# Race Condition: Singleton Member Field

Many Servlet developers do not understand that a Servlet is a singleton. There is only one instance of the Servlet, and that single instance is used and re-used to handle multiple requests that are processed simultaneously by different threads.

A common result of this misunderstanding is that developers use Servlet member fields in such a way that one user may inadvertently see another user's data. In other words, storing user data in Servlet member fields introduces a data access race condition.

# Race Condition: Singleton Member Field Prevention

Do not use Servlet member fields for anything but constants. (i.e. make all member fields static final).

Developers are often tempted to use Servlet member fields for user data when they need to transport data from one region of code to another. If this is your aim, consider declaring a separate class and using the Servlet only to "wrap" this new class.

# Example

The following Servlet stores the value of a request parameter in a member field and then later echoes the parameter value to the response output stream.

public class GuestBook extends HttpServlet {

String name;

protected void doPost (HttpServletRequest req, res) {

name = req.getParameter("name");

...

out.println(name + ", thanks for visiting!");

}

}

While this code will work perfectly in a single-user environment, if two users access the Servlet at approximately the same time, it is possible for the two request handler threads to interleave in the following way:

Thread 1: assign "Dick" to name

Thread 2: assign "Jane" to name

Thread 1: print "Jane, thanks for visiting!"

Thread 2: print "Jane, thanks for visiting!"

Thereby showing the first user the second user's name.

# References

<http://www.hpenterprisesecurity.com/vulncat/en/vulncat/java/singleton_member_field_race_condition.html>

<https://www.owasp.org/index.php/Member_Field_Race_Condition>