



TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

Maintaining state through multiple forms



HTTP – stateless protocol

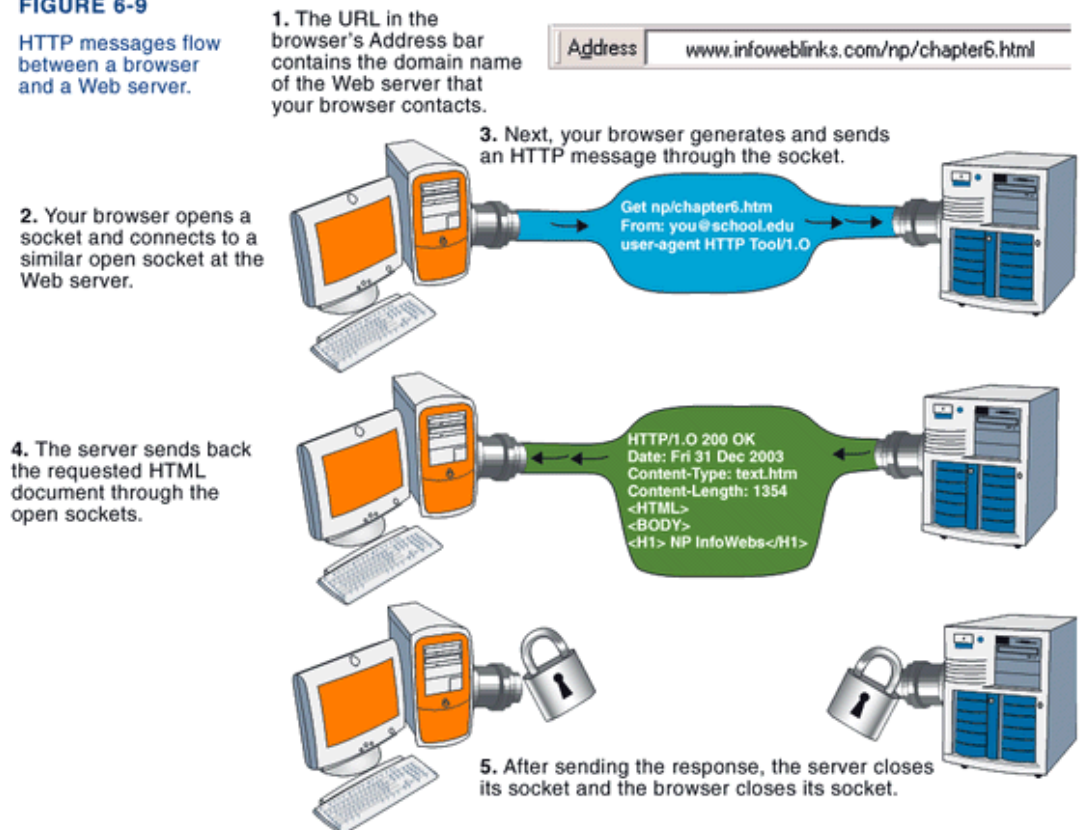
HTTP is a **stateless protocol**

→ Once a web server completes a client's request for a web page, the connection between the two goes away.

→ There is no way for a server to recognize that a sequence of requests all originate from the same client.

FIGURE 6-9

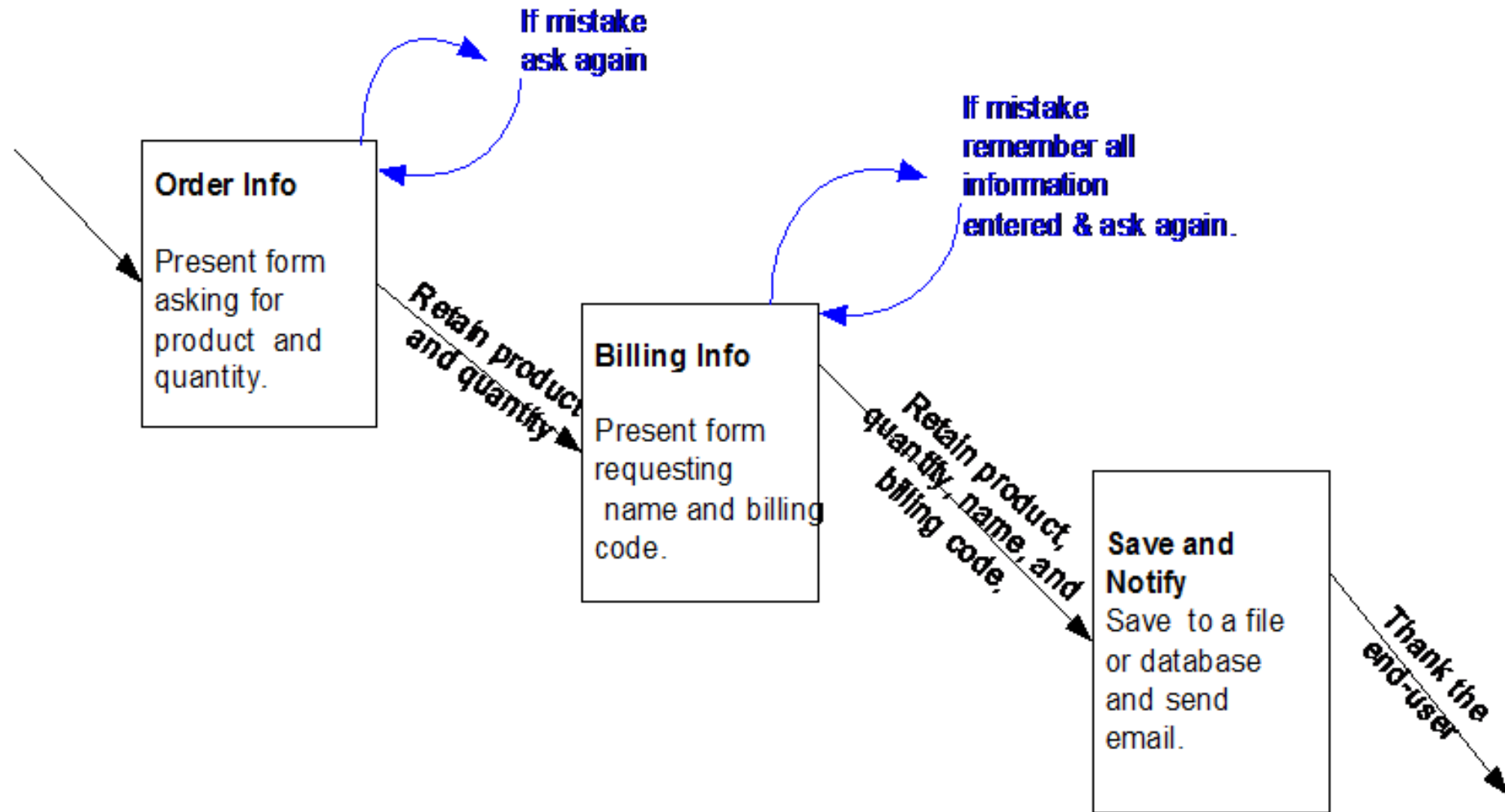
HTTP messages flow between a browser and a Web server.



What Are Multiple-Form Web Sessions?

- A multiple-form Web session leads the user through a series of HTML forms that work together and pass data from form to form.
- E.g.
 - To build a shopping cart or on-line survey.
 - To save user authentication information from page to page
 - To store persistent user preferences on a site

Example Multiple Screen Session



How to maintain the state through multiform?

- Use tricks to keep track of state information between requests (session tracking)
 - Using hidden form fields
 - URL rewriting: every local URL on which the user might click is dynamically modified to include extra information
 - `http://www.example.com/catalog.php?userid=123`
 - Using cookies: a bit of information that the server give to a client → depends on the client
 - Using session

Content



1. Hidden fields

2. User browser cookies


3. PHP session

1. Hidden fields

- Hidden fields are part of HTML forms
 - Not displayed but value can be accessed in receiving script like any other variable.
 - Can still be viewed by user's who view source.

```
<input type="hidden" name="preference" value="Likes Power Tools">
```

**Variable name
that will be available
to your PHP program**



**Variable's value in the
PHP program**



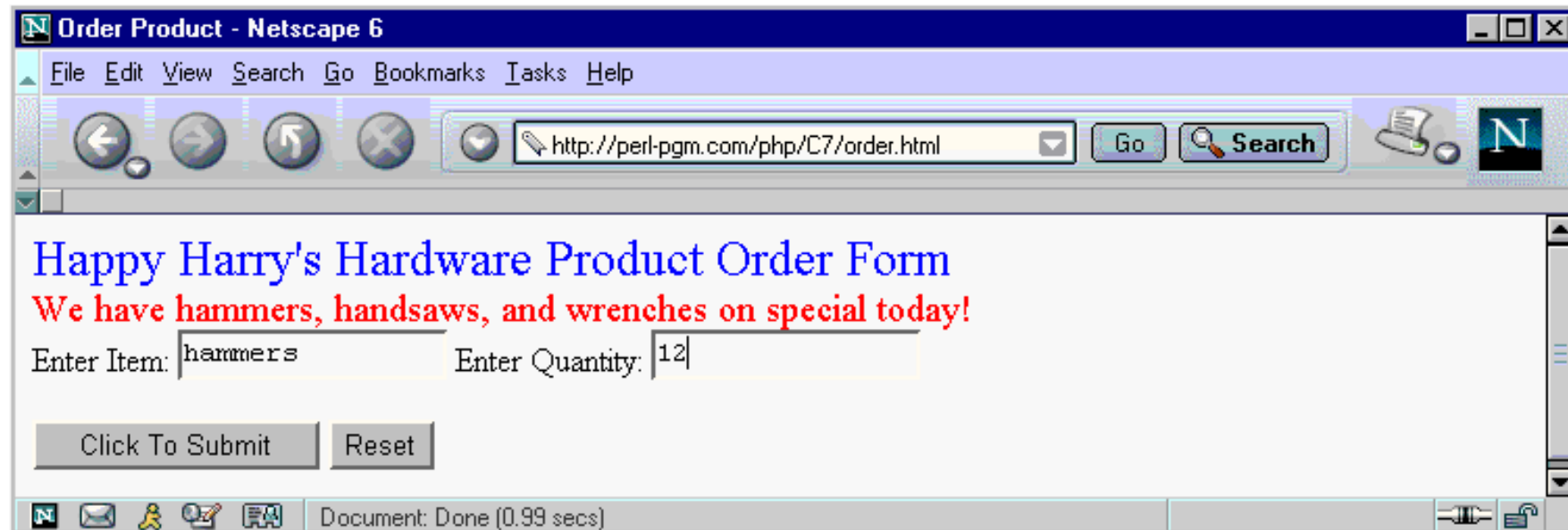
A Full Script Example

- Consider an example script sets a hidden field
 - Implements the Order Info form
 - On submit sends data to order2.php

PHP Script – order.html

```
1. <!DOCTYPE html><html><head><title>Order Product</title></head><body>
2. <form action="order2.php" method="post">
3. <h1> Hardware Product Order Form</h1>
4. <p class="highlight">
5. We have hammers, handsaws, and wrenches on special today!
6. </p>
7. <input type="hidden" name="sample_hidden" value="Welcome"><br/>
8. Enter Item: <input text type="text" size="15" maxlength="20"
   name="product">
9. Enter Quantity: <input text type="text" size="15" maxlength="20"
   name="quantity"><br/><br/>
10. <input type="submit" value="Click To Submit">
11. <input type = "reset" value="Reset">
12. </form></body></html>
```

The Output ...



Receiving Hidden Fields in Web Sessions

- Your scripts can receive data from hidden fields like any other data.
- File: order2.php

```
1. <html><head><title> Order Product 2 </title> </head>
2. <body>
3. <form action="order3.php" method="post">
4. <?php $sample_hidden = $_POST["sample_hidden"];
5. $product = $_POST["product"]; $quantity = $_POST["quantity"];
6. print "<p class='highlight'>";
7. print "Hidden value=$sample_hidden </p><br>";
8. print "You selected product=$product and quantity=$quantity";
```

Receiving PHP Script

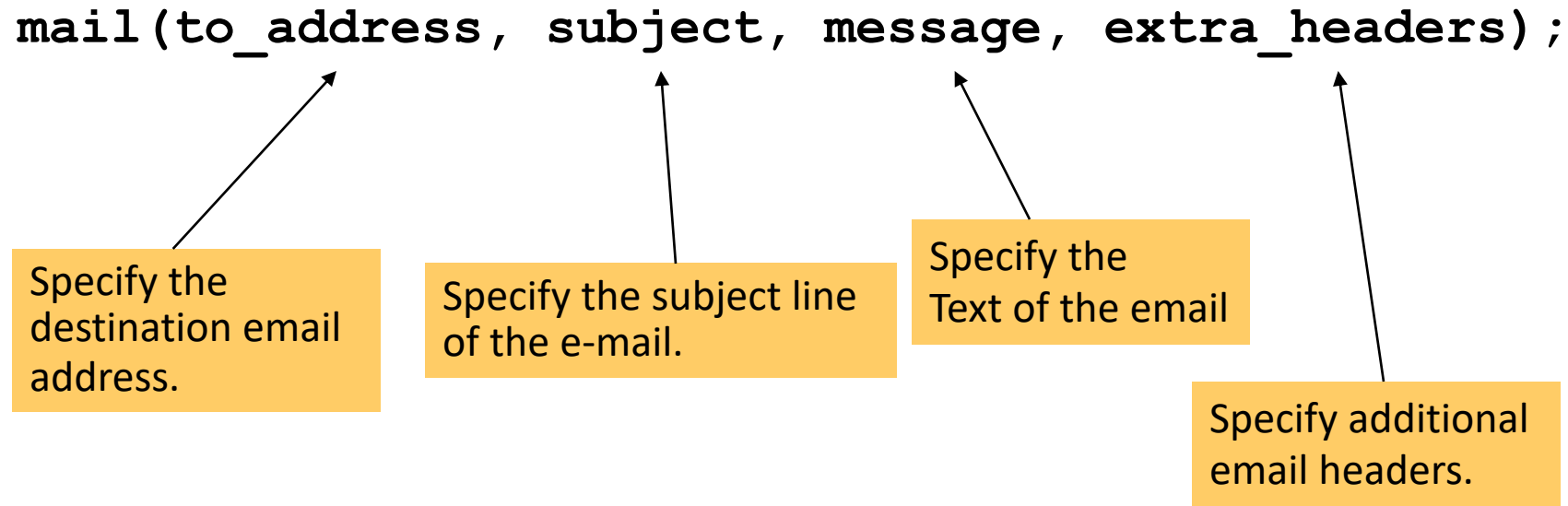
```
9. print "<br><br><input type=\"hidden\" name=\"product\" value=\"$product\"> ";
10. print "<input type=\"hidden\" name=\"quantity\" value=\"$quantity\">";
11. print "<input type=\"hidden\"
    name=\"sample_hidden\"value=\"$sample_hidden\">";
12. print 'Please enter your name: ';
13. print '<input type="text" size="15" maxlength="20" name="name">';
14. print ' and billing code: (5 digits)';
15. print '<input type="text" size="5" maxlength="5" name="code">';
16. print '<br/> <input type=submit value="Process Order">';
17. print '<input type=reset>';
18. ?></form></body></html>
```

Sending email from PHP scripts

- Sometimes it is useful to send email from a PHP script:
 - PHP uses mail() that by default sends e-mail via the Simple Mail Transfer Protocol (SMTP).

```
mail(to_address, subject, message, extra_headers);
```

Specify the
destination email
address.

A diagram illustrating the parameters of the PHP mail() function. Four yellow boxes with black text are positioned below the function signature. Arrows point from each box to its corresponding parameter: from 'Specify the destination email address.' to 'to_address', from 'Specify the subject line of the e-mail.' to 'subject', from 'Specify the Text of the email' to 'message', and from 'Specify additional email headers.' to 'extra_headers'.

Specify the subject line
of the e-mail.

Specify the
Text of the email

Specify additional
email headers.

Consider the following example ...

```
1. $dest='orders@hardwareville.com';  
2. $subject = 'New Hardware Order';  
3. $message = 'Enclosed is a new order for 12 hammers.\n Thanks.';  
4. $extra = 'From: harry@hardwareville.com';  
5. mail( $dest, $subject, $message, $extra );
```

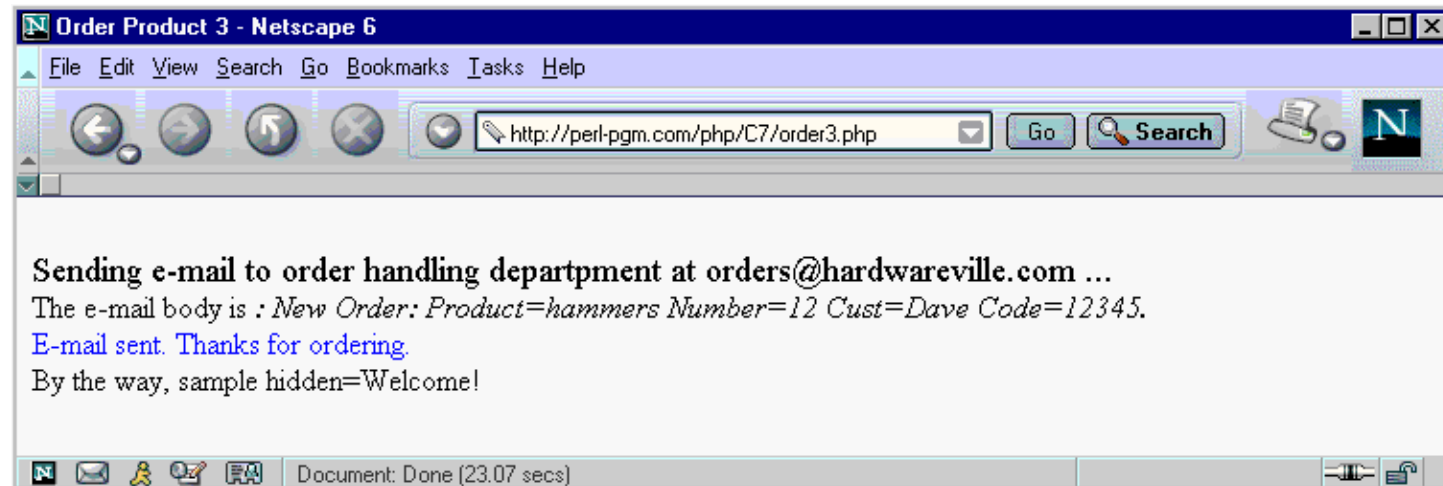
Consider the following full example ...

- Implements save and notify
- Called from order2.php and saved at order3.php
- Can access variables \$product, \$quantity, and \$sample_hidden sent as hidden fields from the Billing Info form.

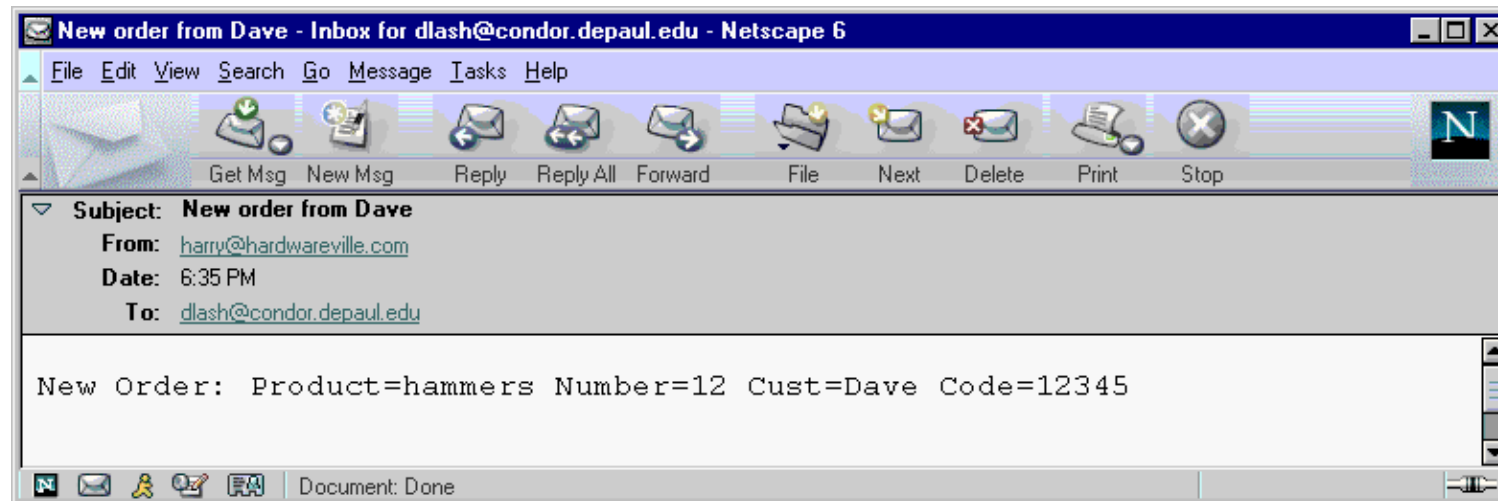
The following PHP Script ...

```
1. <html><head><title>Order Product 3</title> </head><body>
2. <?php
3. $sample_hidden = $_POST["sample_hidden"]; $quantity=$_POST["quantity"];
4. $product = $_POST["product"]; $name=$_POST["name"];
5. $email='orders@hardwareville.com';
6. $body = "New Order: Product=$product Number=$quantity Cust=$name Code=$code";
7. print '<font size=4>';
8. print "<br>Sending e-mail to order handling department at $email ...
   </font>";
9. print "<br>The e-mail body is <i>: $body. </i>";
10. $from = 'harry@hardwareville.com';
11. $subject = "New order from $name";
12. mail($email, $subject, $body, "From: $from");
13. print '<br><font color="blue"> E-mail sent. Thanks for ordering. </font>';
14. print "<br>By the way, sample hidden=$sample_hidden";
15. ?></body></html>
```


Would have the following output ...



Would have the following output ...



Content

1. Hidden fields
- 2. User browser cookies
3. PHP session

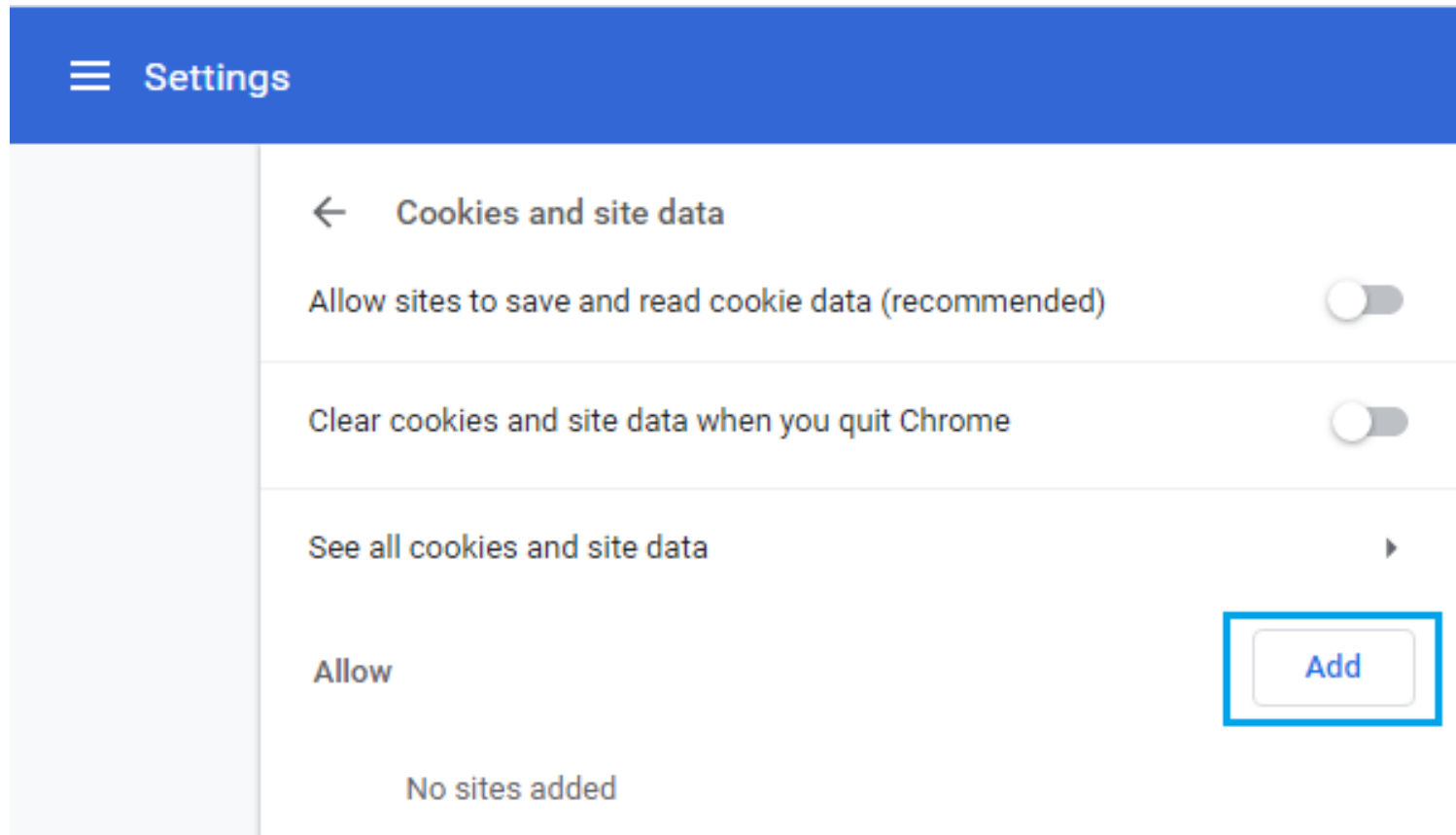
Using Browser Cookies ...

- Cookies are small pieces of data that a Web application can save when a user visits the Web page.
 - Stored on the visitor's hard drive
 - a Web page script can read the previously stored browser cookie data

Understanding Cookie Limitations

- Users can easily disable the cookies feature.
- Users may delete cookies.
- PHP sets limit on cookies

The disable cookie screen in Netscape



Setting and Reading Cookies

- A cookie is often used to identify a user.
- A cookie is a small file that the server embeds on the user's computer
- Each time the same computer requests a page with a browser, it will send the cookie too.

```
setcookie('Customer_name', 'Denise');
```

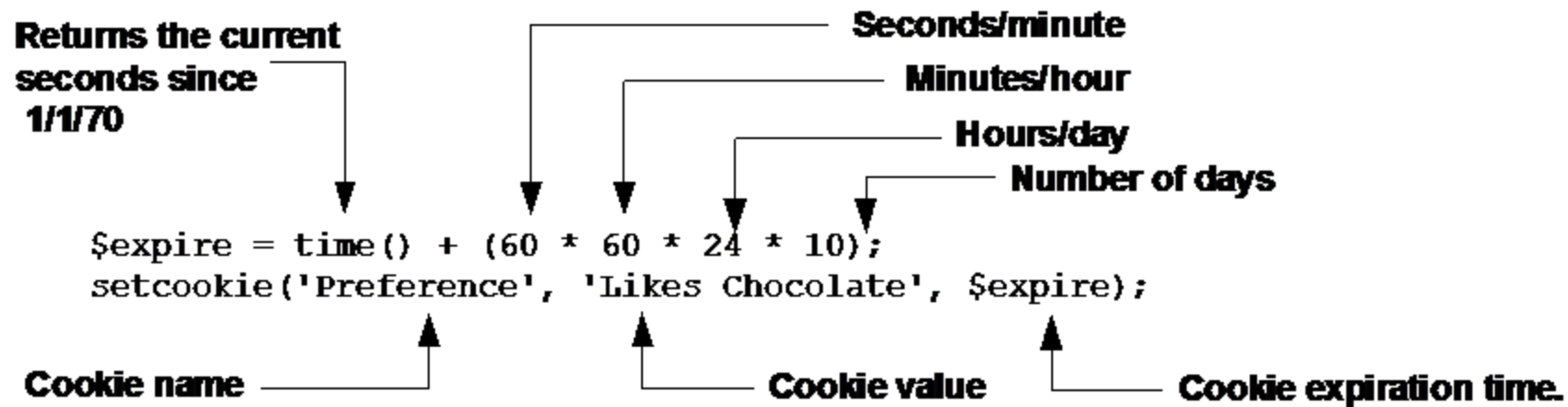
Directs browser
to create a cookie

Specify the
cookie's name

Specify the
cookie's value

Setting A Cookie on a Hard Drive

- You need to use the time() function when want to set a cookie on a hard drive.



A full example of setting a cookie....

- Suppose a front-end web page asks for some survey information:

```
<input type="text" size="15" maxlength="20" name="custname">
```

```
<input type="radio" name="prefers" value="power tools" checked > Power Tools?
```

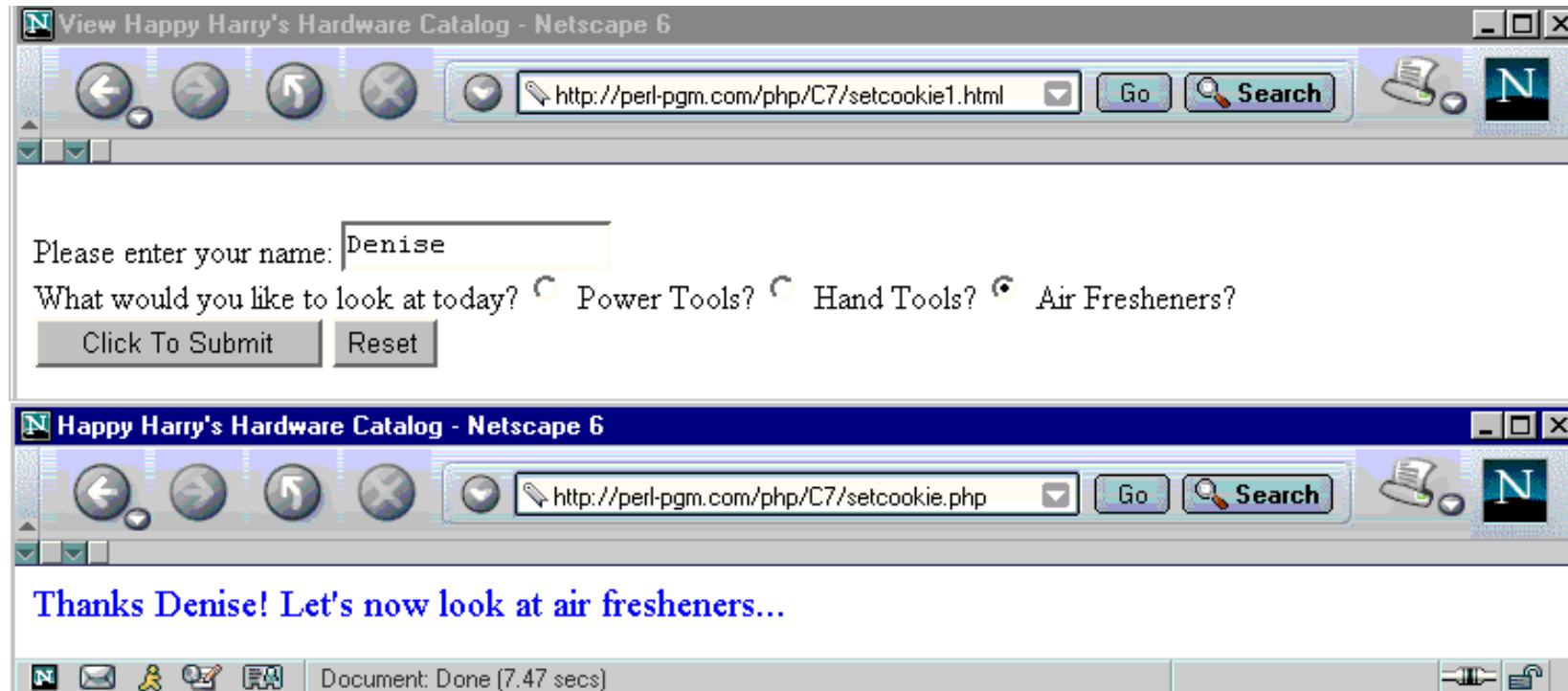
```
<input type="radio" name="prefers" value="hand tools"> Hand Tools?
```

```
<input type="radio" name="prefers" value="air    fresheners"> Air Fresheners?
```

The script runs when submitted setcookie.php

```
1. <?php $prefers = $_POST["prefers"];      $custname=$_POST["custname"];
2.     $expire = time() + (60 * 60 * 24 * 30);
3.     setcookie("custname", $custname, $expire);
4.     setcookie("preference", $prefers, $expire);
5. ?>
6. <html>
7. <head><title>Happy Harry's Hardware Catalog </title></head>
8. <body><font size=4 color="blue">
9. <?php
10.         print "Thanks $custname! ";
11.         print "Let's now look at $prefers... ";
12. ?> </font></body></html>
```

Would output:



Reading Cookies

- You can read a cookie by using a variable name with the same name as a cookie:
 - `print "$cust_name";`

Reading Cookies

- To read a cookie value use the `$_COOKIE[]` associative array to get the cookie function
- `$cust_name= $_COOKIE["cust_name"];`

Example Script that read a cookie

```
1. <html>
2. <head><title>Happy Harry's Hardware Catalog</title>
3. </head><body>
4. <?php $name = $_COOKIE["custname"]; $preference = $_COOKIE["preference"];
5.     print '<font color="blue" size=4>';
6.     if (isset($name)){
7.         print "Welcome back to our humble hardware site, $name.";
8.     } else {
9.         print '<font color="red">';
10.            print 'Welcome to our humble hardware site.</font>';
11.        }
12.        if ($preference == 'hand tools'){
13.            print '<br> We have hammers on sale for 5 dollars!';
14.        } elseif ($preference == 'power tools'){
15.            print '<br> We have power drills on sale for 25 dollars!';
16.        } elseif ( $preference == 'air fresheners'){
17.            print '<br> We now carry extra-strength air fresheners!';
18.        } else {
19.            print '<br> <font color="red">';
20.            print 'We have drills and hammers on special today!';
21.        }
22.    ?></font></html>
```

Content

1. Hidden fields
2. User browser cookies
- 3. PHP session

PHP Sessions

- PHP enables you to retain data between forms
 - `session_start()`
 - starts a new session
 - This function must be the very first thing in your document. Before any HTML tags.
 - Global variable `$_SESSION`
 - `$_SESSION['name'] = $name;`

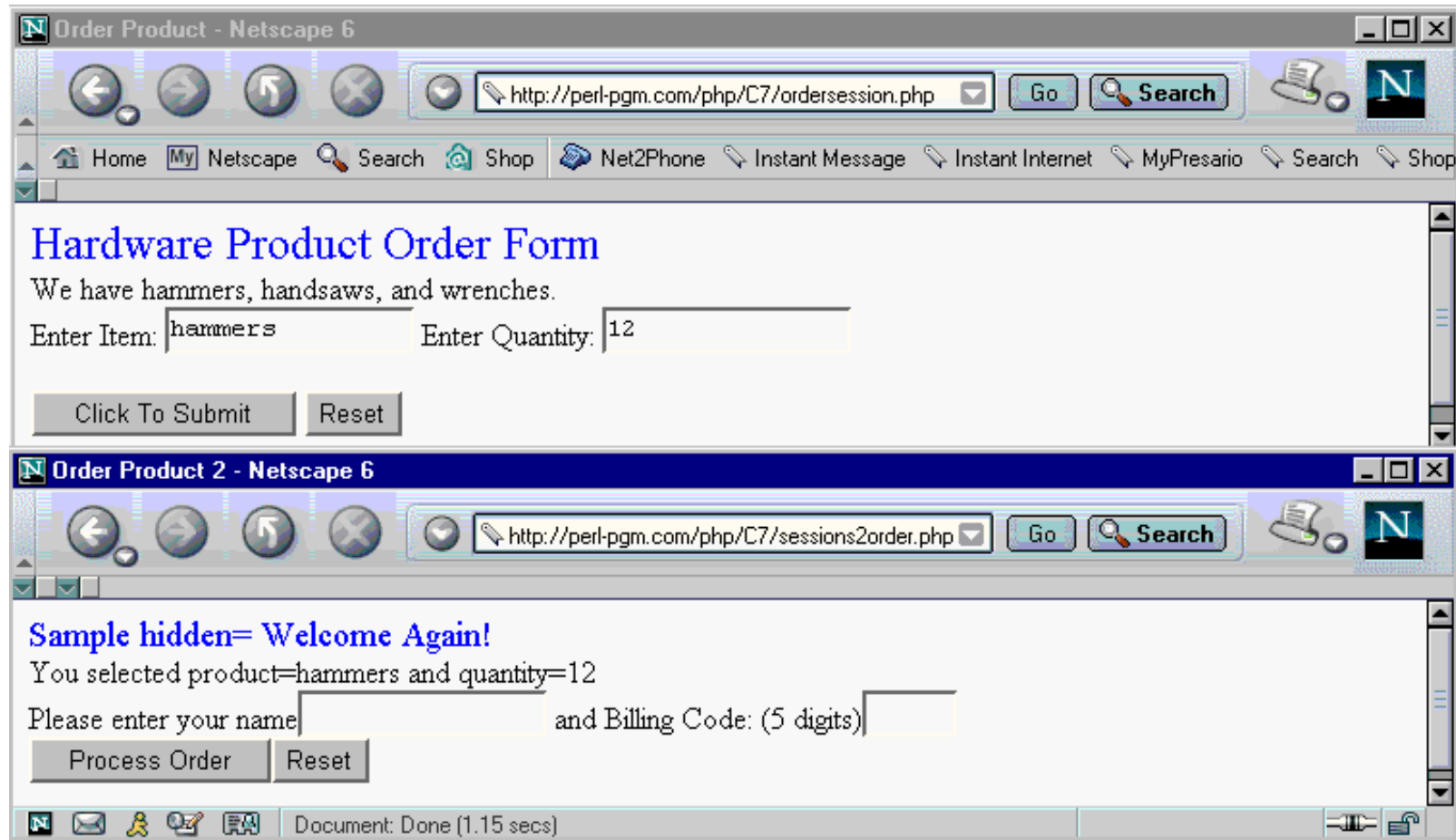
Example PHP Code

```
<?php session_start(); ?>
<html>
<head><title>Order Product</title></head>
<body>
    <form action="session.php" method="post">
        <font color="blue" size="5"> Hardware Product Order Form </font><br/>
        We have hammers, handsaws, and wrenches.<br/>
        Enter Item: <input text type="text" size="15" maxlength="20" name="product">
        Enter Quantity: <input text type="text" size="15" maxlength="20"
        name="quantity"><br/>
        <?php
            $sample_hidden='Welcome Again!';
            $_SESSION["sample_hidden"] = $sample_hidden;
        ?>
        <br/>
        <input type="submit" value="Click To Submit">
        <input type = "reset" value = "Reset" >
    </form>
</body>
</html>
```

sessions.php

```
<?php session_start();?>
<html><head><title> Order Product 2 </title> </head>
<body>
<form action="session2.php" method="post">
<?php
    $sample_hidden = $_SESSION["sample_hidden"];
    print "<h1> Sample hidden= $sample_hidden</h1>";
    $product = $_POST["product"];
    $quantity = $_POST["quantity"];
    print "<br>You selected product=$product and quantity=$quantity";
    $_SESSION["product"] = $product;
    $_SESSION["quantity"] = $quantity;
    print '<br>Please enter your name';
    print '<input text type="text" size="15" maxlength="20" name="name">';
    print ' and Billing Code: (5 digits)';
    print '<input text type="text" size="5" maxlength="5" name="code">';
    print '<br> <input type=submit value="Process Order">';
    print '<input type=reset>';
?>
```

Example output



Some session extras

- isset- can be used to determine if a variable comes from a session:

```
if (isset($_SESSION['name'])) {  
    print "got name=$name from session";  
} else {  
    print "name=$name not set from session";  
}
```

Summary

- Hidden fields are HTML form fields you can use to set a variable name and variable value without displaying them on a form.
- Cookies provide a way for Web server applications to store small pieces of data on the user's hard disk.
- PHP provides a convenient way to retain data between PHP scripts.
 - Use `session_start()` and `$_SESSION` to start sessions and define session variables, respectively

Question?

