State Management (Cookies)

Web Architecture and Information Management [./] Spring 2009 — INFO 190-02 (CCN 42509)

Erik Wilde, UC Berkeley School of Information

2009-03-04

Contents

Abstract (2)

HTTP is a stateless protocol, where each request/response interaction is a separate interaction and there is no protocol support for longer sessions (such as a user logging in and working on a Web site as an identified user). State management refers to mechanisms which provide support for this kind of scenario, the most popular choice for state management are *cookies*. Another possibility is URI-based state management. This lecture is a glimpse into the world of *Representational State Transfer (REST)*, the Web's fundamental model of handling interaction with resources.

Session

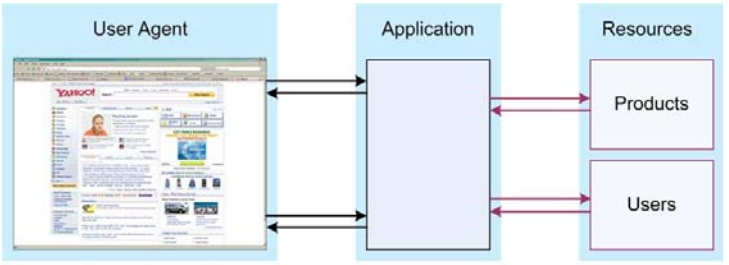
HTTP and Sessions (4)

- HTTP has no session concept
  - interactions are HTTP request/response pairs and not site visits
  - HTTP/1.1 [Web Foundations (URI and HTTP); HTTP Performance (1)] does not change this, it is only a performance optimization
  - servers can not reliably identify users interacting with a Web site
- Sessions should not be used to track resource state
  - the semantics of resource interactions should not depend on client state
  - application behavior can depend on client state
- HTTP's concept of *stateless interaction* is important
  - the Web's idea is to use *loose coupling* between clients and servers/resources
  - retrofitting the Web with *tight coupling* through server state is bad design

Client-Side State (5)

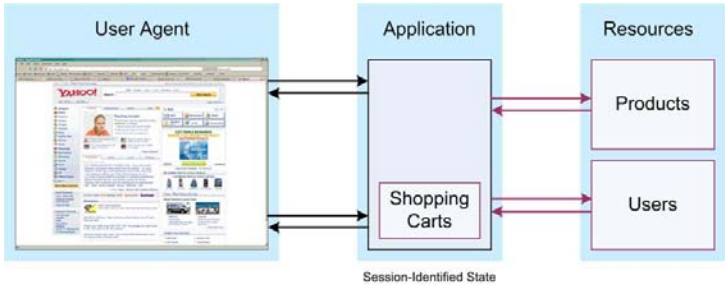
- Sessions should be maintained on the client
  - the client has all relevant information about a session
  - when the server restarts, no information will be lost
  - if something has to be persistent, it should be a resource
- Small and short-term solutions may work well with server state
  - *scaling* these solutions typically introduces many problems
  - *debugging* can be hard because the state is transient
  - *integration* with other clients can become a difficult problem
- Three ways of client-side state are possible
  1. sending back and forth state as part of the interaction
  2. store state in the server and refer to it from the client (not recommended)
  3. store state at a URI and use the URI to refer to that state

State in HTML or HTTP (6)

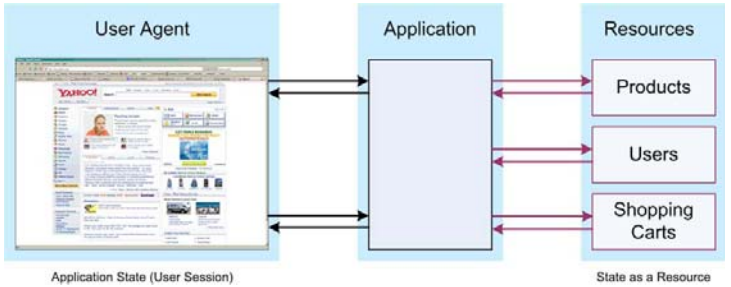


State in HTML or HTTP Authentication Info

State in the Server Application (7)



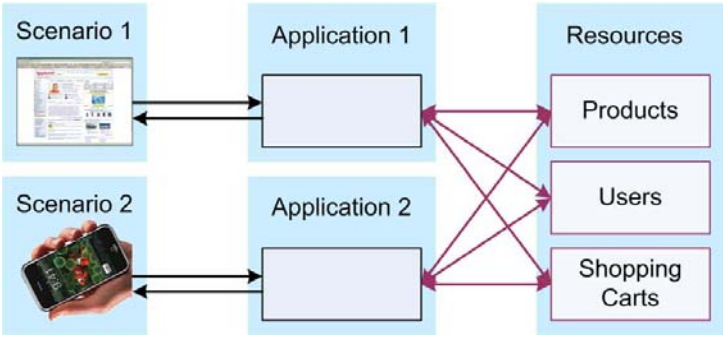
State as a Resource (8)



Stateless Shopping (9)

- Typical "session scenarios" can be mapped to resources [http://www.peej.co.uk/articles/no-sessions.html]
  - Client: Show me your products
  - Server: Here's a list of all the products
  - Client: I'd like to buy 1 of http://ex.org/product/X, I am "John"/"Password"
  - Server: I've added 1 of http://ex.org/product/X to http://ex.org/users/john/basket
  - Client: I'd like to buy 1 of http://ex.org/product/Y, I am "John"/"Password"
  - Server: I've added 1 of http://ex.org/product/Y to http://ex.org/users/john/basket
  - Client: I don't want http://ex.org/product/X, remove it, I am "John"/"Password"
  - Server: I've removed http://ex.org/product/X to http://ex.org/users/john/basket
  - Client: Okay I'm done, username/password is "John"/"Password"
  - Server: Here is the total cost of the items in http://ex.org/users/john/basket
- This is more than just renaming "session" to "resource"
  - all relevant data is stored persistently on the server
  - the shopping cart's URI can be used by other services for working with its contents
  - instead of *hiding the cart in the session*, it is *exposed as a resource*

Reusing Resources (10)

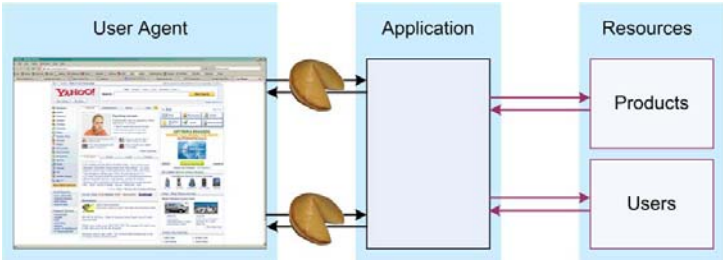


Cookie

Tracking Sessions (12)

- Invented as a way to compensate for HTTP's lack of state
  - application state is being sent to the client (SetCookie2)
  - the client transmits application state in requests (Cookie)
- Cookies do not contain code that is executed
  - some data that represents application state (by value or by reference)
  - this data is stored by the client and returned to the server
  - the client is not supposed to interpret the data in any way
- Cookies can be used in many different ways
  - when used for tracking application state they are unproblematic
  - when used for tracking resource state they introduce problems
- Cookies tightly bind clients to opaque concepts on the server

Cookies for State Management (13)



Web Architecture and Information Management [./]
Spring 2009 — INFO 190-02 (CCN 42509)
2009-03-04

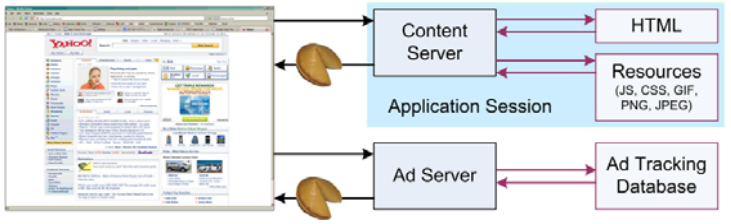
Third-Party Cookie

Advertising & Privacy (15)

- Big ad servers are digital hubs in the commercial Web
  - consumers switch content providers but get the same ad provider
  - tracking consumers *across* content providers is very valuable
- Cookies set by ad providers are sent very frequently
  - each site that uses the ad provider triggers the cookies to be sent
  - detailed profiling can be employed for creating consumer profiles
- Content and ad providers can cooperate for better profiling
  - consumers log in to content providers are are reliably identified
  - their personal profile can be matched with the ad provider's profile
  - ad provider consolidation makes this scenario realistic

Browsers Assemble Web Pages (16)

Typical Web resources (HTML pages) are assembled from a number of resources retrieved by HTTP. Any resource not originating on the server that is hosting the HTML page is considered a "third-party resource". If the HTTP response for such a resource contains a cookie, it is a "third-party cookie".



Cookie-Less State Tracking

Cookie Support (18)

- Authentication can be tracked with HTTP Authentication [Web Foundations (URI and HTTP); HTTP Authentication (1)]
  - this is possible because authentication is built into HTTP
- Other session concepts are not supported by HTTP
  - cookies have become the generic solution for all session tracking
- Cookies are increasingly limited by browsers
  - cookies have gained some notoriety as privacy invaders
  - browsers have more restrictive default settings
  - an increasing number of users restricts cookie support
- Session-oriented Web sites often depend on cookies

URI Rewriting (19)

- Cookie [Cookie (1)]s are a piece of information stored on the client
  - they are sent by the server as a result of a request
  - they are returned by the browser in a response to the same site
- The same information can also be encoded in the URI
  - normally a response contains a cookie and an HTML page
  - the same effect is achieved when all links include the "cookie value"
  - this method often results in very long URIs
- Some Web application frameworks switch automatically
  - J2EE checks for cookie support and switches to URI rewriting if required
- Problems with bookmarks and caches

Hidden Form Fields (20)

- Cookie [Cookie (1)]s transmit session information via HTTP
- URI Rewriting [URI Rewriting (1)] encodes session information in URIs
- HTML Forms [HTML Forms] are a way to send data to a server
  - in most cases this is data that is entered by the user
- Hidden form fields can be used to send data that is part of the HTML
  - hidden form fields are never displayed to the user
  - their predefined values are sent as part of the form submission
- Hidden form fields are essentially the same as URI Rewriting [URI Rewriting (1)]
  - they can only be used if the interaction is based on forms
  - they also require the Web page to be dynamically generated for each request
  - the values end up as URI query string or request entity

Conclusions (21)

- Sessions should only be used for application state
- Cookies are the best way to track sessions
  - cookies should be self-contained rather than referential
- Alternative methods are URI rewriting and hidden form fields
  - more robust than cookies but unpleasant side-effects