INTRODUCTION TO EXCEPTION ERRORS

An exception error occurs when a problem is encountered during the processing of a JSP page. When an exception error occurs, an object that stores information about the error is created. Error handling is achieved by accessing the properties of this object.

The two main types of exception errors are RuntimeException errors and Error exception errors. `RuntimeException` and `Error` are the names of the classes that create objects when one of these types of errors is encountered.

RuntimeException errors are the most common type of exception error. These errors can arise from a variety of problems ranging from simple mathematical errors, such as dividing a number by zero, to more complex programming errors, such as specifying an incorrect type when attempting to cast an object.

Error exception errors occur when a problem related to the processing environment arises. For example, a problem with the Java Virtual Machine or a problem with a supporting file that is required by a JSP page will generate an Error exception.

Encountering an exception error does not necessarily mean that the processing of a JSP page must stop. Some errors can be handled within the code for the page. For example, you can create a `try` block and a `catch` block to handle exception errors that could potentially arise when a section of code is processed. This allows your code to recover from an exception error. For information about creating a `try` block and `catch` block, see page 176.

Some exception errors cannot be recovered from. For example, an Error exception error generated by a problem with the Java Virtual Machine cannot be fixed within the code for a JSP page. In such cases, the object can be accessed to determine valuable information about the error and how it may have been caused.

Extra

Although exception errors can arise from a wide variety of situations, the situations can be grouped into three general categories.

Logical Errors

Logical errors are the most common type of errors and usually result when a programmer has not validated parameters or values before performing an action. An example of a common logical error is dividing a number by zero.

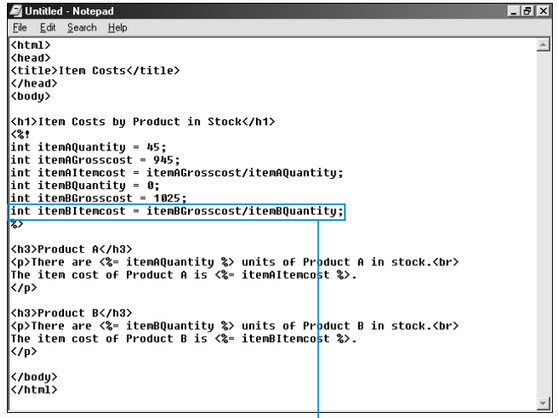
Standard Errors

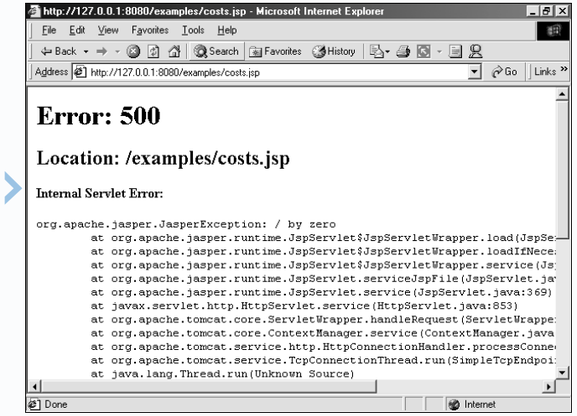
Most of the methods that make up the Java class library contain code that will generate errors when certain situations arise. For example, an error will result if you use a number where a string is expected or assign a value to an array element that does not exist.

Program Errors

Problems with the Java Virtual Machine or a Web server that processes Java code can cause errors to occur. Applications that are not yet stable, such as beta releases, are more likely to generate errors.

RUNTIMEEXCEPTION ERRORS



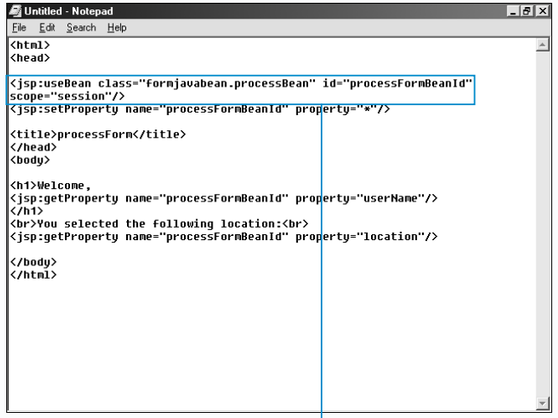


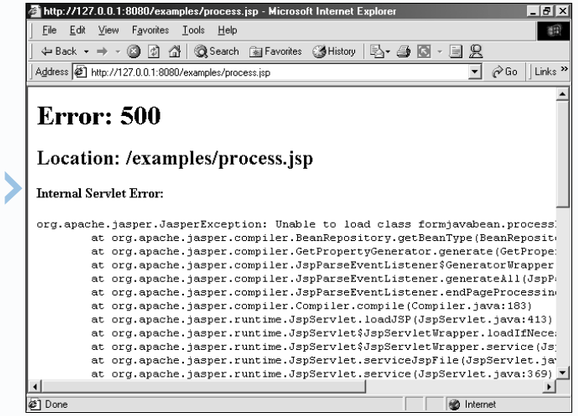
■ A RuntimeException error is typically generated by an error in the code for a JSP page.

■ In this example, the JSP page contains code that divides a number by zero.

■ When the JSP page is displayed in a Web browser, a message appears displaying information about the RuntimeException error.

ERROR EXCEPTION ERRORS





■ An Error exception error is typically generated by a problem with the environment that processes a JSP page.

■ In this example, the JSP page contains code that attempts to access a JavaBean that does not exist.

■ When the JSP page is displayed in a Web browser, a message appears displaying information about the Error exception error.

CREATE A TRY BLOCK AND CATCH BLOCK

If a section of code in a JSP page may generate an exception error, you can create a `try` block and a `catch` block to handle the error.

A `try` block detects if an exception error has occurred in a section of code. To create a `try` block, use the keyword `try` and surround the code that may cause an exception error in braces.

A `catch` block contains the code that is executed when the `try` block detects an error. The `catch` block must immediately follow the `try` block. To create a `catch` block, use the keyword `catch` and enclose the code you want to execute in braces. The `catch` keyword is followed by a parameter enclosed in parentheses. The parameter specifies the class of the exception error and a name for the object that is created when the error occurs.

A `catch` block can only catch the type of exception error specified by the parameter. If the `try` block generates a different type of exception error, the code in the `catch` block will not be executed.

When an exception error occurs in a line of code, the line of code is said to throw an error. When a line of code in a `try` block throws an error, the processing of code in the `try` block stops immediately and any remaining statements in the `try` block are not executed. The `catch` block catches the error thrown by the `try` block and processing continues on the first line of code in the `catch` block.

The code in a `catch` block can display a customized error message to notify a user that an error has occurred. The customized error message should be specific to the error and easy to understand. When a `try` block and `catch` block are not used to handle errors, Java generates cryptic error messages that can be difficult to comprehend.

Extra

As with any Java code, there are strict rules governing the scope of variables used in `try` and `catch` blocks. Variables declared in a `try` block are not available for use in the `catch` block. In the following example, the locationMessage variable is not available in the `catch` block.

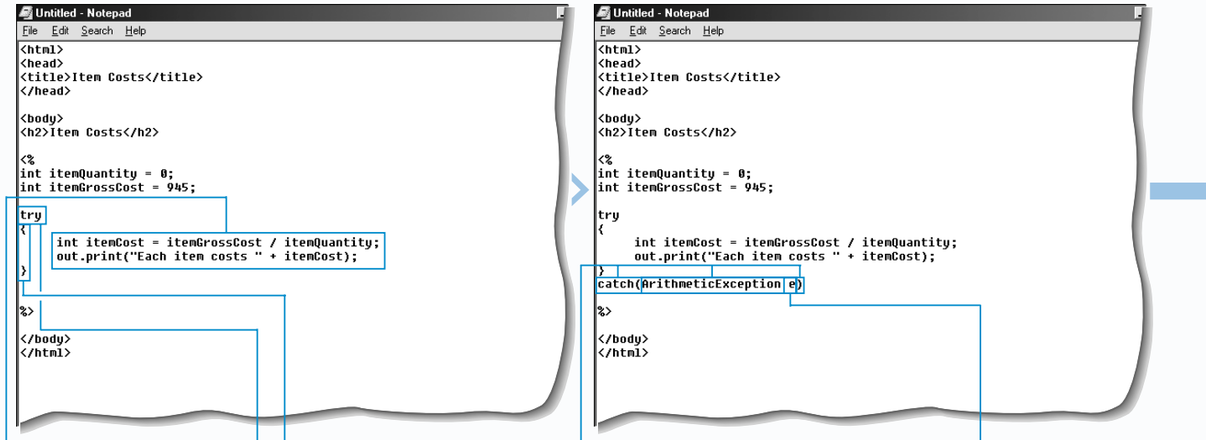
Example:

```
try
{
     String locationMessage = "determining item cost";
     int itemCost = itemGrossCost / itemQuantity;
     out.print("Each item costs " + itemCost);
}
catch(ArithmeticException e)
{
     out.print("Error has occurred at " + locationMessage);
}
```

The above code can easily be rewritten to change the scope of the locationMessage variable so that it is available to both the `try` and `catch` blocks.

```
String locationMessage = "";
try
{
     locationMessage = "determining item cost";
     int itemCost = itemGrossCost / itemQuantity;
     out.print("Each item costs " + itemCost);
}
catch(ArithmeticException e)
{
     out.print("Error has occurred at " + locationMessage);
}
```

CREATE A TRY BLOCK AND CATCH BLOCK



1 Type the code that will generate an exception error.

2 To create a `try` block, enclose the code that will generate an exception error in braces.

3 On the line directly above the opening brace, type **try**.

4 To create a `catch` block, type **catch()** on the line immediately following the `try` block.

5 Between the parentheses, type the class of the exception error that will be thrown by the `try` block.

6 Type a name for the object that will be created when an exception error is thrown.

7 Type the code you want to execute when the `catch` block is processed. Enclose the code in braces.

8 Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of creating a `try` block and `catch` block.

CATCH MULTIPLE EXCEPTION ERRORS

Although a `try` block may be capable of throwing different types of exception errors, a `catch` block can catch only one specific type of exception error. You can create multiple `catch` blocks to catch different types of exception errors.

When a `catch` block is created, the exception error class the block can handle is specified. If the `try` block throws an exception error of a different class, the code in the `catch` block will not be executed. A `try` block that contains a complex section of code may throw different types of exception errors. Creating multiple `catch` blocks allows a section of code to be executed for each type of exception error the `try` block throws.

The first `catch` block must immediately follow the `try` block and each subsequent `catch` block must be placed one right after the other. There cannot be any lines of code between the `try` block and the first `catch` block. You also cannot place lines of code between any of the subsequent `catch` blocks.

When using multiple `catch` blocks, the order of the `catch` blocks is important. For example, since the `Exception` class is a superclass of the `RuntimeException` class, a `catch` block that uses the `Exception` class will catch most of the exception errors thrown by a `try` block. If you place a `catch` block that uses the `Exception` class before other `catch` blocks in your code, the code in the other `catch` blocks may never be executed. As a rule, you should place `catch` blocks that handle exception error subclasses before `catch` blocks that have a broader scope.

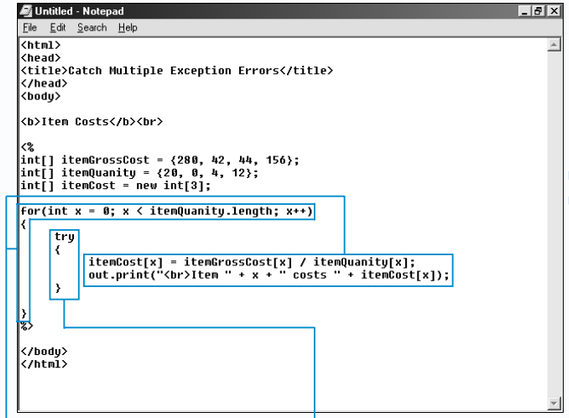
Extra

When using multiple `catch` blocks, you may want to add a `finally` block to your code. A `finally` block executes a section of code regardless of which `catch` block is processed. The `finally` block must immediately follow the last `catch` block.

Example:

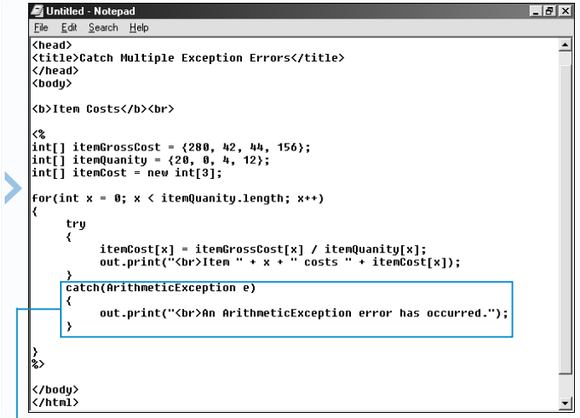
```
for(int x = 0; x < itemQuanity.length; x++)
{
      try
      {
            itemCost[x] = itemGrossCost[x] / itemQuanity[x];
            out.print("<br>Item " + x + " costs " + itemCost[x]);
      }
      catch(ArithmeticException e)
      {
            out.print("<br>An ArithmeticException error has occurred.");
      }
      catch(ArrayIndexOutOfBoundsException e)
      {
            out.print("<br>An ArrayIndexOutOfBoundsException error");
            out.print(" has occurred.");
      }
      finally
      {
            out.print("<hr>");
      }
}
```

CATCH MULTIPLE EXCEPTION ERRORS

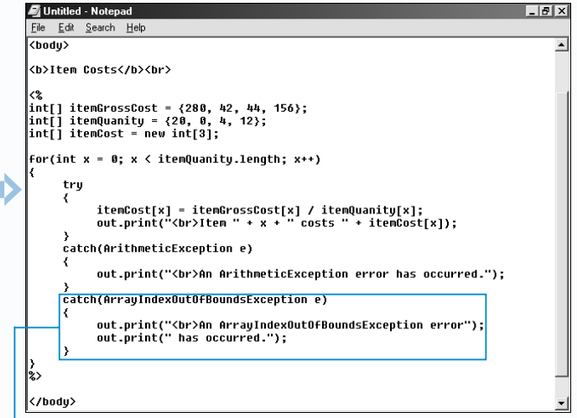


1 Type the code that will generate multiple exception errors.

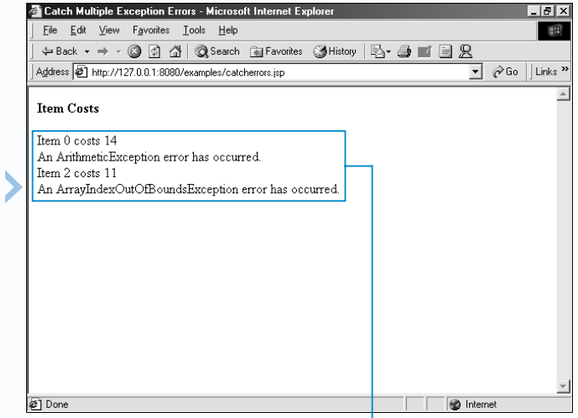
2 Type the code that creates a `try` block.



3 Type the code that creates a `catch` block to handle exception errors of the `ArithmeticException` class.



4 Type the code that creates a `catch` block to handle exception errors of the `ArrayIndexOutOfBounds Exception` class.



5 Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of catching multiple exception errors.

CREATE A FINALLY BLOCK

When a `try` block throws an exception error, the processing of code in the `try` block stops and any remaining statements in the `try` block are not executed. This can cause problems if the `try` block contains code that is important to the execution of your JSP page. To ensure important code is executed regardless of whether an exception error is thrown, you can place the code in a `finally` block.

To create a `finally` block, use the keyword `finally` and enclose the code you want to execute in braces. A `finally` block is useful for performing tasks that 'tidy up' a JSP page. For example, it is common for a `finally` block to contain code that closes a connection to a database or finishes writing data to a file.

There are strict rules governing the scope of variables used in `try`, `catch` and `finally` blocks. Variables declared in a `try` or `catch` block are not available for use in a `finally` block.

When a `try` block uses a `finally` block, a `catch` block is not required. If a `catch` block is used, the `finally` block must immediately follow the `catch` block. If a `catch` block is not used, the `finally` block must immediately follow the `try` block. There can be no lines of code between a `finally` block and a `catch` or `try` block.

When a JSP page containing a `finally` block is processed, the code in the `try` block is executed first. If an error is thrown, the code in the appropriate `catch` block is then executed. The code in the `finally` block is executed last. The `finally` block is executed whether or not an exception error occurs and regardless of the type of exception error thrown.

Apply It

Although the main purpose of a `try` block is to identify code that may generate an exception error, a `try` block can also be used with a `finally` block to save you time when typing code. For example, if a series of `if` statements will all have the same result, you can place the `if` statements in a `try` block and the result in a `finally` block. This saves you from having to type the same result for each `if` statement.

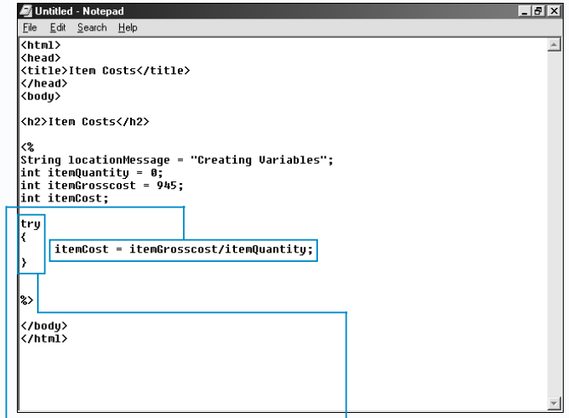
TYPE THIS:

```
try
{
      if (winningScore > 10)
            return 10;
      if (winningScore > 20)
            return 20;
      if (winningScore > 30)
            return 30;
}
finally
{
      out.print("The winning number has been determined.");
}
```

RESULT:

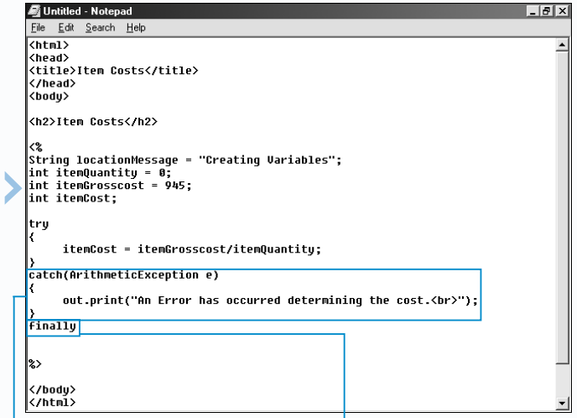
The winning number has been determined.

CREATE A FINALLY BLOCK



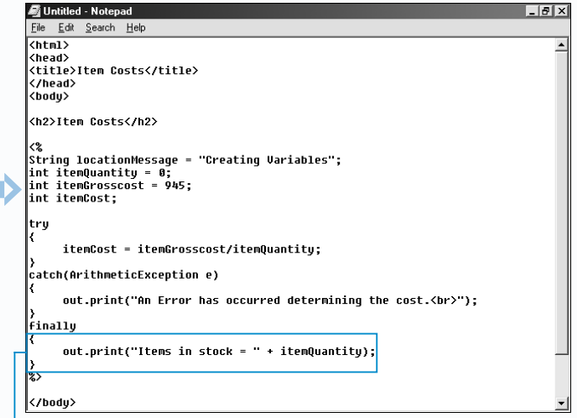
1 Type the code that will generate an exception error.

2 Type the code that creates a `try` block.

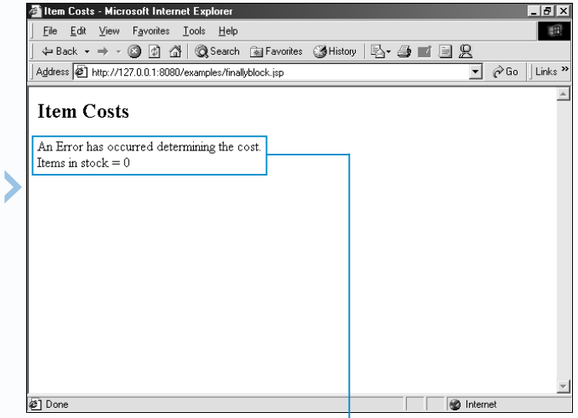


3 Type the code that creates a `catch` block.

4 To create a `finally` block, type **finally** on the line immediately following the `catch` block.



5 Type the code you want to execute when the `finally` block is processed. Enclose the code in braces.



6 Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of using a `finally` block.

REDIRECT TO AN ERROR PAGE

There are many types of exception errors that can be generated by the JSP pages in a Web site. Instead of trying to catch each specific type of exception error that could occur, you can configure the JSP pages to redirect to another page when an error occurs. The error page can be a JSP page or other type of Web document, such as an HTML document.

When an exception error occurs in a JSP page, the Web server stops processing the page and sends an error message to the Web browser to notify the client about the error. The type of exception error that occurs determines the information displayed in the error message. While the error information generated by the server may be useful to someone troubleshooting the JSP page, the information is usually not helpful to clients. Creating an error page allows you to determine the information that a client sees when an exception error occurs. For example, you may want to display a user-friendly page that provides clients with helpful instructions.

To redirect a JSP page to another page in the event of an exception error, you use the `errorPage` attribute of the `page` directive. For more information about the `page` directive, see page 74. The `errorPage` attribute takes the URL of the error page, enclosed in quotation marks, as its value. The URL of the error page must be a relative URL. This means that the JSP page and error page must be stored on the same Web server.

Multiple JSP pages can use the same error page. You must include the redirection instructions on each JSP page you want to use the error page.

Extra

On most Web servers, the default value for the `autoFlush` attribute of the `page` directive is `true`, which means that the buffer is set to automatically flush when it is full. When the buffer is flushed, information in the buffer is sent to a client's Web browser. If the buffer is flushed before a JSP page is redirected to the error page, an additional error will be generated. To avoid this, you can set the value of the `autoFlush` attribute to `false` when using the `errorPage` attribute.

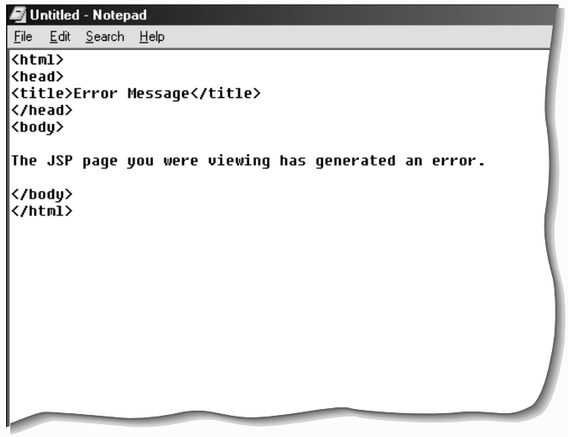
Example:

```
<@ page autoFlush="false" errorPage="error.jsp" %>
```

Information available to the JSP page, such as application values, session values and data stored in a `request` object, will not be available to the error page. For example, if the JSP page processes data from a form, the error page will not be able to access the form information.

When an exception error occurs in a JSP page that uses an error page, the Web server stops processing the JSP page, executes the `page` directive and processes the code in the error page. The Web server does not return to the JSP page.

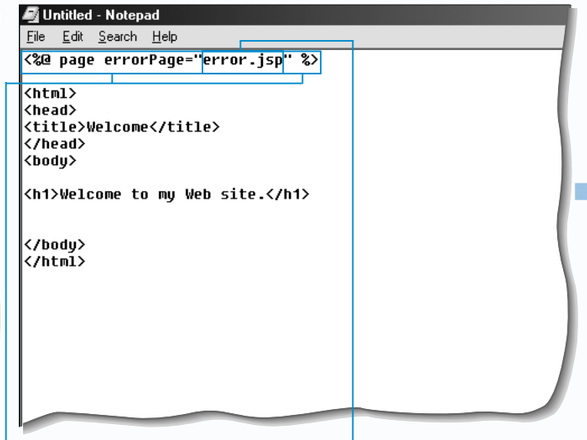
REDIRECT TO AN ERROR PAGE



CREATE AN ERROR PAGE

1 In a text editor, create the page you want to display when an error occurs.

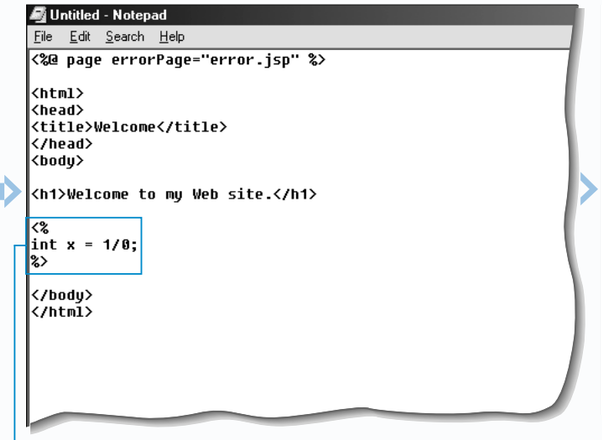
2 Save the page on the Web server.



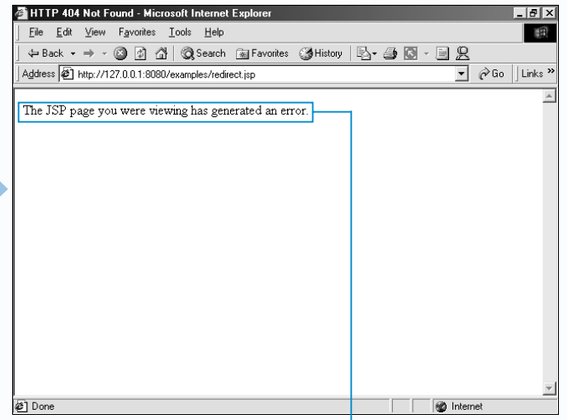
REDIRECT A JSP PAGE TO AN ERROR PAGE

1 On the first line of code in a JSP page you want to redirect to an error page, type **<%@ page errorPage="" %>**.

2 Between the quotation marks, type the URL of the error page.



3 Type the code that will generate an error.



4 Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of redirecting to an error page.

CREATE A DETAILED ERROR PAGE

You can create an error page that accesses detailed information about an exception error that has occurred in a JSP page. Accessing detailed information can help you troubleshoot the page. You can choose to simply display the detailed information about an exception error or you can log the information in a file or database.

When a JSP page generates an exception error of the `Exception` class, an `exception` object is created. The object holds information about the exception error. You can access the `exception` object in an error page to find detailed information about the error that occurred.

To make the `exception` object available to an error page, you use the `isErrorPage` attribute of the `page` directive. For more information about the `page` directive, see page 74. The `isErrorPage` attribute can have a value of either true or false. A value of true will make the `exception` object available to an error page. False is the default value of the `isErrorPage` attribute.

The `getMessage` method of the `exception` object can be called to access an error message that describes the type of error that has occurred. You can use an expression to display the information returned by the `getMessage` method. Some exception errors do not have an error message associated with them. In this case, the `getMessage` method will return a null value. For more information about the methods of the `exception` object, refer to the Java SDK documentation.

To redirect a JSP page to a detailed error page in the event of an error, you use the `errorPage` attribute of the `page` directive. The `errorPage` attribute takes the URL of the detailed error page, enclosed in quotation marks, as its value. The URL of the detailed error page must be a relative URL.

Extra

The `exception` object is only available to the detailed error page. However, there are techniques you can use to make the information in the `exception` object available to other JSP pages. For example, in the detailed error page, you can use the `setAttribute` method of the `session` object to store the error message as a session value.

Example:

```
<%
session.setAttribute("errorMessage", exception.getMessage());
%>
```

You may be able to use the `getLocalizedMessage` method of the `exception` object to access even more detailed information about an exception error. However, in most cases, the `getLocalizedMessage` method returns the same information as the `getMessage` method.

Example:

```
<%= exception.getLocalizedMessage() %>
```

The `toString` method of the `exception` object can be used to display the class name of an exception error. The result of the `toString` method may also contain the information returned by the `getMessage` method.

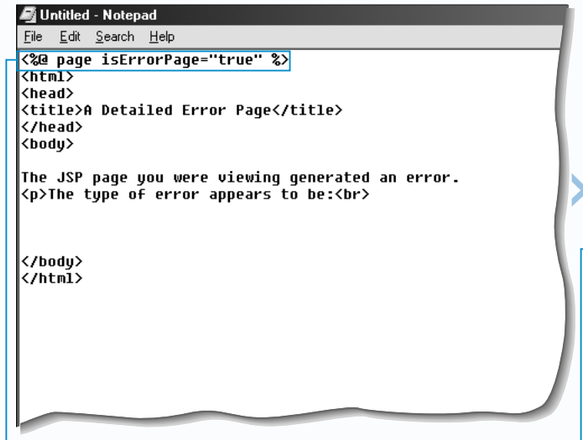
Example:

```
<%= exception.toString() %>
```

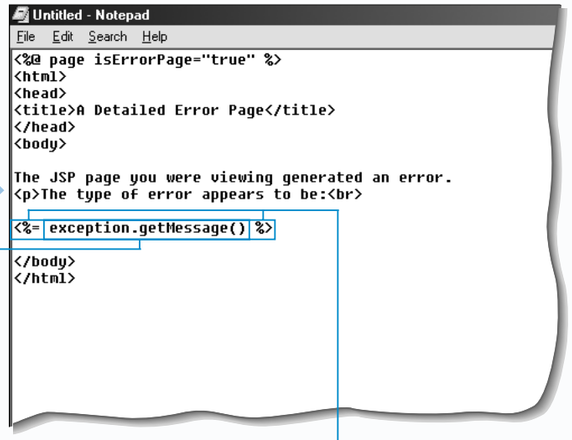
Returns:

java.lang.ArithmeticException: / by zero

CREATE A DETAILED ERROR PAGE



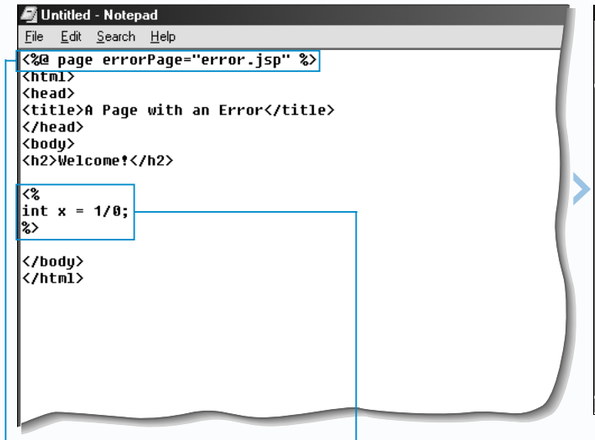
1 On the first line of code in the error page, type **<%@ page isErrorPage="true" %>** to access the `exception` object.



2 Type **exception.getMessage()** where you want to access an error message.

3 Type the code that displays the error message on the detailed error page.

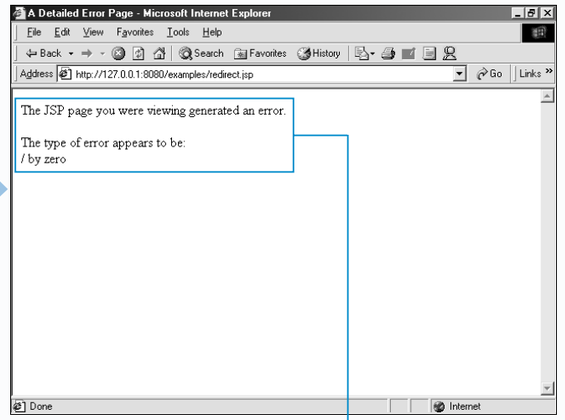
4 Save the page on the Web server with the .jsp extension.



REDIRECT A JSP PAGE TO A DETAILED ERROR PAGE

1 On the first line of code in a JSP page, type the code that redirects the page to the detailed error page.

2 Type the code that will generate an error.



3 Save the page with the .jsp extension and then display the JSP page in a Web browser.

■ The Web browser displays the result of redirecting to a detailed error page.