1. Goal

Understand the finer points of using cookies and sessions within a REST API.

2. Lesson Notes

Do Sessions (and session cookies) violate REST?

- the simple answer is **yes** - because they make the API stateful

- however, in practice, you may trade strict adherence to that constraint with security and performance

- so, until we get to Module 6 and JWT - we can look at sessions as a necessary compromise

Basic Auth, Digest Auth

- we started looking at Basic Authentication - which is stateless but has other issues

- an optional next step would be Digest Authentication, but the overall model this security protocol attempts to use is not very realistic, so we'll skip it for the practical implementation

Form-based Auth

- uses cookies and the session => not stateless

- besides the issue with statelessness, when cookies are by themselves used for auth - the solution will be vulnerable to CSRF attacks

- this leads us along the path of looking for a better solution to this problem and towards a token based approach where we're no longer using cookies for authentication (but we can certainly use them for storage)

Cookies and CSRF

- cookies **used as the primary authentication mechanism** are inherently vulnerable to CSRF attacks

- using tokens sent via HTTP Headers is not

- we can still use cookies as a storage mechanism only

OAuth 2

- OAuth tokens are explicitly session identifiers

Authentication Mechanisms

- stateless: Basic Auth, Digest Auth

- stateful: form-based, OAuth (OAuth tokens are explicitly session identifiers)

- stateless: JWT (Server Signed Tokens) - sending all data to the client

- of course, sending a self-contained token to the client means you need to make sure it doesn't get manipulated => signing

Conclusion

Having a good understanding of the advantages and tradeoffs of each solution is very important and it's perfectly OK to chose the solution that fits your API and your system best.

3. Resources

- Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet