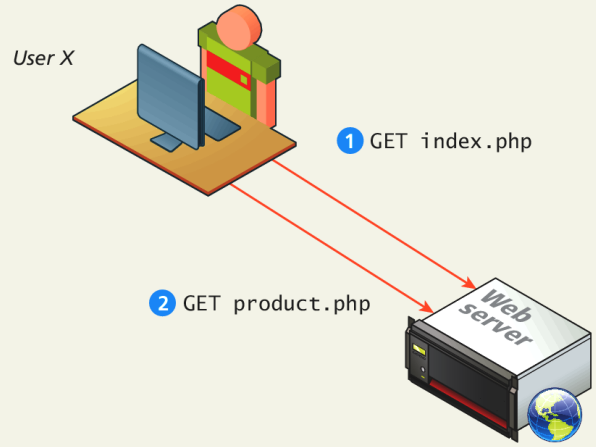
Managing State

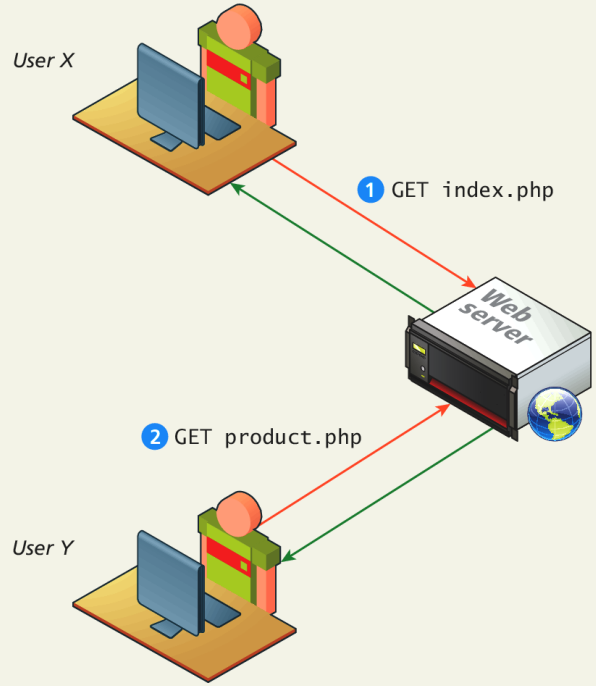
Chapter 13

State in Web Applications

What's the issue?

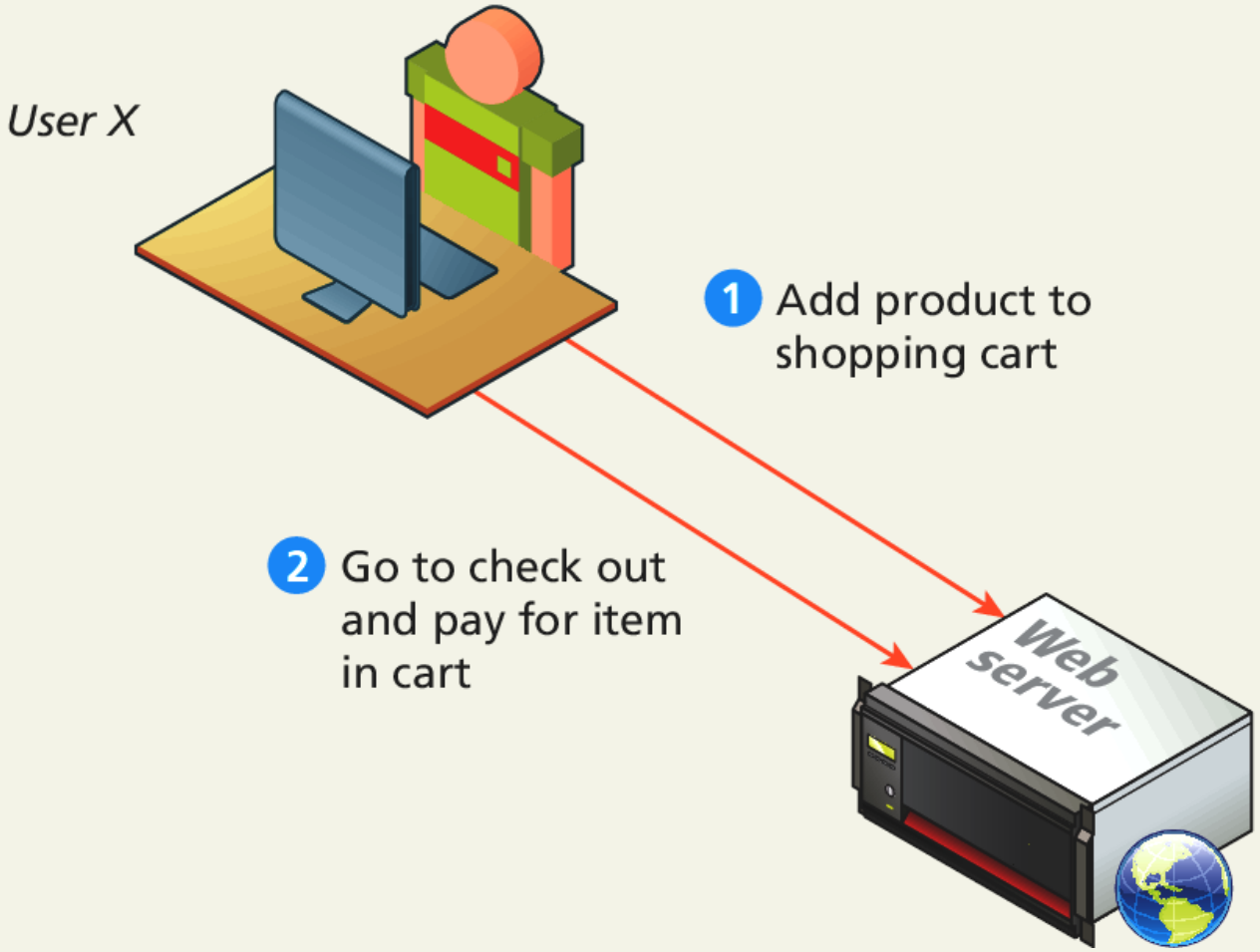


... is for the server not really any different than ...



State in Web Applications

What's the desired outcome



State in Web Applications

Not like a desktop application

Unlike the unified single process that is the typical desktop application, a web application consists of a series of disconnected HTTP requests to a web server where each request for a server page is essentially a request to run a separate program.

The HTTP protocol does not, without programming intervention, distinguish two requests by one source from two requests from two different sources

State in Web Applications

How do we reach our desired outcome?

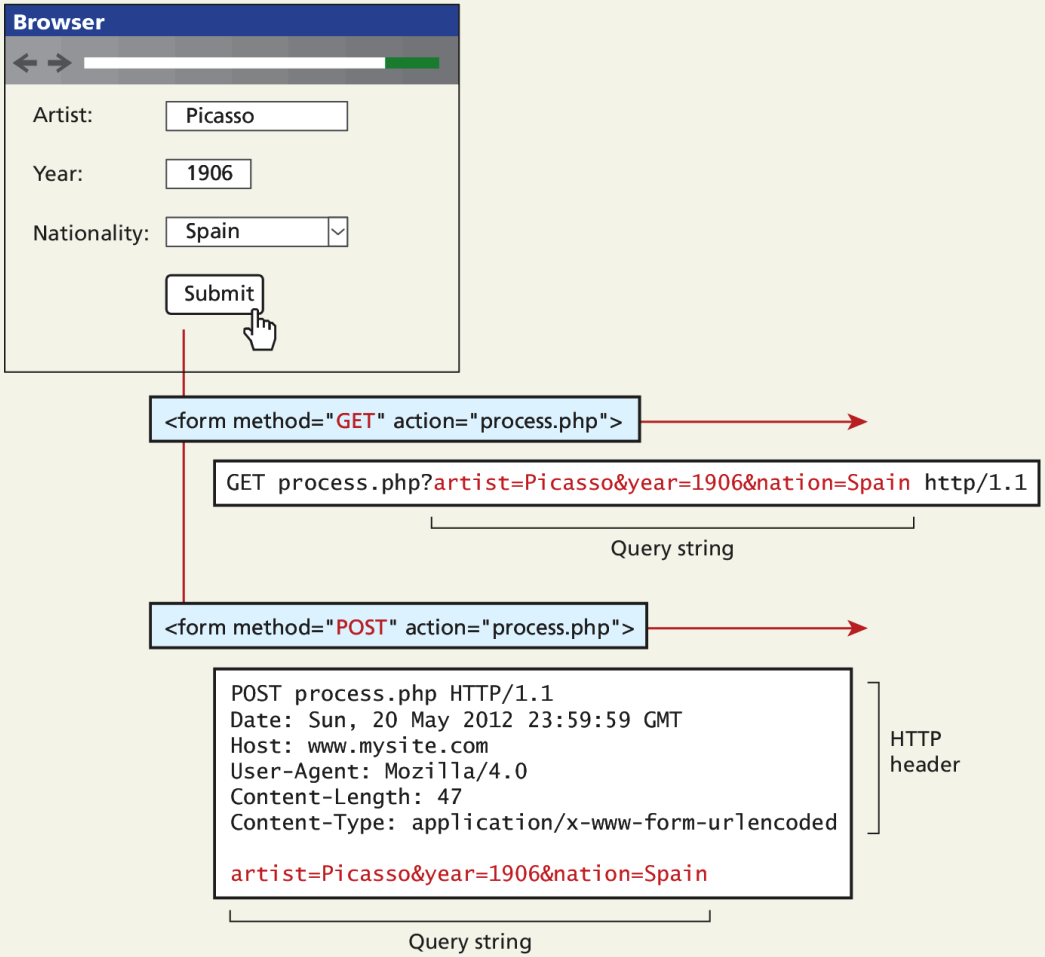
What mechanisms are available within HTTP to pass information to the server in our requests?

In HTTP, we can pass information using:

- Query strings (not to be confused with database query strings!) either from:
  - Forms or
  - Hyperlinks
- Cookies
- Sessions

Info in Query Strings

Recall GET and POST



Passing Info via URL Path

Another way to pass information from one page to the next is by appending a query string to the URL in a hyperlink:

```
<ul>
   <li><a href="list.php?category=7">Business</a></li>
   <li><a href="list.php?category=2">Computer Science</a></li>
   <li><a href="list.php?category=3">Economics</a></li>
   <li><a href="list.php?category=9">Engineering</a></li>
   <li><a href="list.php?category=4">English</a></li>
   <li><a href="list.php?category=6">Mathematics</a></li>
   <li><a href="list.php?category=8">Statistics</a></li>
   <li><a href="list.php?category=5">Student Success</a></li>
</ul>
```

Cookies

Cookies are a client-side approach for persisting state information.

They are name=value pairs that are saved within one or more text files that are managed by the browser.

They can accompany server requests and responses within the HTTP header.

They do not contain viruses, but third-party tracking cookies are common.

Cookies

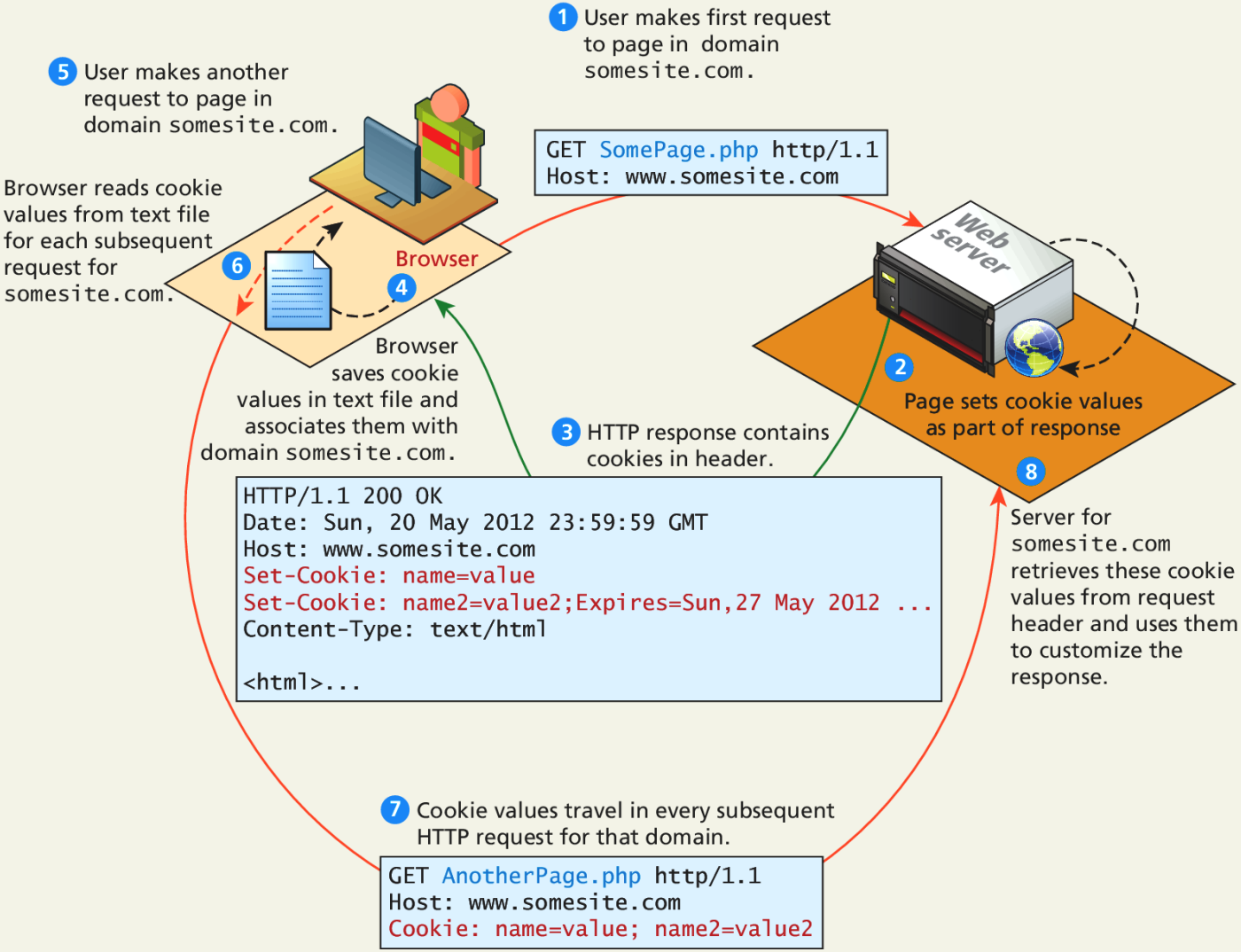
How do they work?

While cookie information is stored and retrieved by the browser, the information in a cookie travels within the HTTP header.

- Sites that use cookies should not depend on their availability for critical features
- The user can delete cookies or edit them or disallow them altogether.

Cookies

How do they Work?



Cookies

Two kinds of cookie

- A **session cookie** has no expiration stated and thus will be deleted at the end of the user's browsing session.
- **Persistent cookies** have an expiration date which can be specified.

Using Cookies

Writing a cookie

```php
<?php
   // add 1 day to the current time for expiry time
   $expiryTime = time()+60*60*24;

   // create a persistent cookie
   $name = "Username";
   $value = "Ricardo";
   setcookie($name, $value, $expiryTime);
?>
```

LISTING 13.1 Writing a cookie

It is important to note that cookies must be written before any other page output.

Using Cookies

Reading a cookie

```php
<?php
   if( !isset($_COOKIE['Username']) ) {
      //no valid cookie found
   }
   else {
      echo "The username retrieved from the cookie is:";
      echo $_COOKIE['Username'];
   }
?>
```

LISTING 13.2 Reading a cookie

Using Cookies

Common usages

In addition to being used to track authenticated users and shopping carts, cookies can implement:

- “Remember me” persistent cookie
- Store user preferences
- Track a user’s browsing behavior

Just remember: Do not depend on cookies for any critical features!

Session State

All modern web development environments provide some type of session state mechanism.

Session state is a server-based state mechanism that lets web applications store and retrieve objects of any type for each unique user session.

Session state is ideal for storing more complex objects or data structures that are associated with a user session.

- In PHP, session state is available to the via the $_SESSION variable
- Must use session_start() to enable sessions.

Session State

Accessing State

session_start() checks to see if a session is already running
- if so, it continues with the existing session variables
- if not, it creates a new session

```php
<?php

session_start();

if ( isset($_SESSION['user']) ) {
   // User is logged in
}
else {
   // No one is logged in (guest)
}
?>
```

LISTING 13.5 Accessing session state

Session State

Checking Session existence

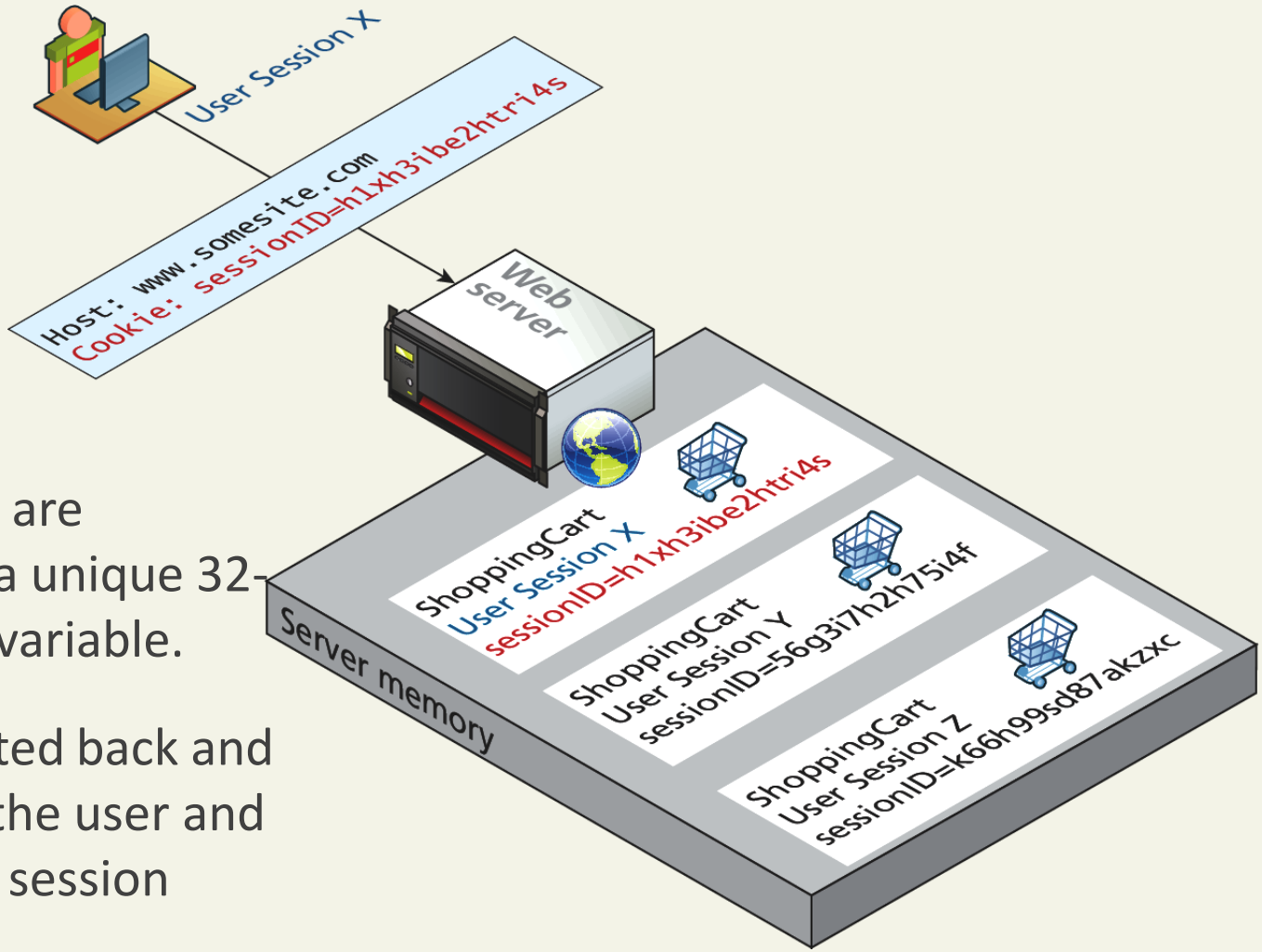
```php
<?php
include_once("ShoppingCart.class.php");

session_start();

// always check for existence of session object before accessing it
if ( !isset($_SESSION["Cart"]) ) {
   //session variables can be strings, arrays, or objects, but
   // smaller is better
   $_SESSION["Cart"] = new ShoppingCart();
}
$cart = $_SESSION["Cart"];
?>
```

LISTING 13.6 Checking session existence

How does state session work?



Sessions in PHP are identified with a unique 32-byte **sessionID** variable.

This is transmitted back and forth between the user and the server via a session cookie

How does state session work?

- For a brand new session, PHP assigns an initially empty dictionary-style collection that can be used to hold any state values for this session.
- When the request processing is finished, the session state is saved to some type of state storage mechanism, called a session state provider
- When a new request is received for an already existing session, the session's dictionary collection is filled with the previously saved session data from the session state provider.

Session Storage

It is possible to configure many aspects of sessions including where the session files are saved.

The decision to save sessions to files rather than in memory (like ASP.NET) addresses the issue of memory usage that can occur on shared hosts as well as persistence between restarts.

Inexpensive web hosts may sometimes stuff hundreds or even thousands of sites on each machine.

Server memory may be storing not only session information, but pages being executed, and caching information

Ending a Session

A session ends:

- after 24 minutes without a request
- when the user closes the browser
- when the code calls the session_destroy() function

But, session_destroy() doesn't do it all, associated data must be deleted as well.

Ending a Session

To remove the data associated with a session:

1. Clear the session array: **$_SESSION = array();**
2. Delete the session cookie:
   setcookie('PHPSESSID', '', time()-3600, '/', '', 0, 0);
3. Destroy the session itself: **destroy_session();**

Web Storage

HTML5 only

Web storage is a new JavaScript-only API introduced in HTML5.4

It is meant to be a replacement (or perhaps supplement) to cookies, in that web storage is managed by the browser.

Unlike cookies, web storage data is not transported to and from the server with every request and response.

In addition, web storage is not limited to the 4K size barrier of cookies.