



Overview Package Class Use [Tree](#) [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#)

Packages

[bookCatalog](#)

Overview Package Class Use [Tree](#) [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#)

[Overview](#) [Package](#) **Class** [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#)

SUMMARY: INNER | FIELD | CONSTR | [METHOD](#)

DETAIL: FIELD | CONSTR | [METHOD](#)

bookCatalog

Class BookListTagExtraInfo

|
+--bookCatalog.BookListTagExtraInfo

public class **BookListTagExtraInfo**
extends TagExtraInfo

A helper class for BookListTag. It's single method, `getVariableInfo(TagData data)` returns an array of VariableInfo objects. Each of these objects describes one of the variables set by the custom tag.

Author:
Dr. Jim Arlow

Method Summary

VariableInfo[]	getVariableInfo (TagData data) Returns an array of VariableInfo objects.
----------------	---

Method Detail

getVariableInfo

public VariableInfo[] **getVariableInfo**(TagData data)

Returns an array of VariableInfo objects. Each of these objects describes one of the variables created by the custom tag.

Dependency Links

to Class [bookCatalog.BookListTag](#)

link dependency

[Overview](#) [Package](#) **Class** [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#)

SUMMARY: INNER | FIELD | CONSTR | [METHOD](#)

DETAIL: FIELD | CONSTR | [METHOD](#)

bookCatalog

Class BookListTag

|

+--bookCatalog.BookListTag

public class **BookListTag**
extends BodyTagSupport

This custom tag is used for displaying lists of books from the book database.

It executes a query on the book database and then iterates over the set of returned rows. For each row, it places each field of the row in a page scope variable with the same name as the field. These variables are accessible directly from the JSP that uses the tag.

If data is returned by the query, the page scope variable dataAvaiable is set to false. Otherwise it is set to true.

Author:
Dr. Jim Arlow

Version: 1.1

Method Summary	
int	doAfterBody () Puts the data from the current row of the result set into the page context variables isbn, title, authors, publisher, rating, review, reviewer, pictureurl, category.
int	doEndTag () Writes the body content of the tag.

int	<u>doStartTag</u> () Opens a connection to the book database.
void	<u>setNoDataPageContextVariables</u> () Sets the page scope attribute dataAvailable to "true"
void	<u>setPageContextVariables</u> () Sets the following page scope attributes with the data from the current row of the result set.

Method Detail

doAfterBody

```
public int doAfterBody()  
           throws JspException
```

Puts the data from the current row of the result set into the page context variables isbn, title, authors, publisher, rating, review, reviewer, pictureurl, category.

Preconditions

The method doStartTag() has succeeded.

Postconditions

The page context variables isbn, title, authors, publisher, rating, review, reviewer, pictureurl, category have been set with the data from the current row of the result set.

doEndTag

```
public int doEndTag()  
           throws JspException
```

Writes the body content of the tag. Closes the statement and connection.

Preconditions None

Postconditions

- The statement is closed.
- The connection is closed.

doStartTag

```
public int doStartTag()  
    throws JspException
```

Opens a connection to the book database. Executes the query stored in the page scope attribute "query" to obtain a result set.

\n

Iterates over the result set passing each row in turn to doAfterBody()

Preconditions

There is a valid query in the page scope attribute called "query".

Postconditions

- A connection to the database is open.
- A statement is open.

setNoDataPageContextVariables

```
public void setNoDataPageContextVariables()  
    throws java.sql.SQLException
```

Sets the page scope attribute dataAvailable to "true"

Preconditions None

Postconditions The page scope attribute called dataAvailable is set to "true".

setPageContextVariables

```
public void setPageContextVariables()  
    throws java.sql.SQLException
```

Sets the following page scope attributes with the data from the current row of the result set.

\n

\n

- isbn

- \n
- title
- \n
- authors
- \n
- publisher
- \n
- rating
- \n
- review
- \n
- reviewer
- \n
- pictureurl
- \n
- category
- \n

Preconditions

A result set has been returned by the query.

Postconditions

The data from the current row of the result set has been placed in page scope variables with the same name.

Association Links

to **Class** java.sql.ResultSet

A collection of data returned by a query on the database.

to **Class** java.sql.Statement

The SQL statement that will generate the result set.

to **Class** java.sql.Connection

The database connection.

to **Class** java.lang.String

The query to be executed on the database. This must return a result set containing zero or more rows where each row contains every column in the database.

to **Class** java.lang.String

An array of graphics files showing one to five stars.

[Overview](#) [Package](#) **Class** [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#)

SUMMARY: INNER | FIELD | CONSTR | [METHOD](#)

DETAIL: FIELD | CONSTR | [METHOD](#)

bookCatalog

Class BookDAO

class **BookDAO**

This class realizes the Data Access Object pattern described in J2EE Patterns, ISBN0130648841.

Objects of this class represent a single row in the book database.

Author:
Dr. Jim Arlow

Constructor Summary	
BookDAO ()	The default constructor for the BookDAO class.
BookDAO (HttpServletRequest req)	This constructor reads parameters from the HttpServletRequest and stores them in private attributes of the BookDAO object.
BookDAO (String targetIsbn)	This constructor takes a book isbn as a parameter.

Method Summary	
static void	delete (String isbnToDelete) This method makes a connection to the database and deletes the row specified by the isbn parameter.
String	getAuthors ()
String	getCategory ()

String	<u>getIsbn</u> ()
String	<u>getPictureurl</u> ()
String	<u>getPublisher</u> ()
String	<u>getRating</u> ()
String	<u>getReview</u> ()
String	<u>getReviewer</u> ()
String	<u>getTitle</u> ()
void	<u>setAuthors</u> (String authors)
void	<u>setCategory</u> (String category)
void	<u>setIsbn</u> (String isbn)
void	<u>setPictureurl</u> (String pictureurl)
void	<u>setPublisher</u> (String publisher)
void	<u>setRating</u> (String rating)
void	<u>setReview</u> (String review)
void	<u>setReviewer</u> (String reviewer)
void	<u>setTitle</u> (String title)
void	<u>store</u> () <p>This method opens a connection to the database and adds a new book record based on the information stored in the BookDAO object's private attributes.</p>

String	toHTML () Returns a String which is an HTML representation of the BookDAO object.
String	toString () Returns a String representation of the BookDAO object.

Constructor Detail

BookDAO

```
public BookDAO()
```

The default constructor for the BookDAO class.

BookDAO

```
public BookDAO(HttpServletRequest req)  
    throws ServletException
```

This constructor reads parameters from the HttpServletRequest and stores them in private attributes of the BookDAO object.

- isbn
- title
- authors
- publisher
- rating
- review
- reviewer
- picuterurl
- category

Preconditions

The HttpServletRequest must contain the following parameters:

Postconditions

The BookDAO object has copied the HttpServletRequest parameters into private attributes.

BookDAO

```
public BookDAO(String targetIsbn)
    throws ServletException
```

This constructor takes a book isbn as a parameter. It opens a connection to the database, and reads the row corresponding to the isbn. The information from the row is stored in the private attributes of the BookDAO object.

Preconditions

The isbn must refer to a row in the database.

Postconditions

The private attributes of the BookDAO object contain book details from the appropriate row of the database.

Method Detail

delete

```
public static void delete(String isbnToDelete)
    throws ServletException
```

This method makes a connection to the database and deletes the row specified by the isbn parameter.

Preconditions

The isbn parameter corresponds to a row in the database.

Postconditions

A row has been deleted from the database.

getAuthors

```
public String getAuthors()
```

getCategory

```
public String getCategory()
```

getIsbn

```
public String getIsbn()
```

getPictureurl

```
public String getPictureurl()
```

getPublisher

```
public String getPublisher()
```

getRating

```
public String getRating()
```

getReview

```
public String getReview()
```

getReviewer

```
public String getReviewer()
```

getTitle

```
public String getTitle()
```

setAuthors

```
public void setAuthors(String authors)
```

setCategory

```
public void setCategory(String category)
```

setIsbn

```
public void setIsbn(String isbn)
```

setPictureurl

```
public void setPictureurl(String pictureurl)
```

setPublisher

```
public void setPublisher(String publisher)
```

setRating

```
public void setRating(String rating)
```

setReview

```
public void setReview(String review)
```

setReviewer

```
public void setReviewer(String reviewer)
```

setTitle

```
public void setTitle(String title)
```

store

```
public void store()  
    throws ServletException
```

This method opens a connection to the database and adds a new book record based on the information stored in the BookDAO object's private attributes.

Preconditions

The BookDOA object contains valid book information stored in its private attributes.

Postconditions

A row in the database has been created or updated.

toHTML

```
public String toHTML()
```

Returns a String which is an HTML representation of the BookDAO object. Each private attribute is given its own row in an HTML table.

This method is only intended for testing and debugging.

Preconditions None

Postconditions A String is returned that contains all of the information in the BookDAO object formatted as an HTML table.

toString

```
public String toString()
```

Returns a String representation of the BookDAO object.

Preconditions None.

Postconditions A String is returned that contains al

l of the information in the BookDAO object.

Association Links

to **Class** java.lang.String

The username for log on to the database.

to **Class** java.lang.String

The URL of the database driver.

to **Class** java.lang.String

The URL to the database itself.

to **Class** java.lang.String

The password for the database.

to **Class** java.lang.String

The ISBN of the book.

to **Class** java.lang.String

The title of the book.

to **Class** java.lang.String

The authors of the book. Author names should be in a comma delimited list.

to **Class** java.lang.String

The publisher of the book.

to **Class** java.lang.String

The rating of the book. This is a number between 1 (lowest) and 5 (highest) that indicates the reviewers opinion of the book.

to **Class** java.lang.String

The review of the book by the reviewer.

to **Class** java.lang.String

The name of the reviewer of the book.

to **Class** java.lang.String

The URL of the catalog picture of the book.

to **Class** java.lang.String

The category of the book.

[Overview](#) [Package](#) **Class** [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#)

SUMMARY: INNER | FIELD | [CONSTR](#) | [METHOD](#)

DETAIL: FIELD | [CONSTR](#) | [METHOD](#)

bookCatalog

Class AddBookServlet

|
+--bookCatalog.AddBookServlet

class **AddBookServlet**
extends HttpServlet

This class adds a new book row to the book database. The data for the book comes from request parameters:

- isbn
- title
- authors
- publisher
- rating
- review
- reviewer
- pictureurl
- category

Author:
Dr. Jim Arlow

Method Summary

void	doPost (HttpServletRequest req, HttpServletResponse res) Process an HTTP POST request.
------	---

Method Detail

doPost

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException,
        java.io.IOException
```

Process an HTTP POST request. The BookDAO object reads book data from parameters in the HttpServletRequest and stores it internally. Finally, the BookDAO stores itself in the database.

Dependency Links

to **Class** [bookCatalog.BookDAO](#)

link dependency

Overview	Package	Class	Use	Tree	Index	Help
PREV CLASS	NEXT CLASS		FRAMES	NO FRAMES		
SUMMARY: INNER FIELD CONSTR METHOD			DETAIL: FIELD CONSTR METHOD			

bookCatalog

Class BookCatalogServlet

|
+--bookCatalog.BookCatalogServlet

public class **BookCatalogServlet**
extends HttpServlet

This class realises the Front Controller pattern described in J2EE Patterns, ISBN0130648841.

Http requests sent to this servlet are routed to either AddBookServlet or DeleteBookServlet

Author:
Dr. Jim Arlow

Method Summary

void	doPost (HttpServletRequest req, HttpServletResponse res)
------	--

Method Detail

doPost

```
public void doPost(HttpServletRequest req, HttpServletResponse res)  
    throws ServletException,  
        java.io.IOException
```

Dependency Links

to Class [bookCatalog.AddBookServlet](#)

link dependency

to **Class** [bookCatalog.DeleteBookServlet](#)

link dependency

[Overview](#) [Package](#) **Class** [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#)

SUMMARY: INNER | FIELD | CONSTR | [METHOD](#)

DETAIL: FIELD | CONSTR | [METHOD](#)

bookCatalog

Class DeleteBookServlet

|
+--bookCatalog.DeleteBookServlet

class DeleteBookServlet
extends HttpServlet

Deletes a book from the database. The book ISBN must be in the HTTP request as a parameter called "isbn".

Author:
Dr. Jim Arlow

Method Summary

void	doPost (HttpServletRequest req, HttpServletResponse res) Gets the book ISBN from the "isbn" parameter in the HttpServletRequest.
------	---

Method Detail

doPost

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
    throws ServletException,
        java.io.IOException
```

Gets the book ISBN from the "isbn" parameter in the HttpServletRequest. Uses the static [delete](#) method of the BookDAO to delete the book row from the table.

Dependency Links

to **Class** [bookCatalog.BookDAO](#)

link dependency

[Overview](#) [Package](#) **Class** [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV CLASS](#) [NEXT CLASS](#)

[FRAMES](#) [NO FRAMES](#)

SUMMARY: [INNER](#) | [FIELD](#) | [CONSTR](#) | [METHOD](#)

DETAIL: [FIELD](#) | [CONSTR](#) | [METHOD](#)

[Overview](#) [Package](#) [Class](#) [Use](#) **Tree** [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#)

Hierarchy For All Packages

Package Hierarchies:

[bookCatalog](#)

Class Hierarchy

- class bookCatalog.[BookDAO](#)

[Overview](#) [Package](#) [Class](#) [Use](#) **Tree** [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#)

[Overview](#) [Package](#) [Class](#) [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#)

[A](#) [B](#) [C](#) [D](#) [E](#) [G](#) [H](#) [I](#) [J](#) [L](#) [M](#) [N](#) [S](#) [T](#) [U](#) [W](#)

A

[Activation1](#) - Activation in [bookDAO](#)(in Sequence Diagram [AddBookToCatalogSequence](#))

[Activation1](#) - Activation in [ListBooks.jsp](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation10](#) - Activation in [bookListTag](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation11](#) - Activation in [bookListTag](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation12](#) - Activation in [bookListTag](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation13](#) - Activation in [bookListTag](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation14](#) - Activation in [addBookServlet](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation15](#) - Activation in [newBook](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation16](#) - Activation in [newBook](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation17](#) - Activation in [deleteBookServlet](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation18](#) - Activation in [existingBook](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation2](#) - Activation in [AddNewBook.htm](#)(in Sequence Diagram [AddBookToCatalogSequence](#))

[Activation2](#) - Activation in [bookListTag](#)(in Sequence Diagram

[UpdateOrDeleteBookFromCatalogSequence](#))

[Activation3](#) - Activation in [bookListTag](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation4](#) - Activation in [bookCatalogServlet](#)(in Sequence Diagram [AddBookToCatalogSequence](#))

[Activation4](#) - Activation in [bookListTag](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation5](#) - Activation in [addBookServlet](#)(in Sequence Diagram [AddBookToCatalogSequence](#))

[Activation5](#) - Activation in [bookListTag](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation6](#) - Activation in [bookDAO](#)(in Sequence Diagram [AddBookToCatalogSequence](#))

[Activation6](#) - Activation in [EditBook.jsp](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation7](#) - Activation in [bookCatalogServlet](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation8](#) - Activation in [ListBooks.jsp](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[Activation9](#) - Activation in [EditBook.jsp](#)(in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#))

[AddBookServlet](#) - class bookCatalog.[AddBookServlet](#)

This class adds a new book row to the book database.

[addBookServlet](#) - Object in Sequence Diagram [AddBookToCatalogSequence](#)

[addBookServlet](#) - Object in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[AddBookServlet.class](#) - Component in [bookCatalog](#)(in Component Diagram [Component](#))

Contains the Java bytecode for the AddBookServlet class.

[AddBookToCatalogSequence](#) - Sequence Diagram in [bookCatalog](#)

[AddNewBook.htm](#) - Object in Sequence Diagram [AddBookToCatalogSequence](#)

[AddNewBook.html](#) - Component in [administration](#)(in Component Diagram [Component](#))

This is an HTML file that contains a form used to submit book details to the

[BookCatalogServlet](#) for addition to the database.

[AddProductToCatalog](#) - UseCase in [bookCatalog Subsystem](#)(in Use Case Diagram [Book Catalog Use Case](#))

Add a new Product to the ProductCatalog.

[administration](#) - Component in [htdocs](#)(in Component Diagram [Component](#))

This directory is protected by Apache security.

[apache](#) - Component in [testServer](#)(in Deployment Diagram [ECP Deployment Model](#))

An instance of the Apache web server.

B

[bookCatalog](#) - package bookCatalog

[bookCatalog](#) - Component in [classes](#)(in Component Diagram [Component](#))

This is where we place all of the Java class files for the ECP bookCatalog subsystem.

[bookCatalog](#) - Class Diagram in [bookCatalog](#)

This package contains the design for the book catalog.

[BookCatalogServlet](#) - class bookCatalog.[BookCatalogServlet](#)

This class realises the Front Controller pattern described in J2EE Patterns, ISBN0130648841.

[bookCatalogServlet](#) - Object in Sequence Diagram [AddBookToCatalogSequence](#)

[bookCatalogServlet](#) - Object in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[BookCatalogServlet.class](#) - Component in [bookCatalog](#)(in Component Diagram [Component](#))

Contains the Java bytecode for the BookCatalogServlet.

[bookCatalog Subsystem](#) - System Boundary in Use Case Diagram [Book Catalog Use Case](#)

[Book Catalog Use Case](#) - Use Case Diagram in [bookCatalog](#)

[BookDAO](#) - class bookCatalog.[BookDAO](#)

This class realizes the Data Access Object pattern described in J2EE Patterns, ISBN0130648841.

[bookDAO](#) - Object in Sequence Diagram [AddBookToCatalogSequence](#)

[BookDAO.class](#) - Component in [bookCatalog](#)(in Component Diagram [Component](#))

Contains the Java bytecode for the BookDAO (Data Access Object) class.

[BookDAO\(\)](#) - Constructor for class bookCatalog.[BookDAO](#)

The default constructor for the BookDAO class.

[BookDAO\(HttpServletRequest\)](#) - Constructor for class bookCatalog.[BookDAO](#)

This constructor reads parameters from the HttpServletRequest and stores them in private attributes of the BookDAO object.

[**BookDAO\(java.lang.String\)**](#) - Constructor for class bookCatalog.[BookDAO](#)

This constructor takes a book isbn as a parameter.

[**BookListTag**](#) - class bookCatalog.[BookListTag](#)

This custom tag is used for displaying lists of books from the book database.

[**bookListTag**](#) - Object in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[**BookListTag.class**](#) - Component in [bookCatalog](#)(in Component Diagram [Component](#))

Contains the Java bytecode for the BookListTag tag library.

[**BookListTagExtraInfo**](#) - class bookCatalog.[BookListTagExtraInfo](#)

A helper class for BookListTag.

[**BookListTagExtraInfo.class**](#) - Component in [bookCatalog](#)(in Component Diagram [Component](#))

Contains the Java bytecode for the BookListTagExtraInfo helper class.

[**booklisttaglib.tld**](#) - Component in [META-INF](#)(in Component Diagram [Component](#))

This is the tag library descriptor for the BookListtag.

C

[**classes**](#) - Component in [WEB-INF](#)(in Component Diagram [Component](#))

The standard directory structure for J2EE web applications specifies that there should be a directory called WEB-INF/classes hanging off the root of the application.

[**Component**](#) - Component Diagram in [bookCatalog](#)

D

[**delete\(java.lang.String\)**](#) - Static method in class bookCatalog.[BookDAO](#)

This method makes a connection to the database and deletes the row specified by the isbn parameter.

[**DeleteBookServlet**](#) - class bookCatalog.[DeleteBookServlet](#)

Deletes a book from the database.

[**deleteBookServlet**](#) - Object in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[**DeleteBookServlet.class**](#) - Component in [bookCatalog](#)(in Component Diagram [Component](#))

Contains the Java bytecode for the DeleteBookServlet class.

[**DeleteProductFromCatalog**](#) - UseCase in [bookCatalog Subsystem](#)(in Use Case Diagram [Book Catalog Use Case](#))

Delete a Product from the ProductCatalog.

[**doAfterBody\(\)**](#) - Method in class bookCatalog.[BookListTag](#)

Puts the data from the current row of the result set into the page context variables isbn, title, authors, publisher, rating, review, reviewer, pictureurl, category.

[**doEndTag\(\)**](#) - Method in class bookCatalog.[BookListTag](#)

Writes the body content of the tag.

[**doPost\(HttpServletRequest,HttpServletResponse\)**](#) - Method in class bookCatalog.[AddBookServlet](#)

Process an HTTP POST request.

[**doPost\(HttpServletRequest,HttpServletResponse\)**](#) - Method in class bookCatalog.[BookCatalogServlet](#)

[**doPost\(HttpServletRequest,HttpServletResponse\)**](#) - Method in class bookCatalog.[DeleteBookServlet](#)

Gets the book ISBN from the "isbn" parameter in the HttpServletRequest.

[**doStartTag\(\)**](#) - Method in class bookCatalog.[BookListTag](#)

Opens a connection to the book database.

E

[**ECP Deployment Model**](#) - Deployment Diagram in [bookCatalog](#)

[**EditBook.jsp**](#) - Component in [administration](#)(in Component Diagram [Component](#))

This is JSP that presents the user with a form cotnaining book details for editing.

[**EditBook.jsp**](#) - Object in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[**existingBook**](#) - Object in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

G

[**getAuthors\(\)**](#) - Method in class bookCatalog.[BookDAO](#)

[**getCategory\(\)**](#) - Method in class bookCatalog.[BookDAO](#)

[**getIsbn\(\)**](#) - Method in class bookCatalog.[BookDAO](#)

[**getPictureurl\(\)**](#) - Method in class bookCatalog.[BookDAO](#)

[**getPublisher\(\)**](#) - Method in class bookCatalog.[BookDAO](#)

[getRating\(\)](#) - Method in class bookCatalog.[BookDAO](#)

[getReview\(\)](#) - Method in class bookCatalog.[BookDAO](#)

[getReviewer\(\)](#) - Method in class bookCatalog.[BookDAO](#)

[getTitle\(\)](#) - Method in class bookCatalog.[BookDAO](#)

[getVariableInfo\(TagData\)](#) - Method in class bookCatalog.[BookListTagExtraInfo](#)

Returns an array of VariableInfo objects.

H

[htdocs](#) - Component in Component Diagram [Component](#)

The Tomcat installation for the ECP requires all HTML and JSP files to be placed in a directory called htdocs, or in one of it's subdirectories.

[htdocsDirectory](#) - Component in [testServer](#)(in Deployment Diagram [ECP Deployment Model](#))

The root directory for our web application.

I

[ie5](#) - Component in [jimsPC](#)(in Deployment Diagram [ECP Deployment Model](#))

An instance of Microsoft Internet Explorer 5.

J

[jimsPC](#) - Node in Deployment Diagram [ECP Deployment Model](#)

The test PC on Jims desk.

L

[ListBooks.jsp](#) - Component in [administration](#)(in Component Diagram [Component](#))

This is a JSP that displays summary information for a list of books.

[ListBooks.jsp](#) - Object in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

M

[META-INF](#) - Component in [htdocs](#)(in Component Diagram [Component](#))

In a J2EE web application, tag library descriptors (XML files that describe the contents of a custom JSP tag library) are usually placed in a subdirectory off the web application root called META-INF.

N

[newBook](#) - Object in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[Note1](#) - Note in Sequence Diagram [AddBookToCatalogSequence](#)

[Note1](#) - Note in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[Note1](#) - Note in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[Note2](#) - Note in Sequence Diagram [AddBookToCatalogSequence](#)

[Note3](#) - Note in Sequence Diagram [AddBookToCatalogSequence](#)

[Note3](#) - Note in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[Note4](#) - Note in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[Note5](#) - Note in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

S

[setAuthors\(java.lang.String\)](#) - Method in class bookCatalog.[BookDAO](#)

[setCategory\(java.lang.String\)](#) - Method in class bookCatalog.[BookDAO](#)

[setIsbn\(java.lang.String\)](#) - Method in class bookCatalog.[BookDAO](#)

[setNoDataPageContextVariables\(\)](#) - Method in class bookCatalog.[BookListTag](#)

Sets the page scope attribute dataAvailable to "true"

[setPageContextVariables\(\)](#) - Method in class bookCatalog.[BookListTag](#)

Sets the following page scope attributes with the data from the current row of the result set.

[setPictureurl\(java.lang.String\)](#) - Method in class bookCatalog.[BookDAO](#)

[setPublisher\(java.lang.String\)](#) - Method in class bookCatalog.[BookDAO](#)

[setRating\(java.lang.String\)](#) - Method in class bookCatalog.[BookDAO](#)

[setReview\(java.lang.String\)](#) - Method in class bookCatalog.[BookDAO](#)

[setReviewer\(java.lang.String\)](#) - Method in class bookCatalog.[BookDAO](#)

[setTitle\(java.lang.String\)](#) - Method in class bookCatalog.[BookDAO](#)

[Shopkeeper](#) - Actor in Use Case Diagram [Book Catalog Use Case](#)

A user of the system who is responsible for managing the catalogue of products.

[Shopkeeper](#) - Object in Sequence Diagram [AddBookToCatalogSequence](#)

[Shopkeeper](#) - Object in Sequence Diagram [UpdateOrDeleteBookFromCatalogSequence](#)

[store\(\)](#) - Method in class bookCatalog.[BookDAO](#)

This method opens a connection to the database and adds a new book record based on the information stored in the BookDAO object's private attributes.

T

[testServer](#) - Node in Deployment Diagram [ECP Deployment Model](#)

This is the test server for the initial deployment of the ECP.

[toHTML\(\)](#) - Method in class bookCatalog.[BookDAO](#)

Returns a String which is an HTML representation of the BookDAO object.

[tomcat](#) - Component in [testServer](#)(in Deployment Diagram [ECP Deployment Model](#))

An instance of the Tomcat JSP and servlet container.

[toString\(\)](#) - Method in class bookCatalog.[BookDAO](#)

Returns a String representation of the BookDAO object.

U

[UpdateOrDeleteBookFromCatalogSequence](#) - Sequence Diagram in [bookCatalog](#)

[UpdateProduct](#) - UseCase in [bookCatalog Subsystem](#)(in Use Case Diagram [Book Catalog Use Case](#))
Update the details of a Product in the ProductCatalog

W

[WEB-INF](#) - Component in [htdocs](#)(in Component Diagram [Component](#))

The standard directory structure for J2EE web applications specifies that there should be a directory called WEB-INF hanging off the root of the application.

[Overview](#) [Package](#) [Class](#) [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#)

How This API Document Is Organized

This API (Application Programming Interface) document has pages corresponding to the items in the navigation bar, described as follows.

Overview

The [Overview](#) page is the front page of this API document and provides a list of all packages with a summary for each. This page can also contain an overall description of the set of packages.

Package

Each package has a page that contains a list of its classes and interfaces, with a summary for each. This page can contain four categories:

- Interfaces (*italic*)
- Classes

Class/Interface

Each class, interface, inner class and inner interface has its own separate page. Each of these pages has three sections consisting of a class/interface description, summary tables, and detailed member descriptions:

- Class inheritance diagram
- Direct Subclasses
- All Known Subinterfaces
- All Known Implementing Classes
- Class/interface declaration
- Class/interface description
- Inner Class Summary
- Field Summary
- Constructor Summary
- Method Summary
- Field Detail

- [Constructor Detail](#)
- [Method Detail](#)

Each summary entry contains the first sentence from the detailed description for that item. The summary entries are alphabetical, while the detailed descriptions are in the order they appear in the source code. This preserves the logical groupings established by the programmer.

Tree (Class Hierarchy)

There is a [Class Hierarchy](#) page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. The classes are organized by inheritance structure starting with `java.lang.Object`. The interfaces do not inherit from `java.lang.Object`.

- When viewing the Overview page, clicking on "Tree" displays the hierarchy for all packages.
- When viewing a particular package, class or interface page, clicking "Tree" displays the hierarchy for only that package.

Deprecated API

The Deprecated API page lists all of the API that have been deprecated. A deprecated API is not recommended for use, generally due to improvements, and a replacement API is usually given. Deprecated APIs may be removed in future implementations.

Index

The [Index](#) contains an alphabetic list of all classes, interfaces, constructors, methods, and fields.

Prev/Next

These links take you to the next or previous class, interface, package, or related page.[Help_Contents_FramesNoFrames.start](#)

[Overview](#) [Package](#) [Class](#) [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV](#) [NEXT](#)

[FRAMES](#) [NO FRAMES](#)

Overview Package Class Use [Tree](#) [Index](#) [Help](#)

PREV NEXT [FRAMES](#) [NO FRAMES](#)

Packages

[bookCatalog](#)

Overview Package Class Use [Tree](#) [Index](#) [Help](#)

PREV NEXT [FRAMES](#) [NO FRAMES](#)

[Overview](#) [Package](#) [Class](#) [Use](#) [Tree](#) [Index](#) [Help](#)[PREV PACKAGE](#) [NEXT PACKAGE](#)[FRAMES](#) [NO FRAMES](#)

Package bookCatalog

Class Diagram Summary

[bookCatalog](#)

This package contains the design for the book catalog.

Component Diagram Summary

[Component](#)

Deployment Diagram Summary

[ECP Deployment Model](#)

Sequence Diagram Summary

[AddBookToCatalogSequence](#)[UpdateOrDeleteBookFromCatalogSequence](#)

Use Case Diagram Summary

[Book Catalog Use Case](#)

Class Summary

[AddBookServlet](#)

This class adds a new book row to the book database.

[BookCatalogServlet](#)

This class realises the Front Controller pattern described in J2EE Patterns, ISBN0130648841.

<u>BookDAO</u>	This class realizes the Data Access Object pattern described in J2EE Patterns, ISBN0130648841.
<u>BookListTag</u>	This custom tag is used for displaying lists of books from the book database.
<u>BookListTagExtraInfo</u>	A helper class for BookListTag.
<u>DeleteBookServlet</u>	Deletes a book from the database.

[Overview](#) **Package** Class [Use](#) [Tree](#) [Index](#) [Help](#)

[PREV PACKAGE](#) [NEXT PACKAGE](#)

[FRAMES](#) [NO FRAMES](#)
