

<body bgcolor="white">
<h2>Book Serialization</h2>

<%
    Book book = new Book();
    book.setTitle("A great Book about Struts");
    book.setPages(100);
    out.println("<member>First Title: " + book.getTitle() + "</member>");
    try {
        XMLEncoder encoder = new XMLEncoder( new BufferedOutputStream(
            new FileOutputStream("/Sample.xml")));
        encoder.writeObject( book );
        encoder.close();
    }catch(Exception ex) { out.println(ex); }

    book.setTitle("A great Book about Struts, Second Edition");
    out.println("<member>Second Title: " + book.getTitle() + "</member>");

    try {
        XMLDecoder decoder = new XMLDecoder(
            new BufferedInputStream(
            new FileInputStream("/Sample.xml")));
        book = (Book) decoder.readObject( );
        decoder.close();
    }catch(Exception ex) { out.println("<h1>" + ex + "</h1>"); }

    out.println("<member>Third Title: " + book.getTitle() + "</member>");
%>
</body>
```

```
    </html:html>
```
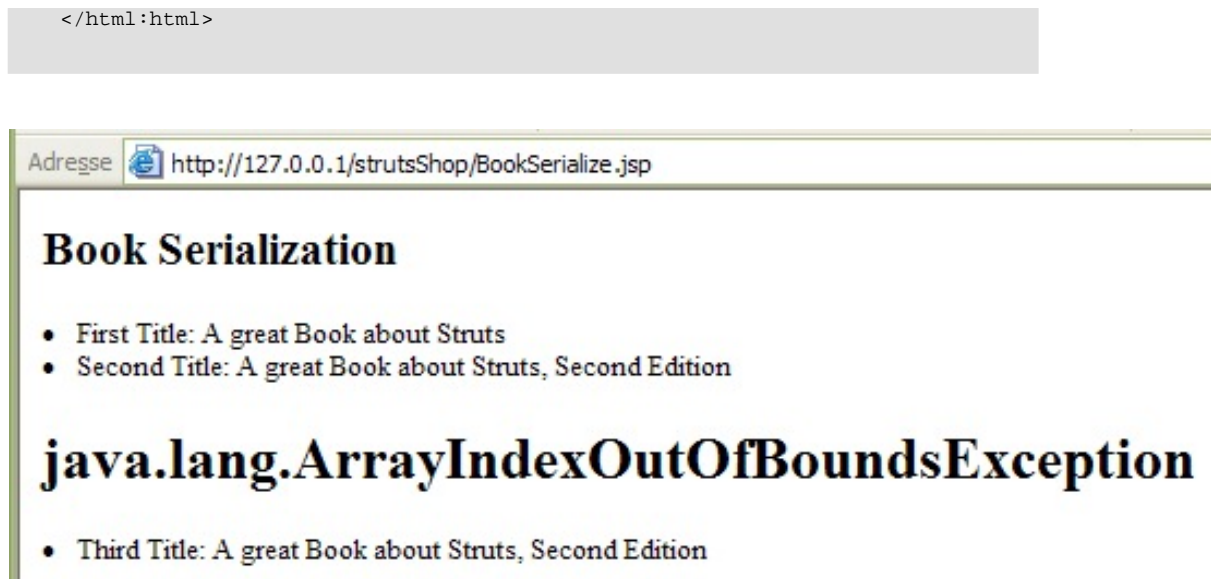

Figure 8.1. Error Serializing a Bean

That is ceratinly not what we expected. What happened? To find out we will start with a look into the generated XML file. Looking into our directories we see: Nothing. No file called Sample.xml seems to be generated. Now comes part one of the tricky part: The file is created on root! For me, running Tomcat on Windows from disk c:\ this means c:\Sample.xml. That is a general, often neglected problem with Tomcat. It is, after all, just a Java application and so it thinks you are running your application from the directory where you startet Tomcat. Root is the root disk for that directory!

> **Question: What is DoucmentRoot for Tomcat?**
>
> **Exercise: Change the code to save the file to 'new FileOutput-Stream("Sample.xml")));' (no slash). What happens?**
>
> Answer: Take a look at the created file Sample.xml. What do you see? Any idea what happened? Take a look at the Java API to see how it is supposed to look. A hint: Take a look at the output generated by Tomcat. Why do you think does that error message appear?

Okay, I will recapitulate: We try to serialize a Bean using a XML file. We encounter two problems: The file is not generated where we want it and the state of the Bean is not even saved.

First lets correct the saving, as that is the most pressing problem. Tomcat says that it can't find our Book.class. Hmh, as that might be a JSP problem, lets try something else. We introduce a new method into the book, enabling it to save itself. That should work, as the book should at least find itself!

Do it! Import the needed classes and copy the code for saving to a method caled

'write()'. Then call this method from your JSP. Don't forget to restart Tomcat!

If you did the last exercise, you might wonder why it still doesn't work. How can it be, that Book.java can't find even itself? It is running, we see that! To be real sure I introduces a main() method into it and ran it from the console. Sure enough, everything worked fine. So, what can we do now?

We adapt our CLASSPATH. We include the directory containing our Book.java file. Don't forget to restart Tomcat, ensuring that the change took effect (on Windows NT/2000/XP you have to open a new dos console, under Win98 reboot to be sure). Now it works!

Note

This seems not an ideal solution. We have to consider how a normal MVC application is deployed though. The idea is to reuse the model classes, so they have to be on the CLASSPATH anyway.

Now, what to do about the generated file beeing saved in the wrong directory? There are different solutions. The easiest, most straightforward one, which is of course the worst, is to hardcode the path into the JSP. In fact this is not too bad an idea. Think about the possibilities to solve it. You can import a page with JSP, so you could define a global JSP page with all the variables you need and import it. That solution is easy to implement, to manage and to adapt in case of an implementation on a foreign system. We could there choose to use a different file structure. I have used this aproach a few times and it worked fine.

We can use a little trick to get a hand on our actual location:

```
out.println("<member>File:" + this.getClass().getResource(
    "Sample.xml" ).toString() + "</member>");

 Will print (on my system):
   File:file:/C:/tomcat/webapps/strutsShop/WEB-INF/classes/Sample.xml.
```

Another solution is to ask Tomcat where we are:

```
config.getServletContext().getRealPath(".");
```

This will give us the absolute path to our root application, in my case it will give me: C:\tomcat\webapps\strutsShop\.

What did we learn in this chapter? I told you how to save/load the state of a Bean in a XML file, using a new serialization mechanism in JDK1.4. This led us to a deeper look into the working of Tomcat, concerning the handling of file locations.

Question: Two authors

Exercise: Change the code so that you create and save a book with 2 different authors. How does the Sample.xml look like now?

Question: Replace JSP with Struts

Exercise: We used 'normal' JSP to display our books in this chapter. Change it to the use of Struts.

Answer: I admit, this one was tricky. You have to add the book to the context in order to access it:

```
<% session.setAttribute("myBook", book); %>
<bean:write name="myBook" property="title"/>
```

Annotations [*add annotation*]

Chapter 9. Replacing JSP with XSLT

Abstract

This chapter will present a way to use XML/XSLT for presentation, replacing JSP as a view component. JDK1.4 and Netscape6 or IE6 are needed for this chapter. It is strongly recommended, that you work through the Chapter 8, *Beans to XML* (XML serialisation) first.

Goal: Use XSLT to transform generated XML, reducing the need for JSP.

The use of XML/ XSL for data transportation and transformation is widely accepted. In this chapter I will show how easy it is to display our Book.java from the first chapters, using those technologies. This is not an introduction into XML, but you don't need to have any experience to understand it.

I will start with a Servlet for the generation of XML.

```
XMLOutServlet.java

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.beans.*;

public class XMLOutServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException
    {
        response.setContentType("text/xml");
        XMLEncoder encoder = new XMLEncoder(response.getOutputStream());
        Book book = new Book();
        book.setPages(100);
        // will NOT be written!
        book.addAuthor("Stephan");
        book.setTitle("XML Book");
        encoder.writeObject( book );
        encoder.close();
    }
}
```

XMLOutServlet creates a book, sets some attributes and then uses serialization to generate a XML representation. Compile it, restart your Tomcat and take a look at the output. You will need JDK1.4 for the XMLEncoder and a XML capable browser for the display.

```
Adresse  http://127.0.0.1/strutsShop/servlet/XMLOutServlet

  <?xml version="1.0" encoding="UTF-8" ?>
- <java version="1.4.0" class="java.beans.XMLDecoder">
  - <object class="Book">
    - <void property="pages">
        <int>100</int>
      </void>
    - <void property="title">
        <string>XML Book</string>
      </void>
    </object>
  </java>
```

Figure 9.1. Display the Book as XML

Question: Why is the author attribute not serialized?

Exercise: Why is the author not part of the generated XML? If you are in doubt whether it is actually set, include a print statement in the setter method.

Answer: The serialization process works only for attributes with a getter and setter method. Our author is set indirectly, though. So, if you want the authors to be displayed you must include public setter and getter methods for the Vector. You still can use the addAuthor() method to set an author, of course.

Now, though that was quite easy, it doesn't look too good. This is were XSLT enters the stage. XSLT is a language to transform XML. A possible target format is, of course, HTML. Modern browsers are capable of doing such transformations themself, so we don't even have to do it on the server. To support older browsers we will do it the hard way, though. I will not cover the huge topic of XSLT in detail here. I will present a working example, though and say a word or two about it.

To be continued . . .
Annotations [*add annotation*]

Chapter 10. Introducing AspectJ

This chapter will present an elegant way to modify your application without touching your code. This will enable you to handle higher complexity and to better seperate your developing environment from the code that goes actually into production.

Use Aspect Oriented Programming (AOP) with AspectJ for logging and timing.

See http://www.stephanwiesner.de/java for a introductional tutorial about AspectJ. I will from here on assume that you know what AspectJ [4] is and have at least tried an example or two. Of course you must have it installed to try the examples. This is not an introduction into AspectJ! I will just show how easy it is to use as an enhancement to existing code.

Let's start with logging. If you tried the tutorial above you should recognize the idea. That is not because I lack imagination, but because I think it so usefull. Considering the fact that the displayed error messages of the Servlet engine can be quite confusing and often don't help, it would be nice to have a logging feature that can be switched on with a single switch, that doesn't cost performance when not needed and that doesn't clutter up our beautiful code. AspectJ to the rescue.

Create a new file called Logger.java. This will be our Aspect file and we will introduce a logging mechanism for every method of our Book, without touching the code of Book.java. Copy the following code into your file:

```
Aspect Logger.java

public aspect Logger
{
    pointcut log(): call(* Book.*(..)) ;

    before() : log()
    {
        System.out.print("Enter " + thisJoinPoint.toLongString()
            + " Args:'");
        Object[] os = thisJoinPoint.getArgs();
        for(int i = 0; i < os.length; i++)
        {
            System.out.print(os[i].getClass() + "(" + os[i] + ") ");
        }
        System.out.println("'\n");
    }
}
```
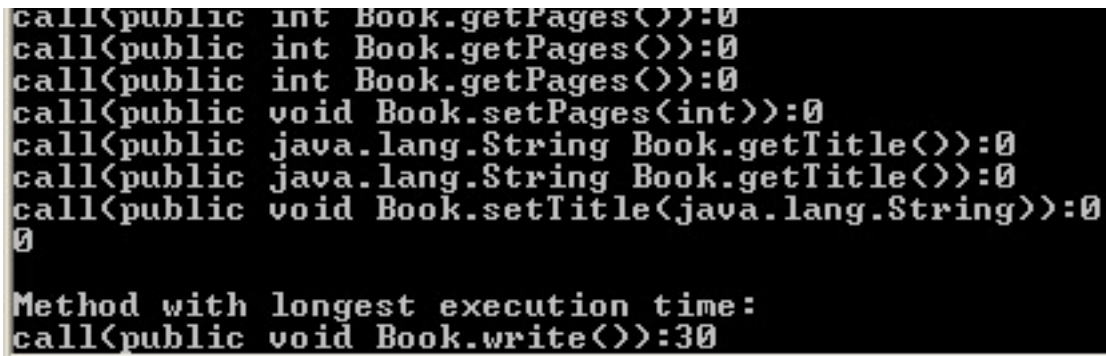
To start it type *'ajc Book.java Logger.java'* in your classes directory. That is all. Restart Tomcat, reload the BookSerialize.jsp page and look at the output from Tomcat.

---

[4]If you don't know what XEROX is, spend some time to find out. They gave us the mouse, the laser printer, network card (to name a few) and now AspectJ

Figure 10.1. Logging with AspectJ

Want to know which method takes how long to exeute?

```
Aspect Timer.java

import java.io.*;
import java.util.*;
import java.text.*;

public aspect Timer
{
    Locale currentLocale = new Locale("DEU", "DEU");
    DateFormat dateFormatter =
        DateFormat.getDateInstance(DateFormat.DEFAULT, currentLocale);
    SimpleDateFormat formatter = new SimpleDateFormat(
        "ssSSSSSS", currentLocale);
    int zeit = 0;
    String maxZeitMethode = "";
    int maxZeit = 0;

    pointcut timer(): call(* Book.*(..)) ;

    before() : timer()
    {
        System.out.print(thisJoinPoint.toLongString() + ":");
        zeit = Integer.parseInt( formatter.format( new Date() ) );
    }

    after() : timer()
    {
        int zeit2 =
            Integer.parseInt( formatter.format( new Date() ) ) - zeit;
        System.out.println( zeit2 );
        if( zeit2 > maxZeit)
        { maxZeitMethode = thisJoinPoint.toLongString() + ":" + zeit2; }
    }

    pointcut cleanUp(): call(void Book.write());
    after() : cleanUp()
    {
        System.out.print( "\nMethod with longest execution time:\n"
            + maxZeitMethode );
    }
}
```

Note

The code above assumes that you have the method write() introduced into your Book.java, as suggested in the chapter about Serialization. You can use any other method for that hook. Just make sure it gets called, and not too often. If you have a fast machine you might get only zeroes. Adapt any method of Book.java and enter anything that slows it down some. A big loop doing some Math (those things you never managed to understand at school are best for this) is always good.

```
call(public int Book.getPages()):0
call(public int Book.getPages()):0
call(public int Book.getPages()):0
call(public void Book.setPages(int)):0
call(public java.lang.String Book.getTitle()):0
call(public java.lang.String Book.getTitle()):0
call(public void Book.setTitle(java.lang.String)):0
0

Method with longest execution time:
call(public void Book.write()):30
```

Figure 10.2. AspecjtJ Times your Methods.

Though AOP is far from beeing a mainstream technology, this chapter should have given you a idea what it is about and how usefull it can be as an addition for your running projects.

Annotations [*add annotation*]

Chapter 11. Ant for Compilation and Distribution

This chapter will step through the process of making a code delivery. I use the code from the examples above and show you all the steps necessary to produce a war-file that can be deployed in the target system.

Use Ant to automate the delivery process for software.
Annotations [*add annotation*]

11.1. Installation of ANT

Download the binary version of ANT.

```
C:\Prog>ant -version
Ant version 1.4.1 compiled on October 11 2001
C:\Prog>
```

Extract the download (for me c:\jdk\ant)and add the following to you environment:

ANT_HOME = c:\jdk\ant
PATH = c:\jdk\ant\bin

That should be all. Make sure your changes to the environment took effekt. Now type 'ant -version' in any directory. You should see a message like the one above.
Annotations [*add annotation*]

11.2. What is ANT

ANT is a very usefull tool for automation. You can create skripts which let you do things like copy files , delete them, compile or jar classes, send mails, etc. All those things are done using a XML file for configuration. I don't want to go into to much detail here, but as ANT is platform independent I used it on my local Windows Laptop to create my JAR files and then on my Linux server to deploy them.

> **Note**
>
> I wrote this chapter before some others, so I don't have all the files you have by now, if you tried all examples. This should not matter, though. You still should be able to create a working example. If you did at least the first few, that is.

Annotations [*add annotation*]

11.3. Configuration of ANT

ANT is configured using a XML file. Create a file called build.xml in your root

directory of the web application.

```
build.xml for ANT

        <project name="MyProject" default="dist" basedir=".">
          <!-- set global properties for this build -->
          <property name="src" value="WEB-INF/classes/"/>
          <property name="build" value="build"/>

          <target name="init">
            <!-- Create the time stamp -->
            <tstamp/>
            <!-- Create the build directory structure used by compile -->
            <mkdir dir="${build}"/>
          </target>

          <target name="compile" depends="init">
            <!-- Compile the java code from ${src} into ${build} -->
            <javac srcdir="${src}" destdir="${build}"/>
          </target>

          <target name="dist" depends="compile">
            <!-- Create the distribution directory -->
            <mkdir dir="${build}/lib"/>

            <jar jarfile="StrutShop-${DSTAMP}.jar">
              <fileset dir="${build}" />
            </jar>
          </target>

        </project>
```

The XML file should be quite self explaining. Take a look at the ANT documentation, if not. There is a part about 'buil-in tasks', which does a very good job at explaining things.

Annotations [*add annotation*]

11.4. Usage of ANT

Now in the same directory as the build.xml type 'ant'.

Figure 11.1. ANT doesn't know Struts

That was clearly not a good idea. The problem is, of course, that ANT doesn't have Struts in it's CLASSPATH. Change the compilation command to:

```
<javac
    srcdir="${src}"
    destdir="${build}"
    classpath="WEB-INF/lib/struts.jar"
/>
```

If you did all the chapters above, it still doesn't compile, though. That is because of the Chapter 10, *Introducing AspectJ*AspectJ classes, which are not really classes. We can exclude those, though.

```
<javac
  srcdir="${src}"
  destdir="${build}"
  classpath="WEB-INF/lib/struts.jar"
  excludes="Logger.java, Timer.java"
/>
```

We now have a JAR file containing our classes. What we want is a WAR file, though. Including the JSP pages and our Struts functionallity from above. Change the build.xml again:

```
build.xml for creation of WAR file

    <project name="MyProject" default="dist" basedir=".">

      <!-- set global properties for this build -->
      <property name="src" value="WEB-INF/classes/"/>
      <property name="build" value="build"/>

      <target name="init">
        <!-- Create the time stamp -->
        <tstamp/>
        <!-- Create the build directory structure used by compile -->
        <mkdir dir="${build}"/>
      </target>

      <target name="compile" depends="init">
        <!-- Compile the java code from ${src} into ${build} -->
        <javac
            srcdir="${src}"
            destdir="${build}"
            classpath="WEB-INF/lib/struts.jar"
            excludes="Logger.java, Timer.java"
        />
      </target>

      <target name="dist" depends="compile">
        <!-- Create the distribution directory -->
        <echo message="Starting JAR" />
            <mkdir dir="dist"/>
            <jar jarfile="dist/StrutShop-${DSTAMP}.jar">
              <fileset dir="${build}" />
            </jar>

        <echo message="Starting WAR" />
        <war warfile="myapp.war" webxml="WEB-INF/web.xml">
          <fileset dir=".">
            <include name="*.jsp"/>
            <exclude name="*.xml"/>
          </fileset>
          <zipfileset dir="WEB-INF/classes"  prefix="WEB-INF/classes">
            <include name="*.properties"/>
          </zipfileset>

          <zipfileset  dir="WEB-INF" prefix="WEB-INF">
            <include name="*.xml"/>
            <include name="*.tld"/>
            <exclude name="web.xml"/>
          </zipfileset >
          <lib dir="WEB-INF/lib" />
          <classes dir="${build}"/>
        </war>
      </target>
    </project>
```
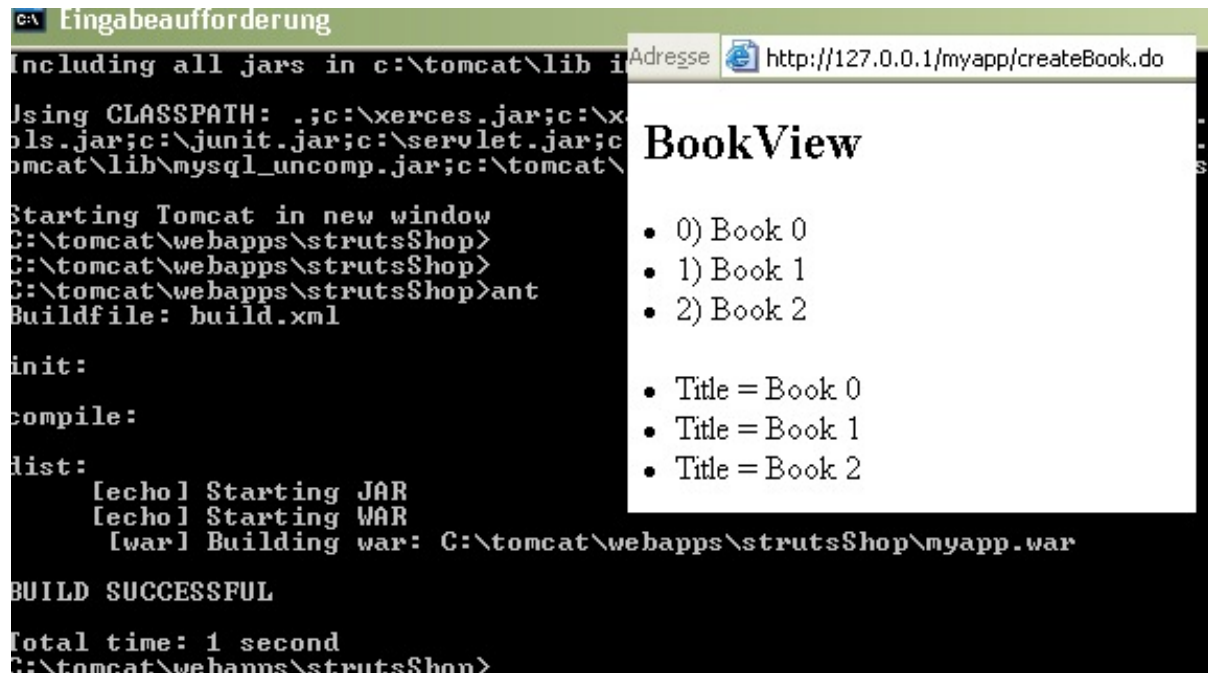
Now, take the (hopefully) generated myapp.war file and copy it into the *TOM-CAT_HOME/webapps* directory. Restart Tomcat and the needed directory structure should be generated, including all files.

Figure 11.2. Voila, application running from WAR

Note

A somewhat different file structure would have made our live somewhat easier. But, as you can test yourself, it does work. FOr a real world application with some more files, more thoughts would have to be spent on this toppic. Mind that the JAR file is created but not used again.

Question: Using the JAR

Exercise: Change the build.xml so that the JAR file is used in place of the class files.

We learned how usefull a tool ANT can be and used it to create a single WAR file for our web application.
Annotations [*add annotation*]

Chapter 12. Conclusions

Recapitulate what you (should) have learned and say a word or two about the pros and contras of using Struts.

> Divide and conquer is a nice way of solving the problem and making the problem manageable. Of course, the sword cuts both ways. The problem is more complex and needs more management.
>
> —Malcolm G. Davis

At the company I work we are usually doing both design and programming and we wouldn't need a seperation such as with Struts. I don't say we couldn't improve our work with it, though...

Annotations [*add annotation*]

Chapter 13. Large Scale Example

Here I present an example application written with Struts and EJBs. This is the same example as can be found in my EJB tutorial on my webpage. The software is a simple webshop featuring Books and CDs. It consists of a server part written using EJBs and ready to be deployed with the deploytool and the J2EE server from sun as described in my tutorial. The other part is the client, which is a standard webapplication, ready to run on Tomcat 4.x (I developed it on Tomcat 4.1.9).

I developed the shop for a course at my university in the summer of 2002. It was my first project with EJBs, so there are a lot of things I would do different on a second try (how comes I always say this? :-). I will neither comment on it, nor publish any user suggested improvements. It's there to see the technology applied and to learn something from it. It is not supposed to be an example of good programming.

The client was developed in a way that the user can switch his language between german and english (internationalisation). When he registers as a new customer, Struts is used to handle the validation of his data. I did not use Struts for iterations, as I don't see the use in such a technology for my personal use. I happen to be able to handle a 'for' loop. I fooled around some with Struts for templates but in the end I used good old includes.

The server consists of two beans. A customer and an article. The customer handles the registration and identification of a new user. I did not explicitly implement any behaviour to change his data, but this is possible. The article consists of books and CDs and the some Session Beans to handle the creation of books and a shopping basket, which is stored on the server.
Annotations [*add annotation*]

Chapter 14. FAQ

Abstract

This chapter presents some of the questions mailed to me and I try to provide an answer. Before mailing me your question, please take a look at those presented here.

Q:.

Can not find class Book

Stephan: With Tomcat 4.x many of my examples don't work anymore. The Book.class can not be found.

A:.

I suppose, I will really have to rework the examples .-(). You must create a package, put the classes inside it and import them.

Q:.

Difference of binary distribution from source code distribution? XML parser.

Jenny Pearl: Hi Stephan, I don't know where to ask this question: What is the difference of binary distribution from source code distribution? from library distribution? What should I download to start Struts? Also, it says in the jakarta-struts site, two of the pre-requisite softwares needed are XML parser and Xalan XSLT Processor. When do i use these? Do I really need these?

A:.

Yes, you have to download (for example) Xerces from xml.apache.org and add it to your CLASSPATH environment variable.

Q:.

My struts-config.xml file is not getting well formed.

Mahesh: I'm facing following problems. 1) My struts-config.xml file is not getting well formed. The error is that,The system is unable to find the path http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd but as soon as I connected to net it became well formed. Ineed help regarding the same.

A:.

Change the DTD reference from PUBLIC to SYSTEM (and point to the corressponding file) or set STANDALONE = TRUE

Q:.

Missing message for key index.title.

Mahesh: I'm facing following problems. 2) My one more exception is as follows == javax.servlet.jsp.JspException: Missing message for key index.title

A:.

Your application finds the properties file, but not the entry. Most propably a typo.

Q:.

package javax.servlet.http does not exist import javax.servlet.http.*

John: From the tutorial, I can compile the bean - Book.java. However, I get compile errors when I try to compile BookAction.java. The compile error is: BookAction.java:1: package javax.servlet.http does not exist import javax.servlet.http.*; [Q] Why am I not finding javax.servlet.http?

A:.

The Servlet API is an add on to Java. You will have it in Tomcat, but your application (Servlet) has to know that too, in order to be compiled. Find the JAR and add it to the CLASSPATH.

Q:.

Will the price increase

Stephan Wiesner Will the price for this tutorial increase? Will it become a book?

A:.

Yes, it might become a real book with an ISBN and all (that is what my diploma thesis is about, after all), but this decision is not yet final and until then the price will stay at a single, symbolic Euro.

Annotations [*add annotation*]

Chapter 15. Epilogue

Annotations [*add annotation*]

15.1. Technical Background of this Document

This document is written in XML. I started using a DTD and a document management system that developed in a project at the University of Applied Sciences in Lüneburg, Germany. I changed to DocBook after some time, though.

Yes, that means that I am sitting here, writing dozens of pages in 'native XML' using just a normal text editor (www.JEdit.org) some little tools I wrote.

The HTML version is transformed with Xalan (xml.apache.org) and an adapted version of the CSS style from http://www.e-novative.de is applied to it.
The PDF version is created with FOP (xml.apache.org/fop).
The HTML-Help version (MS Windows only) is created with the default style from DocBook and the Microsoft HTML Help tool ().

At the moment I am writing my diploma thesis with the (much too long) title *Implementing a System for the Publication and Distribution of Copyrighted Digital Books*. I suppose, this means that this document might turn into a real book, with ISBN and all. I didn't decide on this, yet, though.

Annotations [*add annotation*]

15.2. About the Author

Stephan Wiesner is currently studying computer science at the University of Applied Science in Lüneburg, Germany. He is working part time at Werum AG in Lüneburg, where he is programming content management systems.

Stephan thinks himself quite adapt in PHP and the Java programming language, has experience in the field of web page development and management since 1998, and 'tons of experience' (his quote) with XML and XSLT. He is an active developer in different open source projects hostet at sourceforge.net.

Figure 15.1. Stephan Wiesner

You can visit his homepage at www.stephanwiesner.de . If you think this tutorial helped you and is worth about 1$, please write a postcard from your hometown, like the ones in Section 16.2, "Post Cards" below. Just to satisfy the curiosity of the author . . .

```
Stephan Wiesner
Höpenweg 7
21357 Wittorf
Germany
```

Annotations [*add annotation*]

Chapter 16. User Comments

Annotations [*add annotation*]

16.1. Mails

This chapter contains some mails I got from users. The collection is just a random choice. My thanks go to everyone who wrote me.

Hello Stephan, I want to thank you for your excellent Struts tutorial. I am developing an application for the University of California, San Francisco and I have been trying for two weeks to understand Struts. I have worked through numerous tutorials only to become lost and frustrated. Your simple example and lucid instructions were exactly what I needed to learn the basic concepts of Forms and Actions. It is 1:07 AM here and I have just successfully completed my first database transaction using Struts and Torque. If you are ever in San Francisco, I owe you a beer! Thanks again, Chris

BTW, YOUR TUTORIAL RULES!!! Really very good and useful!!! Thanks, Daniel Novy.

Your Struts tutorial has proved immensely helpful - thank you very much for putting it together - it is much appreciated --- Regards, - Chris

Hello Stephan, I just wanted to thank you for publishing the 'Struts Tutorial.' I have found it immensely helpful. Keep up the great work. Sincerely, Claude Rizzo California

Hello. My name is Kazui Fujii lives in Tokyo, Japan. I have read the Struts Tutorial you wrote. I must use the Struts framework on my work. However I could not understand the Struts framework's concepts thoroughly. Therefore I had look for the document written about the Struts framework. At that time I found the Struts tutorial you wrote!!! This document give me a good understanding of the Struts framework. In particular creating a application step by step is very good! I could understand what I must do on the Struts framework very well.

Dear Stephan Wiesner, This is Hong Wang. I am following your struts tutorial using Tomcat 4.1.7 and Struts 1.1(i think). I am at Chapter 5.Struts for Forms. I have compiled the Book and BookAction classes in \strutsShop\WEB-INF\classes. i have modified the struts-config.xml and created CreateBook.jsp. but i still got following error. 'Cannot find ActionMappings or ActionFormBeans colloection.' could you please help sort it? Thank you very much. Yours sincerely, Hong Wang [This mail made me publish the source-code]

Well, here I am one last time Stephan. :) [we exchanged some mails] You must be pretty annoyed by now ;), [well, ... :-)] but this mail is just to let you know that I finished your tutorial. I must admit that I am not really convinced yet of the advantages struts can offer, apart from the better seperation between design and coding. I think I will need to learn more about the "tags" that can be used. Perhaps this will give me more insight in the usefulness. I am going to try and build a small application with the help of struts, and see how the actionForm

<-> actionBean can add me in developing. Thank you very much for your tutorial and input. Best regards, J.C. van der Made The Netherlands
Annotations [*add annotation*]

16.2. Post Cards

Figure 16.1. One World, one Tutorial :-)

Balog from Budapest was the first to send me a post card. As the first time is always special, I was very happy about it. In fact I even laughed aloud. Thanks Balog.

Figure 16.2. Balog from Budapest

Luca send me this picture of the silent watchers of the beautiful city of Paris.

Figure 16.3. Luca from Paris

Puri Sumeet from Singapore send me this postcard with the Fountain of Wealth, he worlds largest fountain, on it..

Figure 16.4. Sumeer from Singapore

Annotations [*add annotation*]

Appendix A. Glossary

Glossary

DocBook

> DocBook is a very popular set of tags for describing books, articles, and other prose documents, particularly technical documentation. DocBook is defined using the native DTD syntax of SGML and XML. Like HTML, DocBook is an example of a markup language defined in SGML/XML. [...]
>
> DocBook is almost 10 years old. It began in 1991 as a joint project of HaL Computer Systems and O'Reilly. Its popularity grew, and eventually it spawned its own maintainance organization, the Davenport Group. In mid-1998, it became a Technical Committee (TC) of the Organization for the Advancement of Structured Information Standards (OASIS). (Taken from www.docbook.org)

Extensible Markup Language

> XML is a platform and programming language idependend standard for formating data. See www.w3c.org for more details.

Model View Controller

> Model View Controller

Extensible Stylesheet Language

> Extensible Stylesheet Language. See www.w3c.org for more details.

Appendix B. Solutions to Exercises

This section presents extended solutions to some of the exercises.

B.1. CDs

The code presented here corresponds to the exercise Q: . Please pay attention to the package 'cds'. The code was contributed by Marcus Biel and slightly adapted by me.

```
     Cd.java

package cds;
import java.util.Vector;

public class Cd
{
    private String title = "";
    private int playtime = 0;

    public void setTitle(String title){
     this.title = title;
    }

    public String getTitle(){
     return this.title;
    }

    public void setPlaytime(int playtime) {
     this.playtime = playtime;
    }

    public int getPlaytime() {
     return this.playtime;
    }
}
```

```
     CdAction.java

package cds;

import javax.servlet.http.*;
import org.apache.struts.action.*;

public final class CdAction extends Action {

      public ActionForward perform(ActionMapping mapping,
   ActionForm form, HttpServletRequest request, HttpServletResponse response)
   {
      System.out.println("Start perform(" + form + ") . . ." );
          return mapping.findForward("cdCreated");
       }
}
```

```
     CdForm.java

package cds;

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.*;

public class CdForm extends ActionForm {

      private Cd cd = new Cd();

      public void setCd(Cd cd) { this.cd= cd; }

      public Cd getCd() { return this.cd;           }

      public void reset(ActionMapping mapping, HttpServletRequest request)
   { this.cd= new Cd(); }
```

```
        public ActionErrors validate(ActionMapping mapping,
        HttpServletRequest request)
   {
                ActionErrors errors= new ActionErrors();
                if( (cd.getTitle()==null || cd.getTitle().length() <3) ) {
                        errors.add("title", new ActionError("error.cd.title") );
                }

                if(cd.getPlaytime() <1) {
                        errors.add("playtime", new ActionError("error.cd.playtime") );
                }
                return errors;
        }
}
```

```
    CdView.jsp

<%@ page import="cds.*" %>

<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean"%>
<%@ taglib uri="/WEB-INF/struts-html.tld" prefix="html"%>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic"%>

<html:html locale="true">
        <head>
                <html:base/>
                <title>
                        <bean:message key="index.title"/>
                </title>
        </head>

        <body bgcolor="white">
                <h2>Cd View</h2>

                <html:form action="createCd">
                    Title:<html:text property="cd.title" /><br/>
                    Playtime:<html:text property="cd.playtime"/><br/>
                    <html:submit property="submit"/>
                </html:form>

        </body>
</html:html>
```

```
    CreateCd.jsp

<%@ page import="cds.*" %>

<%@ page language="java" %>
<%@ taglib uri="/WEB-INF/struts-bean.tld"  prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-html.tld"  prefix="html" %>
<%@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>

<html:html locale="true">
    <head>
      <html:base/>
      <title><bean:message key="index.title"/></title>
    </head>

    <body bgcolor="white">
          <h2>Create a Cd</h2>

    <html:errors property="title"/>
    <html:errors property="playtime"/>

    <html:form action="createCd.do" method="GET">
                    Title:<html:text property="cd.title"/> <br/>
                  Playtime:<html:text property="cd.playtime"/> <br/>
                  <html:submit property="submit"/>
                </html:form>
        </body>
</html:html>
```

```
    struts-config.xml

<?xml version="1.0" encoding="ISO-8859-1" ?>

 <!DOCTYPE struts-config PUBLIC
```

58

```
    "-//Apache Software Foundation//DTD Struts Configuration 1.0//EN"
    "http://jakarta.apache.org/struts/dtds/struts-config_1_0.dtd">

 <struts-config>
   <form-beans>
     <form-bean name="cdForm" type="cds.CdForm"/>
      <form-bean name="bookForm" type="books.BookForm"/>
   </form-beans>

   <global-forwards>
     <forward name="cdCreated" path="/CdView.jsp"/>
     <forward name="bookCreated" path="/BookView.jsp"/>
   </global-forwards>

   <action-mappings>

     <action path="/createBook"
        type="books.BookAction"
        name="bookForm"
        scope="request"
        input="/CreateBook.jsp">
     </action>

     <action     path="/createCd"
       type="cds.CdAction"
       name="cdForm"
       scope="request"
       input="/CreateCd.jsp">
     </action>
   </action-mappings>
 </struts-config>
```
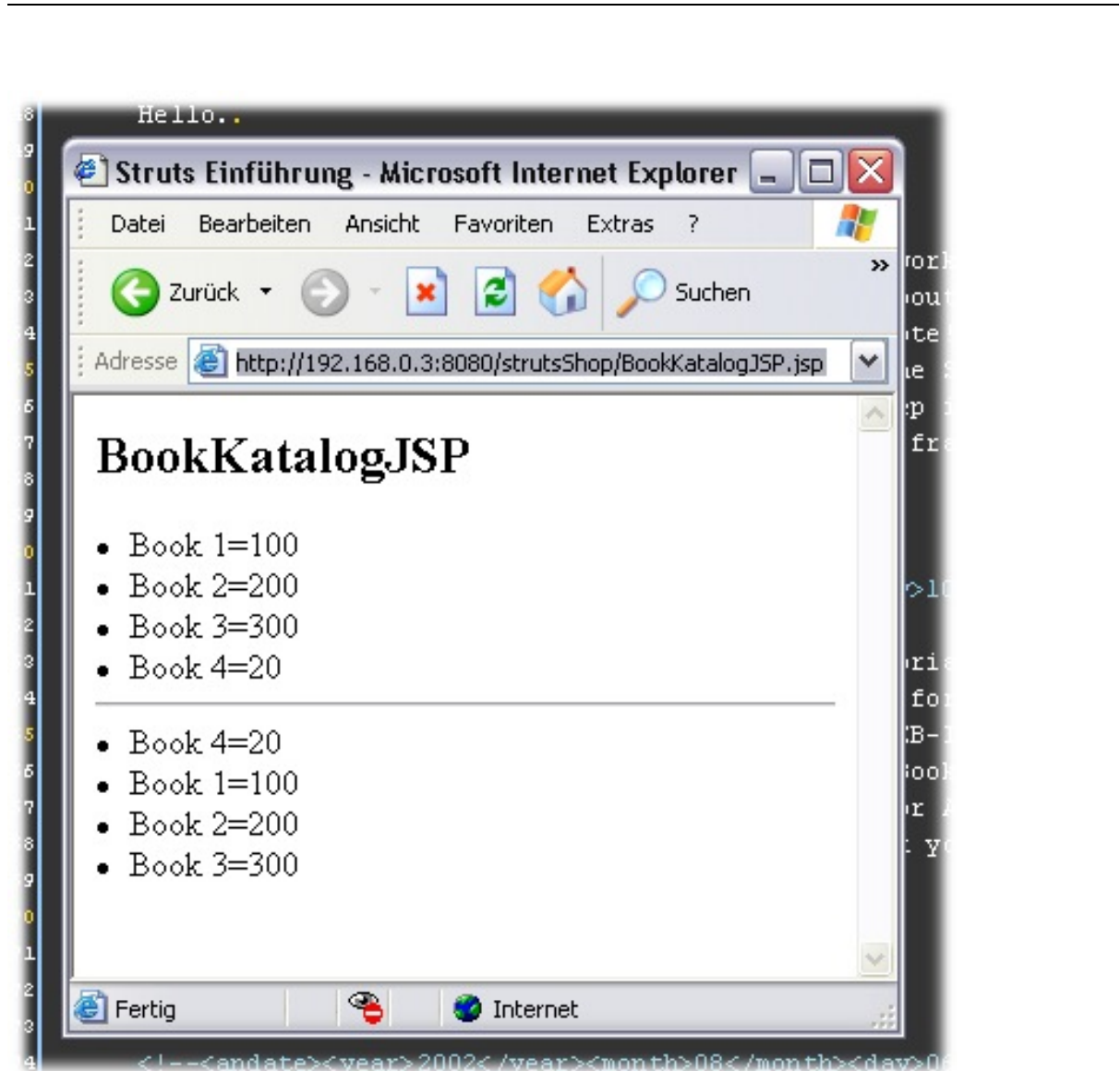
Annotations [*add annotation*]

B.2. Mixed Solutions

Figure B.1. Sorted List of Books

See Section 6.2, "Advanced Logic" for the corresponding text.

Annotations [*add annotation*]

Appendix C. Bibliography

Bibliography

Books

[DocBook] Walsh, Norman. Leonard Muellner. *DocBook: The Definitive Guide*. O'Reilly. USA. ISBN 156592-580-7.

Take a look at www.docbook.org, as the book is quite outdated by now.

[GoF] Gamma, Helm. Johnson. Vlissides. *Design Patterns*. Addison-Wesley. Indianapolis USA . 1995. ISBN 0-201-63361-2.

The standard book about Design Patterns.

Weblinks

[AspectJ] *The AspectJ Homepage*.

[bibApacheXML]

[bibSunXML]

[Jakarta] *The Apache Jakarta Homepage*.

[JavaMagazin] *Javamagazin 04.2002*.

[Malcom] *Malcolm G. Davis*.

[Struts] *The Struts Homepage*.

[SW] *My Own Home Page*.