Walter Schlosser
CSCI446
Unit8

1) HTTP stores no information between pages and requests. This makes it stateless. Variables used for one request are not visible to any other requests.

2) A session id is an encrypted value stored on the client. The id is used to identify which session belongs to the user. A session hash object is created with the id, and values can then be stored. Values in the session hash are persisted, which means they are available across requests, and therefore add state to the app.

3) A session is created when a user visits the webpage. They get a unique session id which is used to look up the correct session hash object when it is referenced in controllers or views. A new session may be created when the user leaves and then visits again. A cookie is a package of data that is stored in the clients browser. This data is persisted across all visits, and therefore is not deleted when the user leaves the page. The application can store the session hash to the cookie so it will be available next time if necessary. Data is stored as a cookie when it must be available between visits to the web app, and a session is used for data storage when data must be available between pages during a visit.

4) -Cookies are only 4kB in size. For this reason, if you are going to store a session, it must contain only simple data, such as database id's. A session that has a complex ruby object should not be stored.

   -Data in a stored session is not encrypted, and so the client will be able to access it. For this reason, a stored session should not have any secrets used by the app for security.

5) CSRF stands for cross site request forgery. When a request is sent to a server, the browser will always send any cookie associated with the domain. This means if the user is logged on to a site and then visits another site, this other site can send requests with the cookie to the site the user logged in on. Since the cookie has the correct session id, the requests will be allowed even though the user was unaware they sent the request. This method can be used by malicious sites to send requests on the user's behalf without them knowing provided their authenticated session is still active.

6) Rails generates a security token to protect the user from CSRF. Every time a form is displayed to the user which would modify a resource, a new security token is created and stored in the session on the server. The token is also hidden in the form the user sees. When the user submits the form, the server compares the security token on the form to the one in the session and will only complete the request if they are the same. In this way, a malicious site cannot send a POST/PUT/DELETE request to another site even if the user's session is still open since their is no way the malicious form can send the correct security token. A GET will still go through, but if the programmer follows RESTful architecture, a GET should never modify a resource.

7) Whitelisting and blacklisting refers to allowing or not allowing certain values in a request. The params hash is where all data sent for the request is stored. A controller usually calls a helper function to get the needed params from the hash. In this function, the params hash has the

function ‘permit’ called on it to make sure only data we expected was sent. This is whitelisting certain fields. A field not in the permit call is blacklisted, and if it is sent, the app will not allow the data to be used.