State Management

Chapter 6

What Are You Going To Learn?

- At the end of this lesson, you will be able to:
  – Explain HTTP Protocol
  – Differentiate cookie, query string, session variable, application variable and cache to retain information
  – Use Global.asax file
  – Choose and apply appropriate mechanism to maintain state

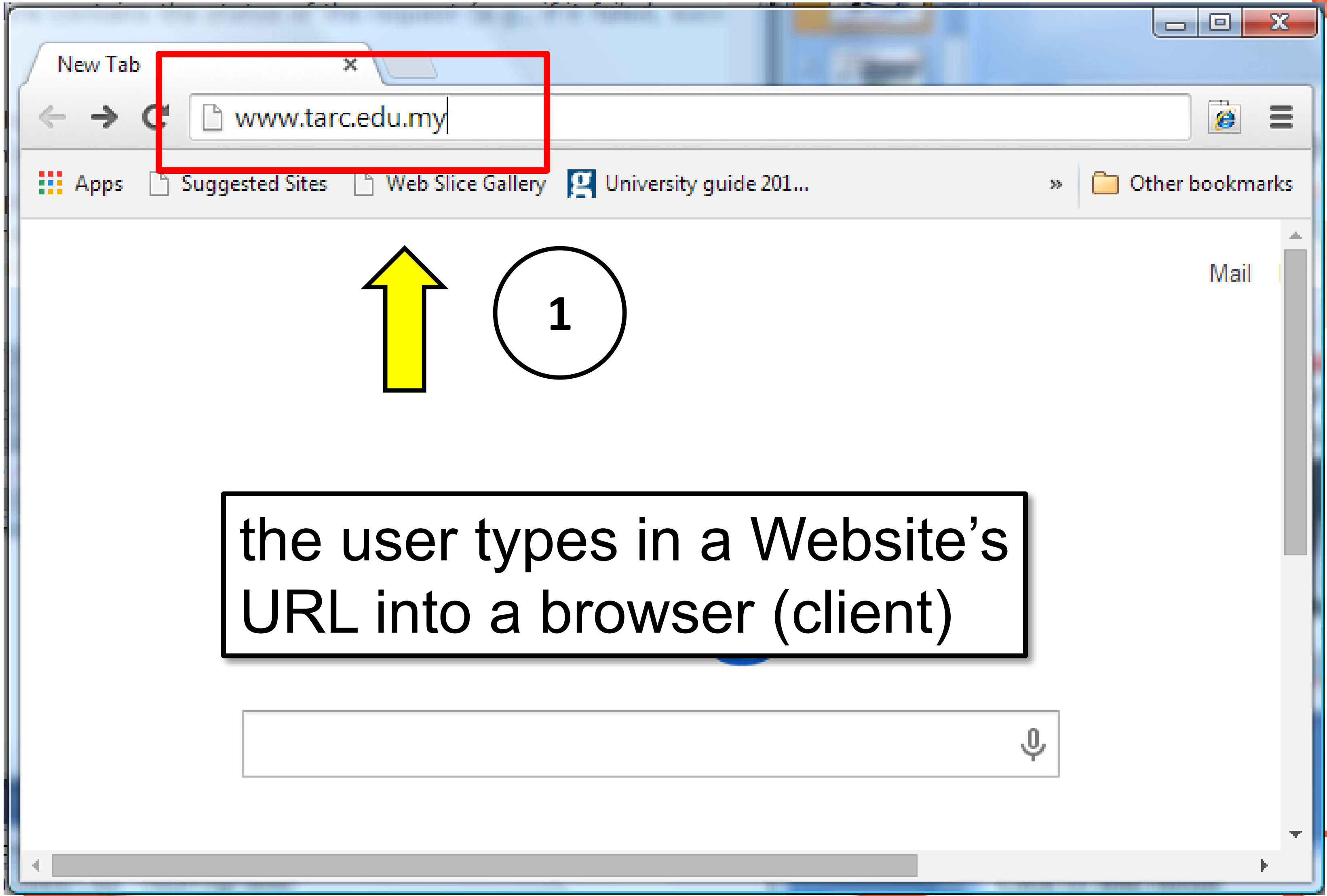
HTTP: HyperText Transfer Protocol

- A communication protocol of the TCP/IP Suit with the Web server, used for retrieving hypertext.
- The most common used protocol is HTTP/1.1
- HTTP is a request/response standard between a client and a server.

HTTP: HyperText Transfer Protocol

- Typically an HTTP client initiates a Request, which establishes a TCP connection to a particular port (usually port 80) on a host (server).
- Upon receiving the request, the server sends back the requested resource as Response.
- Resources to be accessed by HTTP are indentified using Uniform Resource Identifiers (URIs) (i.e. Uniform Resource Locator (URL) using the http: (or https:) URI schemes.

How HTTP works:

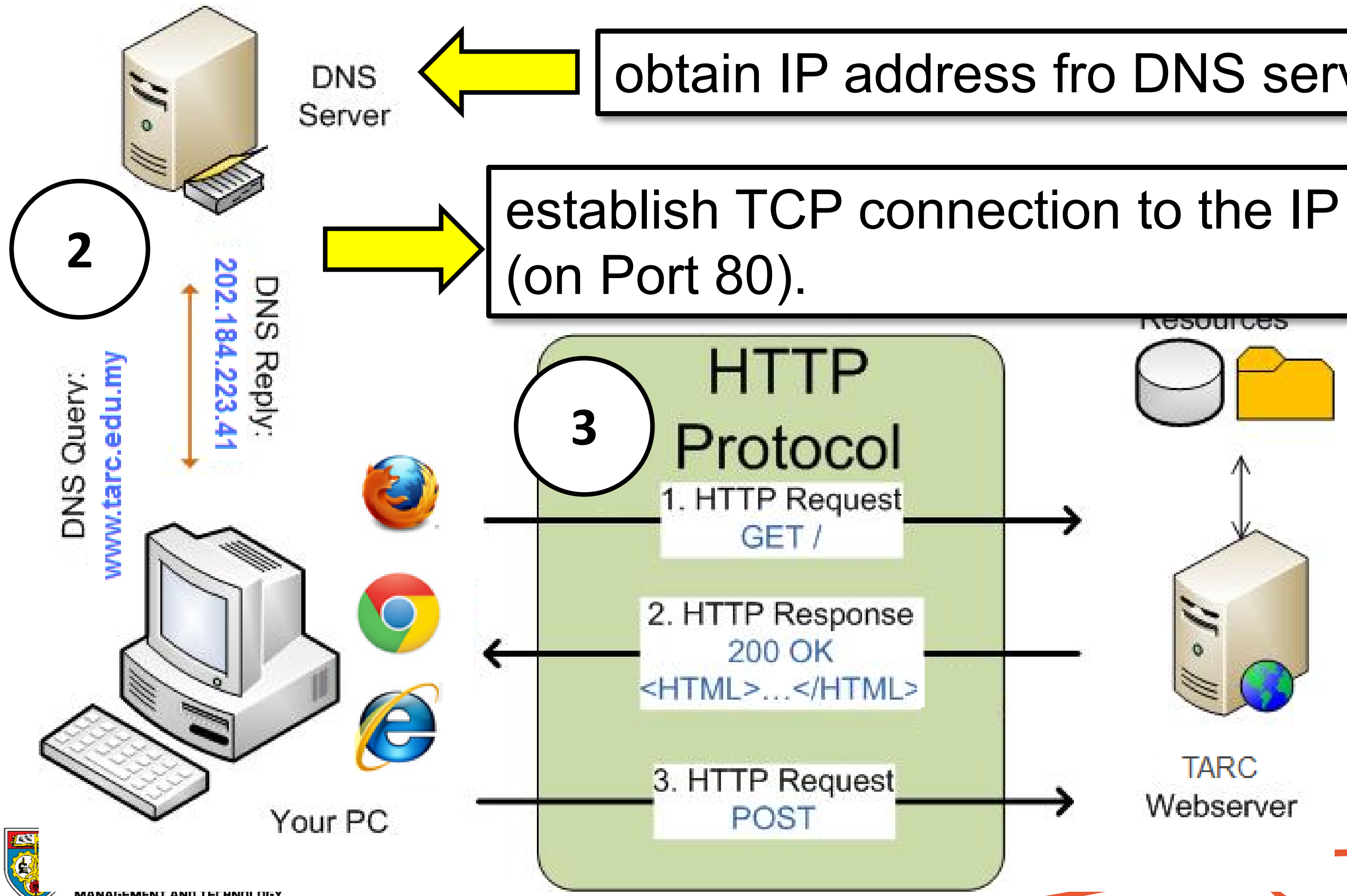


the user types in a Website's URL into a browser (client)

How HTTP works:

obtain IP address fro DNS services

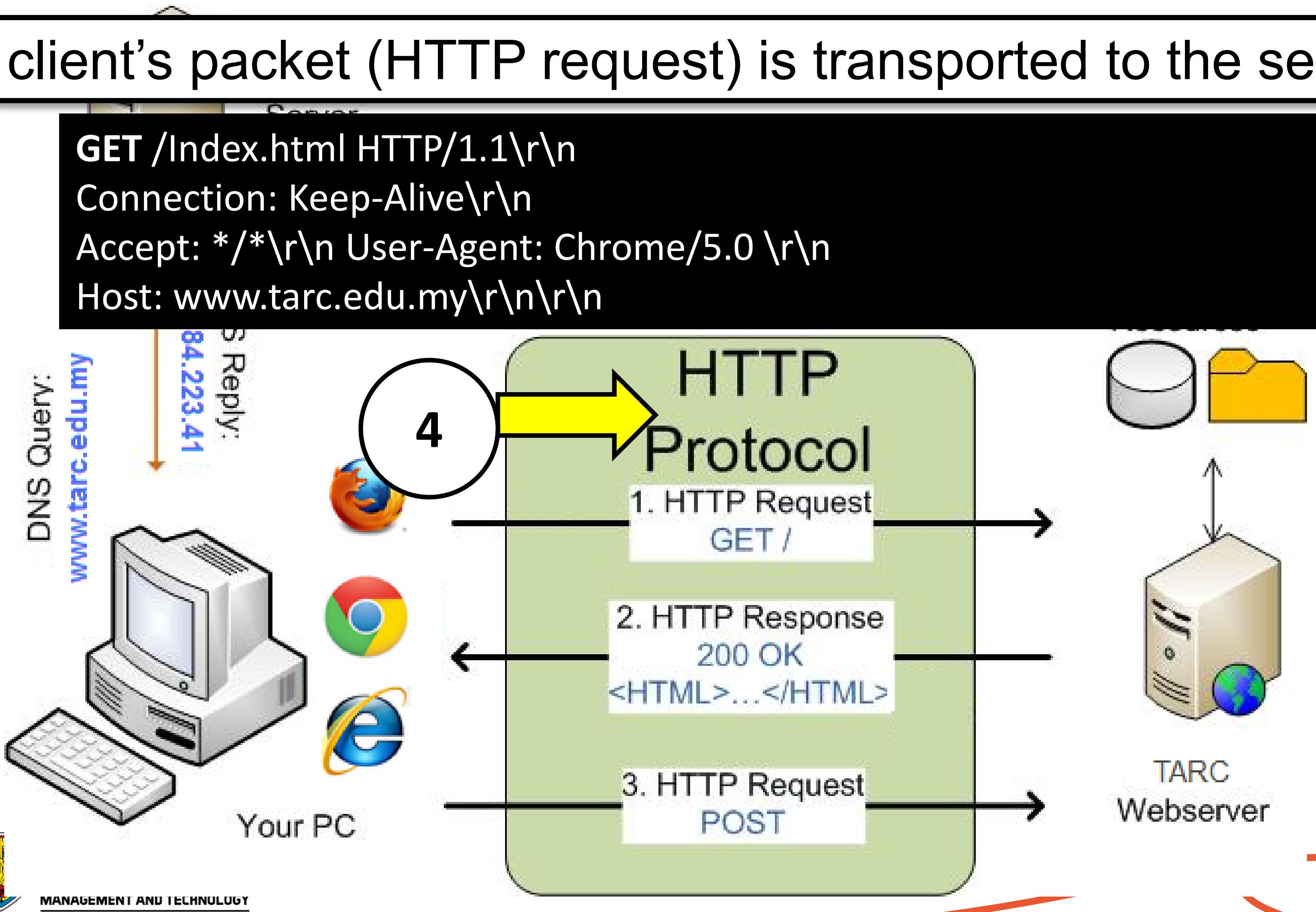
establish TCP connection to the IP (on Port 80).



How HTTP works:

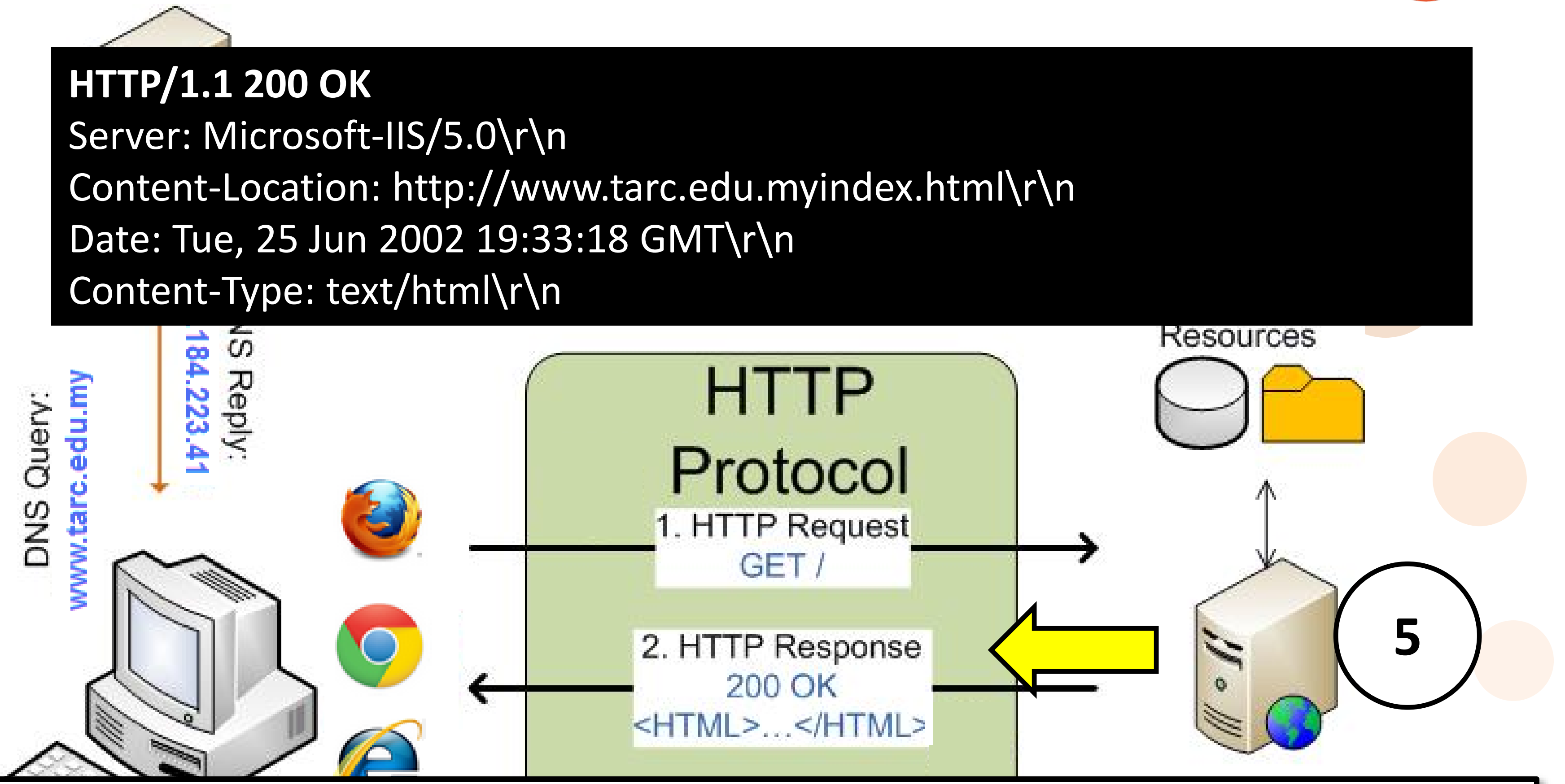
client's packet (HTTP request) is transported to the server

```
GET /Index.html HTTP/1.1\r\n
Connection: Keep-Alive\r\n
Accept: */*\r\n User-Agent: Chrome/5.0 \r\n
Host: www.tarc.edu.my\r\n\r\n
```



How HTTP works:

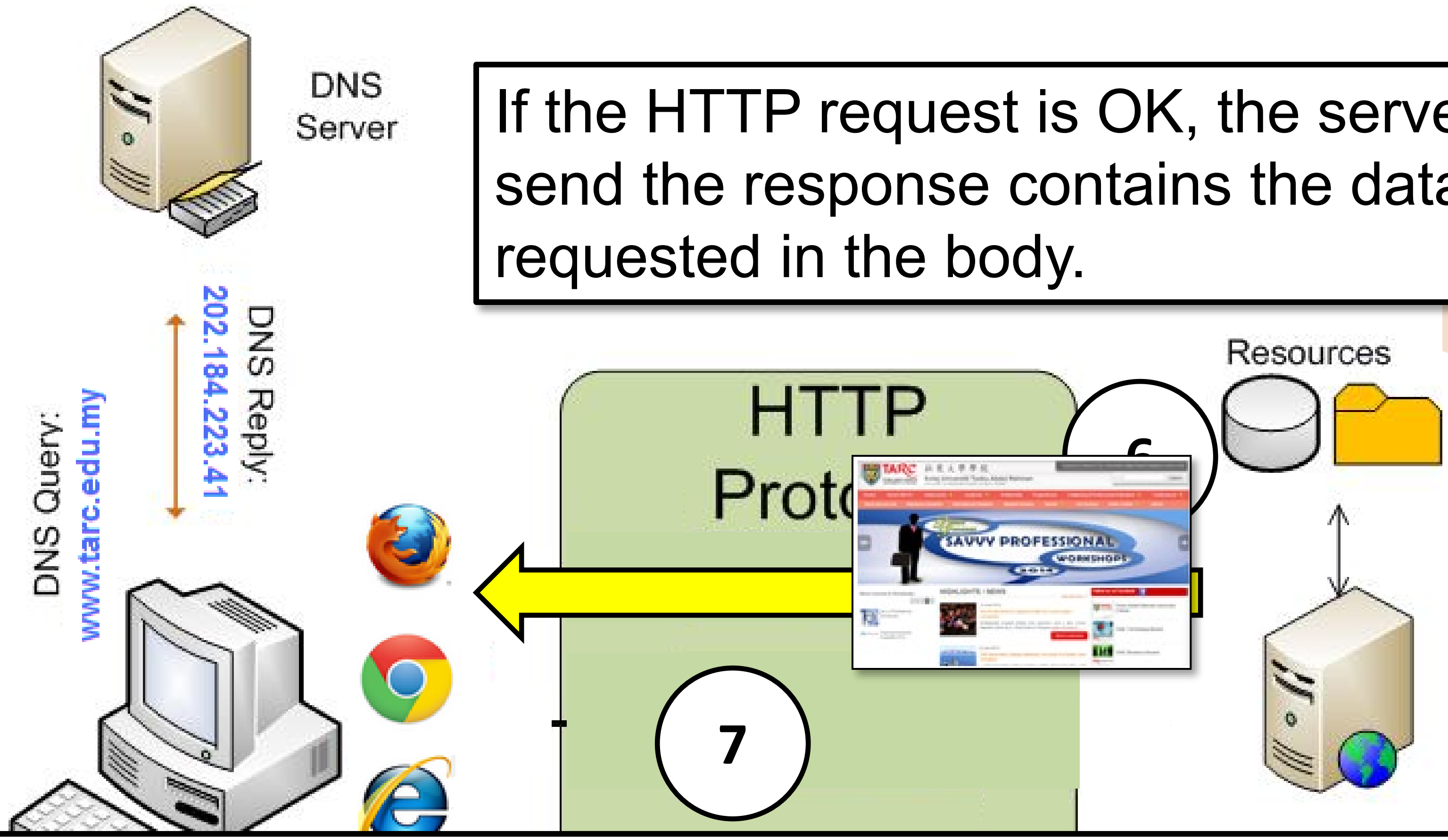
```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0\r\n
Content-Location: http://www.tarc.edu.myindex.html\r\n
Date: Tue, 25 Jun 2002 19:33:18 GMT\r\n
Content-Type: text/html\r\n
```



the server then sends a HTTP response along with HTTP status code (e.g. 200 means OK, 404 means file not found)

How HTTP works:

If the HTTP request is OK, the server will send the response contains the data requested in the body.

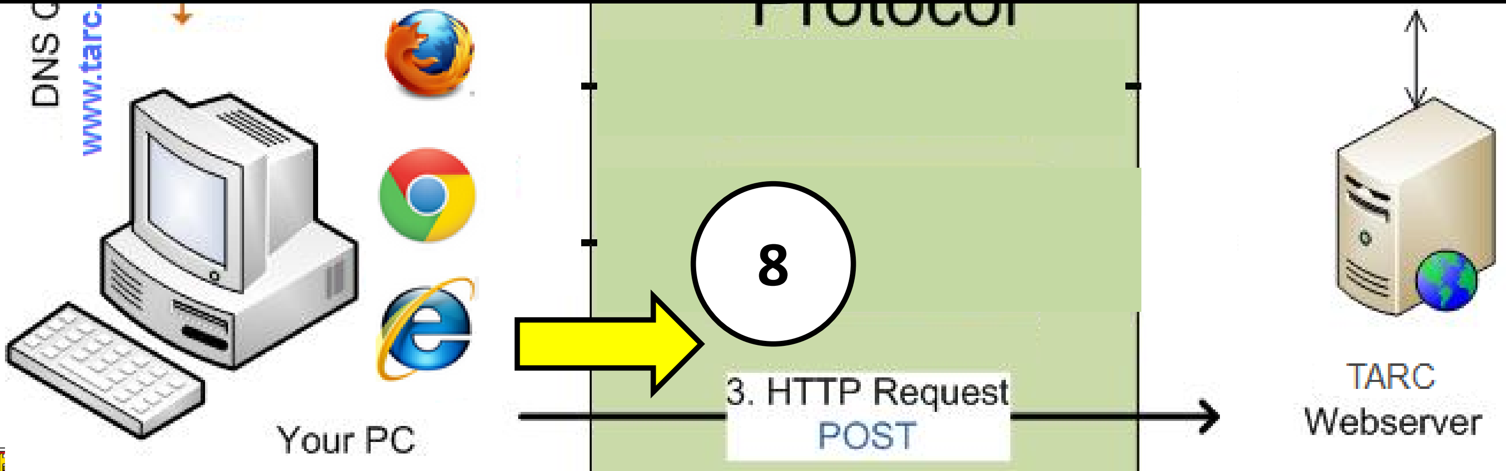


Once the response body has been transmitted, the HTTP server will be **disconnected**

How HTTP works:

The subsequent request (of the same page) will be using POST method, to submit information to the server (e.g. submit enquiry)

```
POST/enquiry.asp HTTP/1.1\r\n
Connection: Keep-Alive\r\n
Accept: */*\r\n User-Agent: Chrome/5.0 \r\n
Host: www.tarc.edu.my\r\n\r\n
```



MANAGEMENT AND TECHNOLOGY

What is state?

- A program stores data in variables (memory locations). The contents of the memory locations at the given point in the program execution, is called **state**.
- State refers to the current status of the properties, variables, and other data maintained by an application for a single user.
- The application must maintain a separate state for each user.

HTTP is stateless

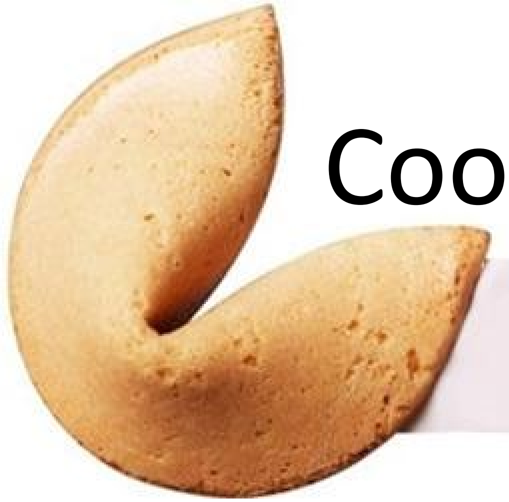
- It **doesn’t keep track of state between round trips**.
- Each request is standalone and considered a new request from a new user.
- Once a browser makes a request that receives a response, the application terminates and its state is lost.

Client-side state management options

- The following are the client-side state management options that ASP.NET supports:
  - View state
  - Control state
  - Hidden fields
  - Cookies
  - Query strings

Server-side state management options

- The following are the server-side state management options that ASP.NET supports:
  - Application state
  - Session state
  - Cache *also available in client
  - Profile properties
  - Database support



Cookies

- Cookies are files created by websites you've visited that store browsing information, such as your site preferences or profile information.
- There are two types of cookies: *First-party cookies* are set by the site domain listed in the address bar. *Third-party cookies* come from other domain sources that have items, such as ads or images, embedded on the page.

TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

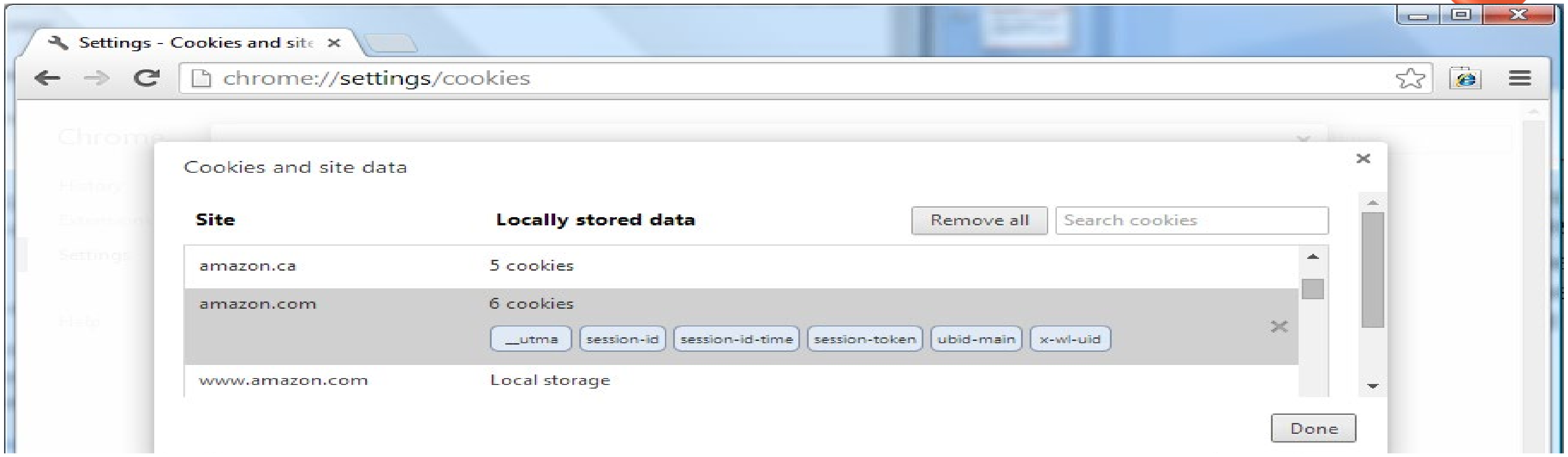
Cookies

- A cookie is a set of properties in the form of name=value pairs (separated by commas)that is stored in the user’s browser or on the user’s disks.

```
username=emily, dogsname=stout, coffee=starbucks;
expires=Sat, 01-Jan-2007 00:00:00 GMT;
```

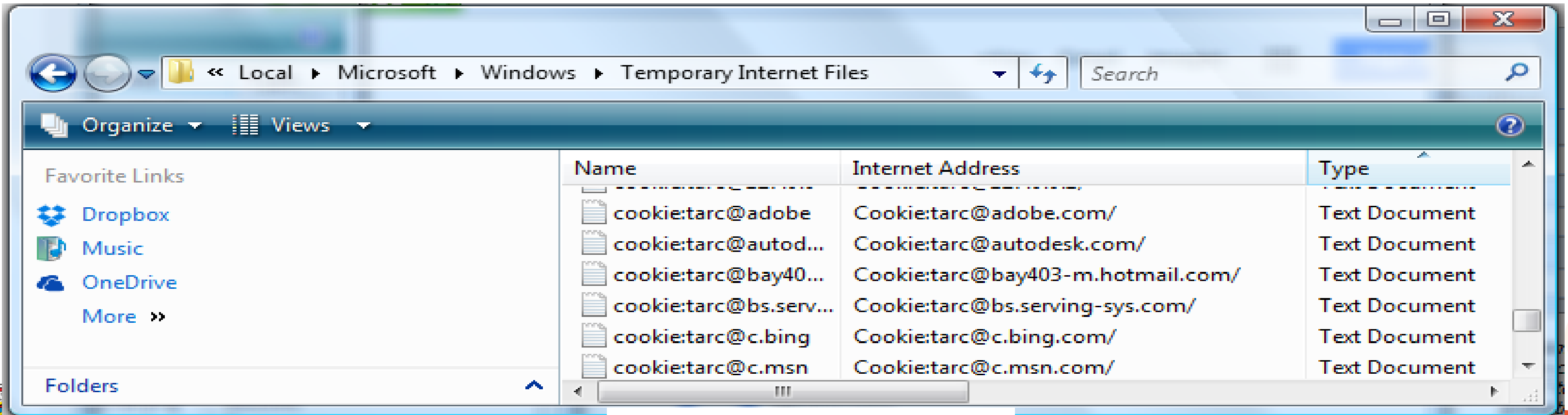
- Can be used by a web server to store information on a client computer and can be retrieved by the same web server only

Viewing cookies from the browser



Site	Locally stored data
amazon.ca	5 cookies
amazon.com	6 cookies: __utma, session-id, session-id-time, session-token, ubid-main, x-wl-uid
www.amazon.com	Local storage

Chrome: Settings\Advanced Settings\Privacy\Content settings\All cookies and site data



Name	Internet Address	Type
cookie:tarc@adobe	Cookie:tarc@adobe.com/	Text Document
cookie:tarc@autod...	Cookie:tarc@autodesk.com/	Text Document
cookie:tarc@bay40...	Cookie:tarc@bay403-m.hotmail.com/	Text Document
cookie:tarc@bs.serv...	Cookie:tarc@bs.serving-sys.com/	Text Document
cookie:tarc@c.bing	Cookie:tarc@c.bing.com/	Text Document
cookie:tarc@c.msn	Cookie:tarc@c.msn.com/	Text Document

internet explorer

Cookies

- A web application sends a cookie to a browser via an HTTP response.
- Then, each time the browser sends and HTTP request to the server, it attaches any cookies that are associated with that server.

Cookies

- A **session cookie** is kept in the browser’s memory and exist only for the duration of the browser session.
- A **persistent cookie** is kept on the user’s disk as a text file and is retained until the cookie’s expiration date.

Cookies usage

- Used to store client preferences, such as:
  - ____________________________
  - ____________________________
- Details of the last visited date and time
- Some user details, such as:
  - user name
  - password?? (should we?)
- Previous search details

Creating cookies

- First method:

```
HttpCookie identifier= new HttpCookie(CookieName);
cookie.Value = SomeValue;
```

- Example:

```
HttpCookie cookie = new HttpCookie ("LastSearch”);
cookie.Value = txtSearch.Text;
```

Creating cookies

- Second method:

```
HttpCookie identifier= new HttpCookie(CookieName,value);
```

- Example:

```
HttpCookie cookie = new HttpCookie ("LastSearch”, txtSearch.Text);
```

Reading a cookie

- Syntax:

```
Request.Cookies[CookieName].Value
```

- Examples:

```
if(Request.Cookies[“LastSearch”] != null)
{
    txtSearch.Text = Request.Cookies["LastSearch"].Value;
}
```

Question: Why is Line 1 (highlighted) important?

TARUMT TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

HttpCookie property

- Expires
  - A DateTime value that indicates when the cookie should expire.
- Name
  - The cookie's name.
- Value
  - The string value assigned to the cookie.

HttpCookieCollection class

- Cookies are managed in collection defined by the HttpCookieCollection class.
- Property
  - Count – The number of cookies in the collection.
- Methods
  - Add(cookie) – Adds a cookie to the collection.
  - Clear( ) – Removes all cookies from the collection.
  - Remove(name) – Removes a cookie.

Advantages of cookies

- Since cookies persist on the client’s computer, space does not need to be allocated on the web server to store user-specific information
- Cookies can save small amounts of information for very long periods of time
- Cookies can be used to customize a user’s visit to your web site

Disadvantages of cookies

- Users can choose not to accept cookies on their Web browsers (they can block the cookies)
- Users can manually delete cookies
- Cookies are unable to save large objects, arrays, or other complex data types. Cookies can only save string, date, or numeric data types

When to use cookies?

- Store small piece of data that are not crucial to your application
- Never use cookies to store sensitive information
- Can be configured to expire after any length of time
- However, cookies can be blocked at the client end.

Query String

- Used in Anchor tags and hyperlinks (URLs) to pass information from one page to the other.
- Query string syntax
  - pass 1 value:

*URL**?name=value***
Example: http://www.shop.com/shop.aspx?prod=tx0002

  - pass >1 value

*URL? **name1=value1&name2=value2***
Example: Order.aspx?cat=1&prod=tx0002

Passing a Query String

- Example 1 (with Anchor tag)

```
<a href="product.aspx?cat=tx&prod=fog01"> Fog machine</a>
```

- Example 2 (with button)

```
<asp:Button PostBackUrl = "shop.aspx?prod=tx0002"
ID="btnShop" runat="server" />
```

- Example 3 (Response.Redirect)

```
Response.Redirect("Order.aspx?cat=" + txtcategoryID.Text);
```

Query String

- Statements that retrieve the values of the query string attributes

```
Request.QueryString[“name”]
```

- Example:
  - The following query string is passed to “product.aspx”

```
<a href=“product.aspx?cat=tx&prod=fog01”> Fog machine</a>
```

  - In “product.aspx”, retrieve the values

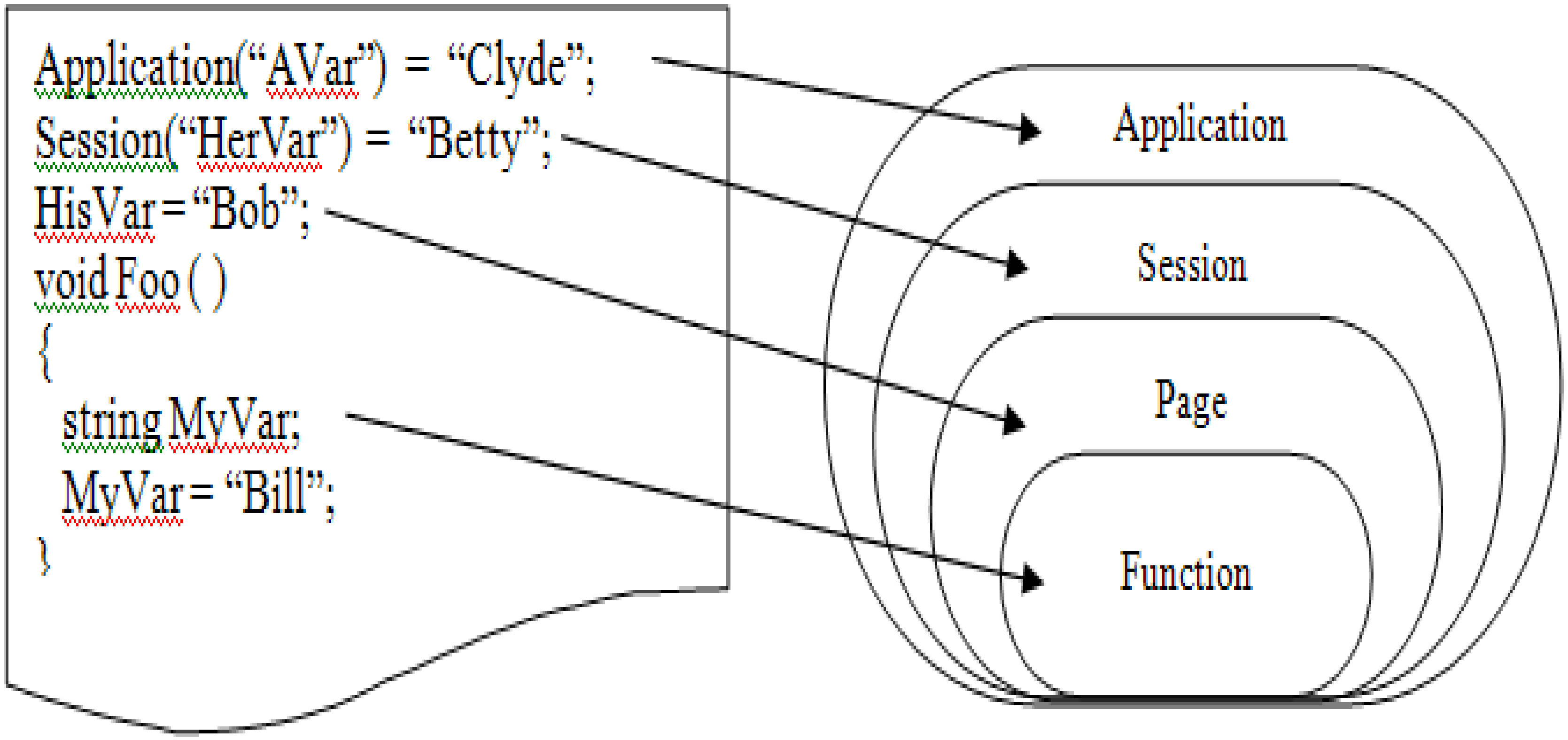
```
string categoryID = Request.QueryString[“cat”];
string productID= Request.QueryString[“prod”];
```

TARUMT
TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

Understanding scope

- Function
  - Visible within the function
- Page
  - Visible within the page
- Session
  - Visible from page to page during a session
  - Session variables can be used in all the pages in an application for a particular session
- Application
  - Visible throughout the whole application lifetime

Understanding scope (cont')



```
Application("AVar") = "Clyde";
Session("HerVar") = "Betty";
HisVar="Bob";
void Foo ( )
{
  string MyVar;
  MyVar= "Bill";
}
```

What is session?

- Time spent browsing a site, from the moment you first start browsing to the moment you close your browser (Persists its value until a session ended or user leaves the website)
- Each visitor is assigned an individual session (Data stored in session variables are not shared among different users)
- A new instance of browser is considered a new session
- A session object is created for each session

Creating session variables

Can store any type

- Syntax
  - Session[“*ItemName*”] = *Content*;
- Examples

```
Session[“username”]=“Patrick”;

Session[“email”]=txtEmailAddress.Text;

Session[“quantity”] = 100;

Session[“StudentList”] = studentList;
```

TARUMT TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

Retrieving session variables

- Session variable used to store objects (in any type)
- Type casting is needed to assign the value of a session object to a variable.
- Examples

```
lblUserName.Text = Session[“username”].ToString();
int quantity = Convert.ToInt32(Session[“quantity”]);
List<string> studList = (List<string>)Session[“studentList”];
```

TARUMT TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

Controlling when a session ends

- Default timeout – 20 minutes
- Set timeout to other values by using Timeout property (in minutes)

```
Session.Timeout = 60; //set session timeout to 60 mins)
```

Session methods (cont')

- Add(name, value)
  - Adds an item to the session state collection.
- Clear( )
  - Clears all values from the session but leaves the session active
- Remove(name)
  - Removes an item
- Abandon( )
  - Force to end the current session

When to use sessions?

- Used to maintain user-specific information
- Can store any variable type
- Best choice when you need to maintain state only for the user's visit to your site.
- However, if you receive many concurrent users or place large object in the Session object, your Web server performance will degrade

Application objects

- Designed to maintain state globally, across the entire website
- Application variables can be accessed by all the pages and by all the users of the application
- Similar to Session, it can store any data types including objects and arrays

Application usage

- To store global application data such as discount terms and tax rates.
- Display tips of the day or news update
- Record the number of times a banner advertisement was clicked
- Record the running count of the visitors

Creating application variables

- Syntax
  - Application[“*ItemName*”] = *Content*;

Can store any type

- Examples
  - Application[“TipsOfTheDay”]=“Failing in the past does not mean you will fail again in the future”;
  - Application[“visitor”] = 0;

TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

Retrieving application variables

- Application variable used to store objects (in any type)
- Type casting is needed to assign the value of a application object to a variable.
- Examples

```
lblDisplay.Text = Application[“TipsOfTheDay”].ToString();

int visitorCount = (int)Application[“visitor”];
```

Application Methods

- Add(name, value)
  - Adds an item to the application state collection.
- Clear( )
  - Removes all items from the application state collection.
- Remove(name)
  - Removes a particular item.

Application Methods

- Application.Lock( )
  - Locks the application state collection so only the current user can access it.
- Application.UnLock( )
  - Unlocks the application state collection so other users can access it.

Pessimistic concurrency—when one user starts updating a record, lock it, thereby preventing any other users from editing or deleting that record until the user commits their modifications.

Application Property

- Count
  - The number of items in the application state collection.

Working with application events

- Global.asax is a code-only file that provides event handlers for responding to application events.
- Only scripts and objects are allowed
- Cannot include scripts that produces output (HTML tags or Response.Write method)

Application events

- Application_Start
  - Raised when the first page of an application is requested by any user.
  - Raised when web server restarted
  - Raised when Global.asax file is edited
- Application_End
  - Raised when application is about to terminate.
- Application_Error
  - Raised when an unhandled error occurs

Application events (cont')

- Session_Start
  - Raised when a user starts a session
- Session_End
  - Raised when a user session ends
- Profile_OnMigrateAnonymous [Not covered]
  - Raised when an anonymous user logs in, and allows migration of any Profile properties

DEMO

Application

When to use applications?

- Used to maintain information that is global to the entire Web site.
- Should NOT use to maintain state on a user-by-user basis
- Can drain your server’s resources
- Can’t use to store anything you need to keep

Caching

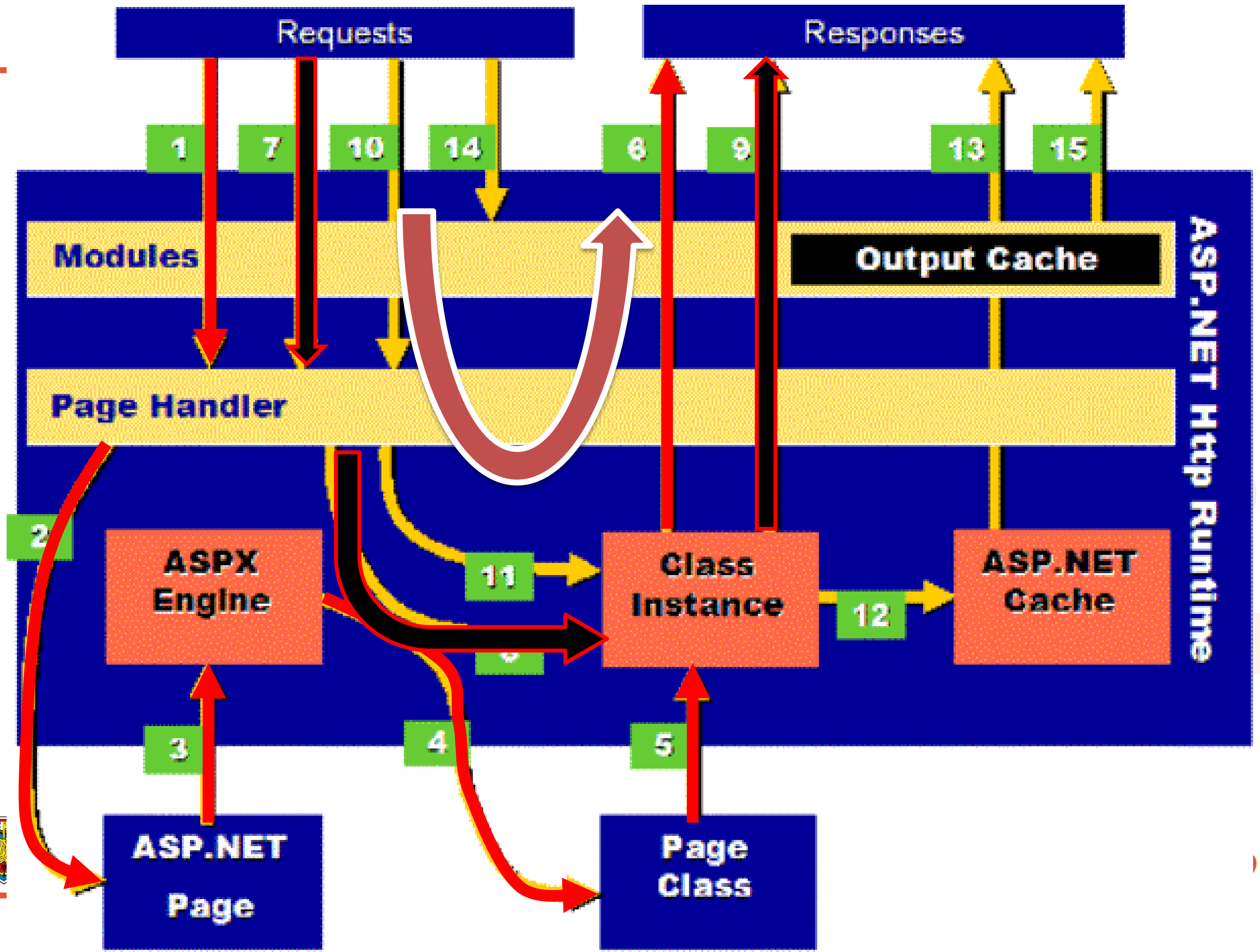
- *Caching* is the process of storing frequently used data on the server to fulfill subsequent requests.
- Three types of caching
  - – Output caching / Page output caching
  - – Fragment caching / Partial page caching
  - – Data caching [Not covered]

TARUMT TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

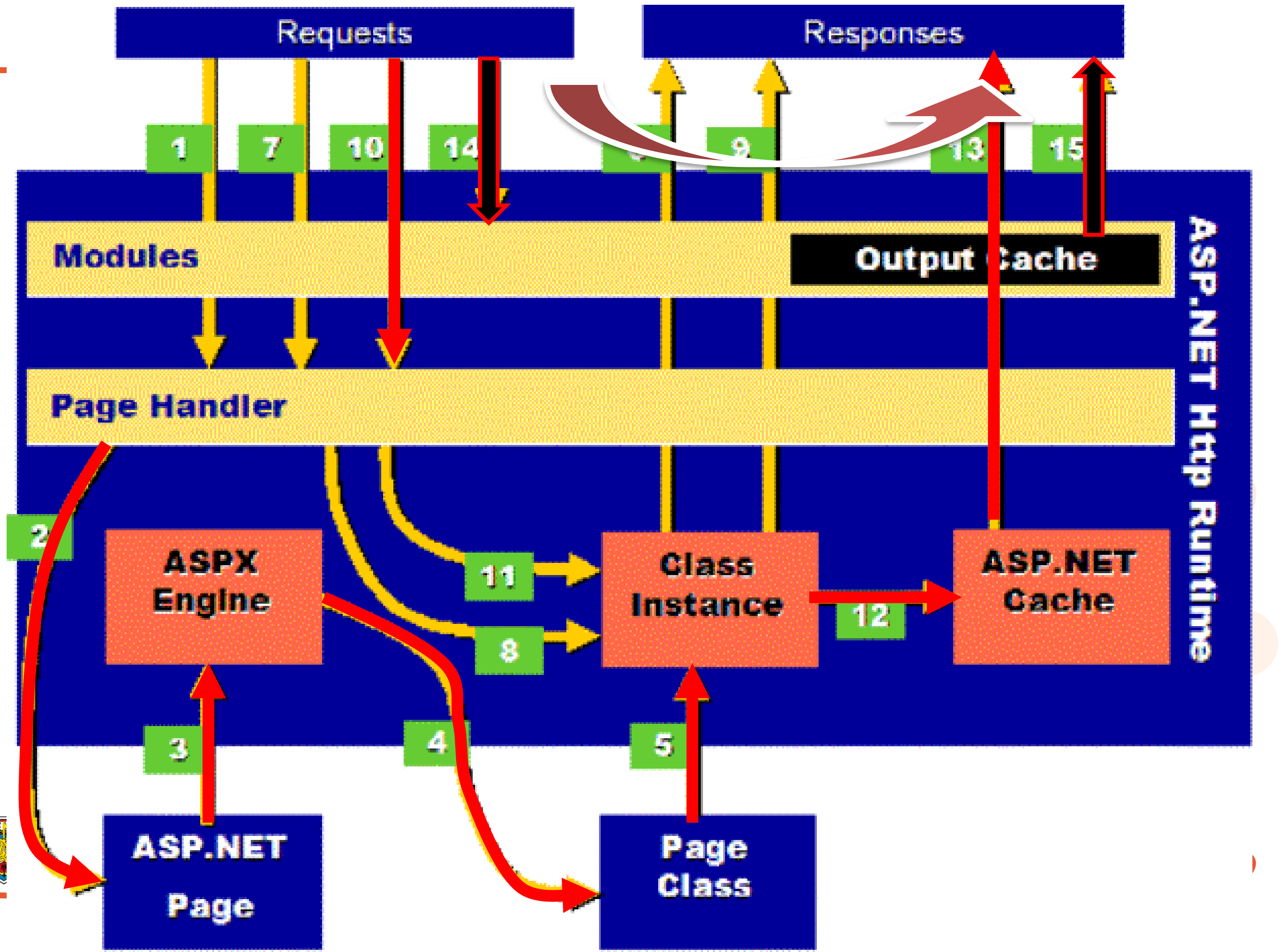
Output Caching

- Allows the entire contents of a page to be persisted to memory and used to fulfill client requests.
- This type of cache saves post-rendered content so it won‘t have to be regenerated again the next time it’s requested.
- After a page is cached, it can be served up again when any subsequent requests are made to the server.

Without Cache



With Cache


Requests
Responses
1
7
10
14
9
13
15
Modules
Output Cache
Page Handler
ASP.NET Http Runtime
2
ASPX Engine
11
8
Class Instance
12
ASP.NET Cache
3
4
5
ASP.NET Page
Page Class

Output Caching

- <%@ OutputCache Duration="60" VaryByParam="None" %>
- Duration attribute - defines the number of seconds a page is stored in the cache.
- You must include either the VaryByParam attribute or the VaryByControl attribute.
- However, if you do not need to vary your cached output by control or parameters, define VaryByParam ="None".

Cache multiple versions of the same page in output cache

- The VaryByParam attribute allows you to vary the cached output depending on the query string.
- The VaryByControl attribute allows you to vary the cached output depending on a control value.

Cache multiple versions of the same page in output cache

- The VaryByHeader attribute allows you to vary the cached output depending on the request's HTTP header.
- The VaryByCustom attribute allows you to vary the cached output by browser type or by a custom string that you define.

Fragment Caching

- Partial page caching allows parts of a page to be cached and other parts to be dynamic.
- Achieved with the caching of user control.
- By enabling the Shared = "true" attribute, the UserControl's output can be shared among multiple pages and on sites

When to use output caching?

- The generated page changes every few hours as information is loaded into a database.

When to use fragment caching?

- The generated page generally stays the same, but there are several tables shown within the output that change regularly.

next

VALIDATION CONTROLS