Session management in Java refers to the process of maintaining the state of a user's interaction with a web application across multiple HTTP requests. Sessions are essential for maintaining user-specific data, such as login credentials, shopping cart items, and other user-related information.

There are several ways to manage sessions in Java, but the most common approach is by using cookies and/or URL rewriting. I'll explain these two methods and provide code examples for each.

1. Using Cookies for Session Management:

Cookies are small pieces of data that the server sends to the client's browser, and the browser stores them locally. They can be used to identify and maintain a session for a user. Here's a simple example of how to use cookies for session management in Java:

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

public class SessionManagementWithCookies extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        HttpSession session = request.getSession();
        String username = (String) session.getAttribute("username");

        if (username != null) {
            response.getWriter().println("Welcome back, " + username);
        } else {
            String newUsername = "JohnDoe";  // You would typically get this from a login form
            session.setAttribute("username", newUsername);
            response.getWriter().println("Hello, " + newUsername);
        }
    }
}
```

In this example, we use the `HttpSession` to create and manage a session. When a user visits the page, the servlet checks if a session exists. If a session does not exist or if the "username" attribute is not set, it sets the "username" attribute and displays a welcome message. If the attribute is already set, it displays a welcome back message.

2. Using URL Rewriting for Session Management:

URL rewriting involves appending session information as query parameters to URLs. This approach is less secure than cookies, but it works in scenarios where cookies might not be available. Here's a code example using URL rewriting:

```java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.IOException;

public class SessionManagementWithURLRewriting extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        String username = request.getParameter("username");

        if (username != null) {
            response.getWriter().println("Welcome back, " + username);
        } else {
            String newUsername = "JohnDoe";  // You would typically get this from a login form
            response.sendRedirect(request.getRequestURI() + "?username=" + newUsername);
            // The above line appends the username as a query parameter
        }
    }
}
```

In this example, the username is appended to the URL as a query parameter. If the username parameter is present in the URL, it displays a welcome message. If not, it redirects to the same page with the username parameter added to the URL.

It's important to note that session management should be used with caution, and security measures should be in place to protect user data and prevent unauthorized access. Additionally, there are more advanced session management techniques, such as using frameworks like Spring Session or custom implementations with databases, depending on the complexity and requirements of your application.