Error handling and exception handling in Java servlets

These are important for managing unexpected issues and errors that may occur during the execution of a servlet. Let's discuss both concepts and provide code examples for each.

Error Handling:
Error handling in Java servlets typically involves dealing with errors related to the servlet container, such as server configuration issues, low-level errors, and other problems that may not be directly related to your application logic. To handle errors, you can use the `web.xml` configuration file to define error pages. Here's a step-by-step explanation and a code example:

1. **Define an error page in your `web.xml` configuration file**:
   ```xml
   <error-page>
      <error-code>404</error-code>
      <location>/error404.jsp</location>
   </error-page>
   ```

   In this example, we define an error page for HTTP error code 404 (Not Found). When a 404 error occurs, the servlet container will redirect the user to the `/error404.jsp` page.

2. **Create the error handling JSP page**:
   Create the JSP page `error404.jsp` that will be shown to the user in case of a 404 error. You can customize this page to provide a user-friendly error message.

   ```jsp
   <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
   <!DOCTYPE html>
   <html>
   <head>
      <title>Error 404 - Page Not Found</title>
   </head>
   <body>
      <h1>Error 404 - Page Not Found</h1>
      <p>The requested page could not be found.</p>
   </body>
   </html>
   ```

Now, when a user requests a non-existent page, the servlet container will handle the 404 error and display the custom error page.

Exception Handling:
Exception handling in Java servlets deals with application-specific exceptions that occur within your servlet code. You can use try-catch blocks to catch and handle exceptions. Here's an explanation and a code example:

1. **In your servlet code, use try-catch blocks**:
   In your servlet, you can use try-catch blocks to catch and handle exceptions. For example, if you're trying to access a database and a SQLException occurs, you can catch it and take appropriate action:

   ```java
   import java.io.IOException;
   import java.sql.SQLException;
   import javax.servlet.ServletException;
   import javax.servlet.annotation.WebServlet;
   import javax.servlet.http.HttpServlet;
   import javax.servlet.http.HttpServletRequest;
   import javax.servlet.http.HttpServletResponse;

   @WebServlet("/my-servlet")
   public class MyServlet extends HttpServlet {
       protected void doGet(HttpServletRequest request, HttpServletResponse response)
               throws ServletException, IOException {
           try {
               // Code that may throw an exception, e.g., database operation
               // ...
           } catch (SQLException e) {
               // Handle the exception, e.g., log it or show an error page
               e.printStackTrace();
               response.sendError(HttpServletResponse.SC_INTERNAL_SERVER_ERROR,
"Database error");
           }
       }
   }
   ```

In this example, we catch a `SQLException` and then send an internal server error (HTTP 500) response to the client with an error message.

Exception handling allows you to gracefully handle application-specific issues, log them, and provide meaningful error messages to users. Error handling, on the other hand, deals with issues related to the servlet container and is configured in the `web.xml` file.