Session Management

Server-Side Web Languages

Uta Priss
School of Computing
Napier University, Edinburgh, UK

Outline

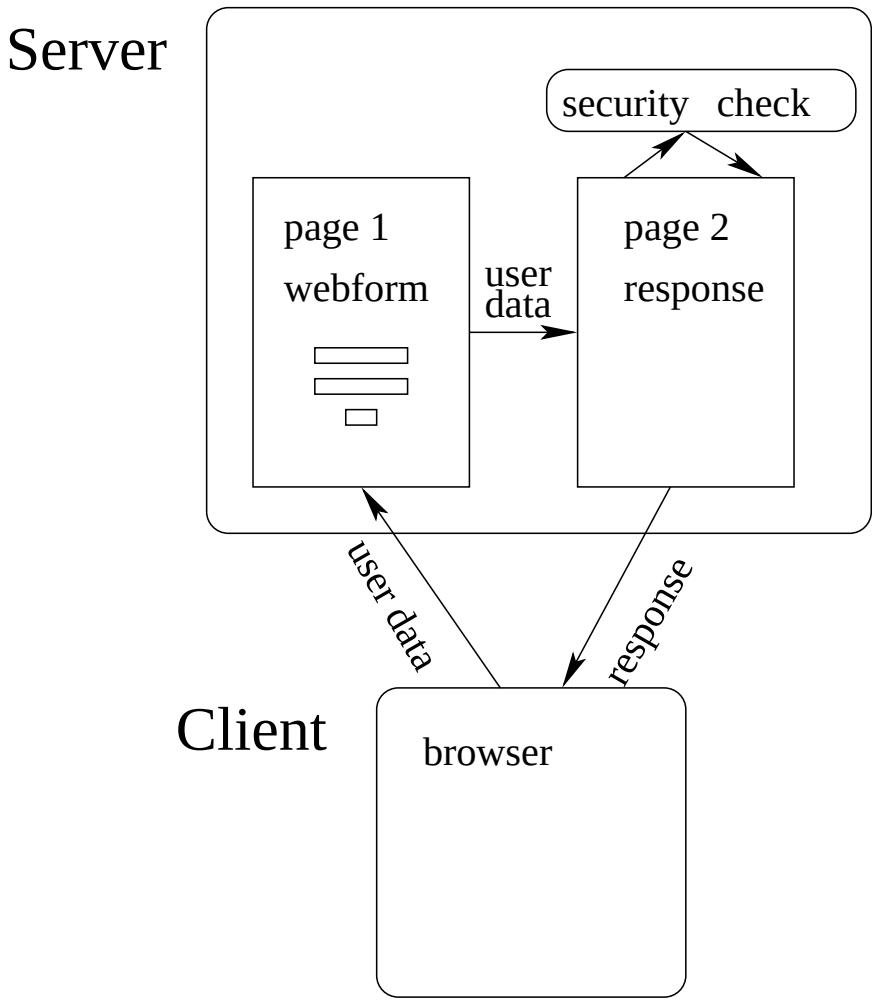
Session Management

Cookies

A challenge for server-side web languages

Session management poses a challenge for server-side web languages because, in contrast to stand-alone applications, server-side applications transfer data between servers and clients. A server-side application does not have full control over the data at the client's computer. A client can delete, modify or add to any data. Therefore client data cannot be trusted.

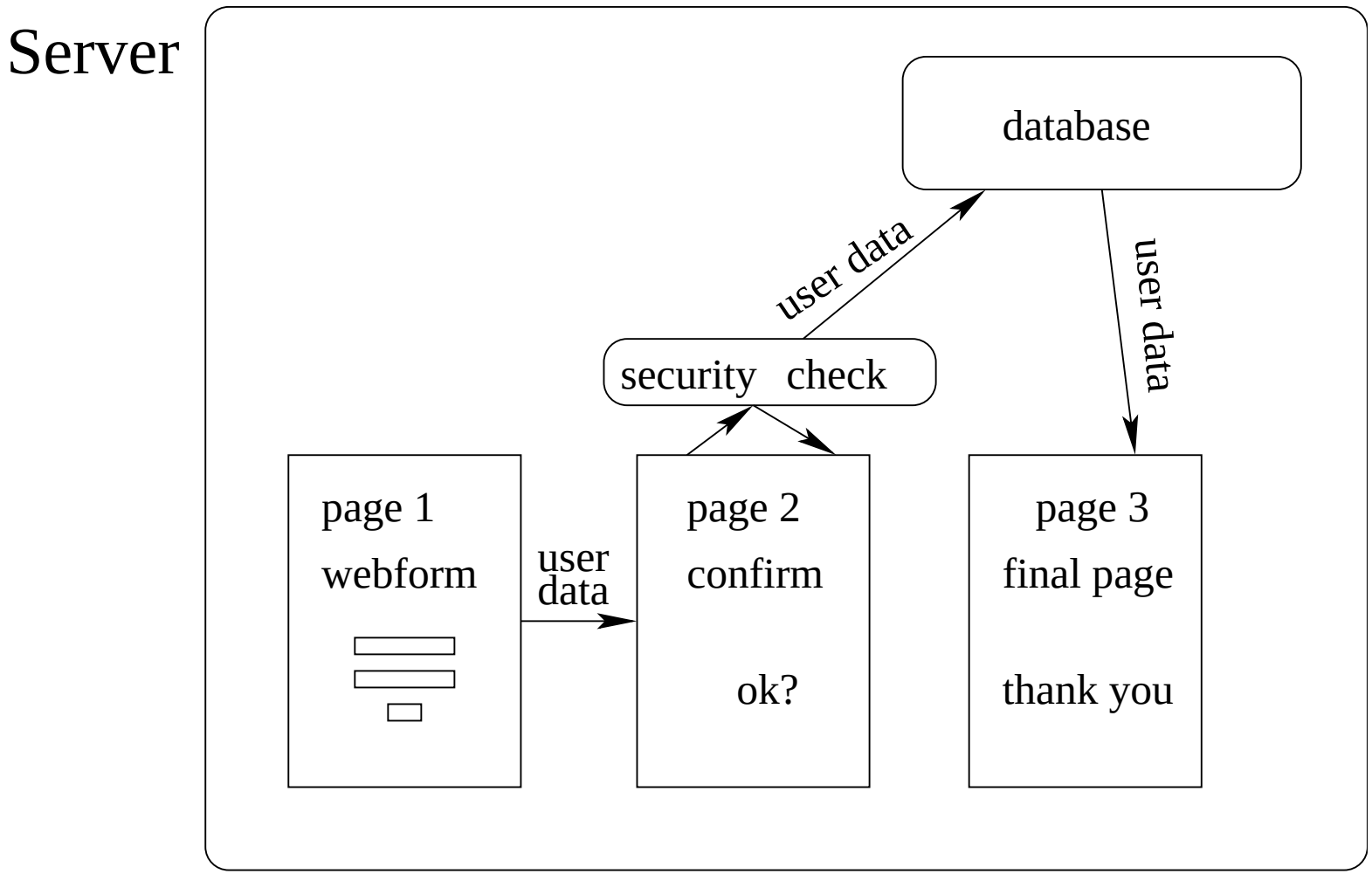
A webform and response without sessions



Using session IDs

- Sessions are created to minimise the risk of data corruption.
- All user data is stored on the server and associated with a session ID. Data retrieval is based on the session ID.
- A security check must be performed on any data, which a user enters, before the data is stored on the server.
- It is not necessary to perform security checks on data that is retrieved from the server. This is in contrast to CGI applications without sessions: in that case a security check must be performed on user data every time the data is used!

Webform, confirmation and response

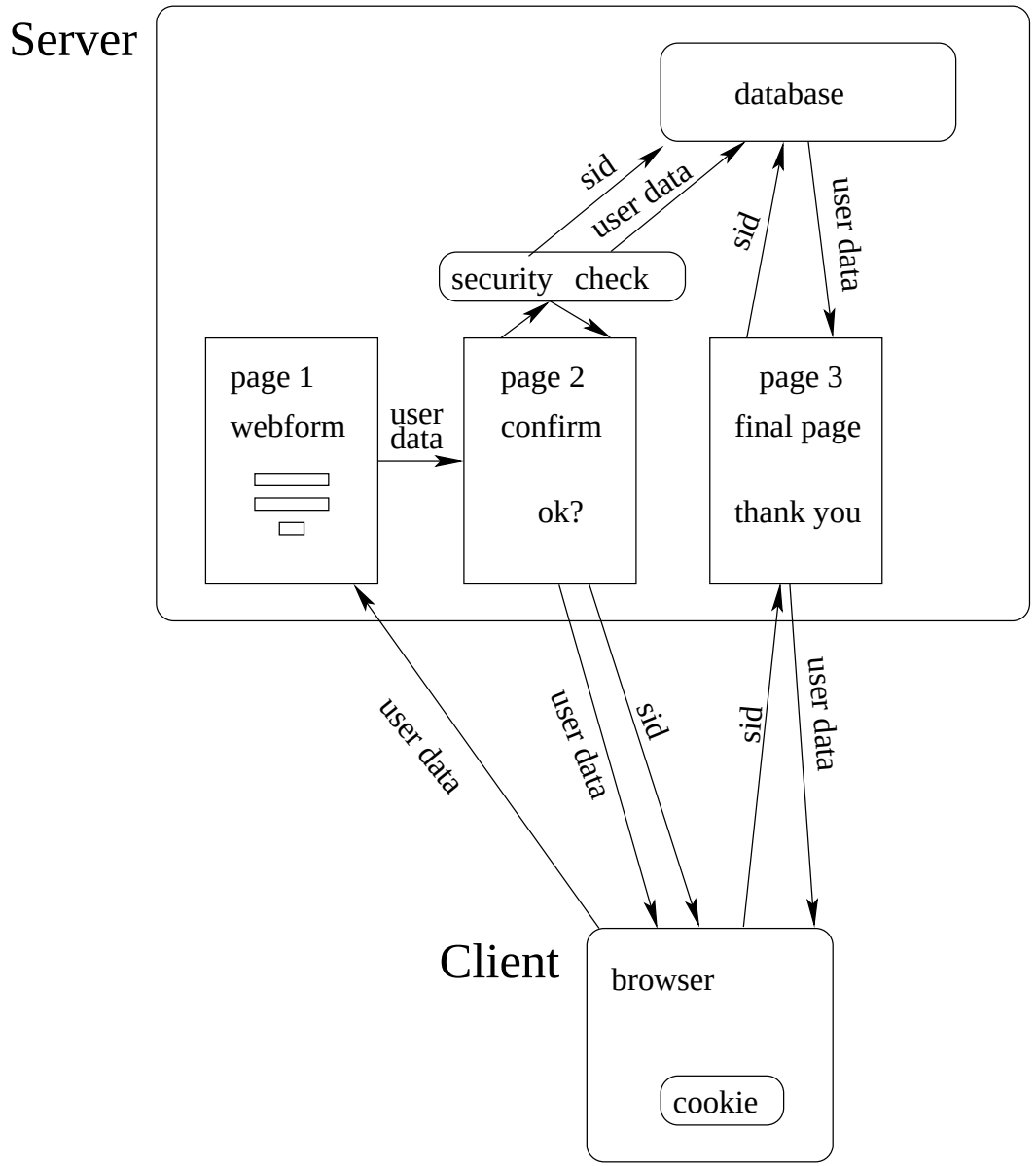


Session IDs

Session IDs are passed from page to page either

- by using hidden HTML form text or
- by using the query string or
- by using cookies (on the client computer).

The session IDs should be generated by the server and should be long and complicated enough to deter users from manual tampering with the IDs.


Server
database
sid
user data
sid
user data
security check
page 1
webform
user
data
page 2
confirm
ok?
page 3
final page
thank you
user data
user data
sid
sid
user data
Client
browser
cookie

Storing user data

The user data, which is associated with a session ID can be stored on the server

- in a file
- in a database.

This approach reduces the number of security checks that must be performed on user data.

Storing user data

The user data, which is associated with a session ID can be stored on the server

- in a file
- in a database.

This approach reduces the number of security checks that must be performed on user data.

Further measures can be taken for applications which require high security:

- the session can be password protected;
- the cookie data can be encrypted;
- the whole session can be encrypted using SSL.

Cookies

Cookies are transmitted between client and server via the HTTP_COOKIE environment variable.

This means that any cookie data needs to be **written** before the HTML header is sent.

Cookies

Cookies are transmitted between client and server via the HTTP_COOKIE environment variable.

This means that any cookie data needs to be **written** before the HTML header is sent.

Cookie data can be **read** at any time.

Workflow for cookies

- First: read any existing cookie.
- Second: compile cookie data and send it.
- Third: Print the HTML header.

Workflow for cookies

- First: read any existing cookie.
- Second: compile cookie data and send it.
- Third: Print the HTML header.

Note:
it is counterintuitive to first read the cookie and then write it!

Example: creating a counter with a cookie

- Read the cookie data.

Example: creating a counter with a cookie

- Read the cookie data.
  If no prior cookie is received, set counter = 0.

Example: creating a counter with a cookie

- Read the cookie data.
  If no prior cookie is received, set counter = 0.
  Read counter value and increase by 1.

Example: creating a counter with a cookie

- Read the cookie data.
  If no prior cookie is received, set counter = 0.
  Read counter value and increase by 1.
- Write the cookie.
- Print the HTML header.