a)

Was ist Cross-site request forgery?

Cross-site request forgery oder kurz CSRF ist ein Angriff gegen Browserapplikationen, wobei eine bösartige App die Interaktion zwischen dem Client Browser und einer Applikation, die diesem Client Browser vertraut, beeinflussen kann. Die Bösartige App benutzt den vertrauenswürdigen Client Browser, um Systemverändernde Aktionen (wie z.B. ein Geldtransfer) auf der ins Ziel genommenen Seite zu tätigen.

Wie lässt es sich in ASP.Net Core unterbinden?

- Es werden automatisch Antiforgery Tokens für HTML form Elemente generiert, wenn der <form> Tag das method="post"-Attribut enthält und das Action-Attribut entweder leer (action="") oder nicht vorhanden ist (<form method="post">).
- Außerdem wird jedesmal Antiforgery middleware zum Dependency Injection Container hinzugefügt, wenn einer der folgenden APIs in Startup.ConfigureServices aufgerufen wird: AddMvc, MapRazorPages, MapControllerRoute, MapBlazorHub
- Zudem generiert IHtmlHelper.BeginForm standartmäßig anyforgery tokens, wenn die Methode der form nicht GET ist.

Was ist der wesentliche Unterschied zwischen GET und POST Anfragen? Gibt es weitere Anfragetypen?

GET enthält die Anfrageparameter angefügt im URL-String, während POST die Parameter im message body enthält, was eine sicherere Methode für den Datentransfer zwischen Client und Server darstellt.
Weiter Anfragetypen sind: PUT, HEAD

b)

Wie lange ist ein Objekt im Cache verfügbar?

Bis entweder die gesetzte Sliding Expiration oder die Absolute Expiration eintrifft.

Was ist der wesentliche Unterschied zwischen dem Cache und einer Datenbank?

Ein Cache ist enthält nur temporär Daten, während eine Datenbank auf unabsehbare Zeit Daten speichern sollte.

Ist der Cache Nutzerspezifisch oder Applikationsweit? Falls letzteres zutrifft: Worauf sollte geachtet werden?

Er ist Applikationsweit. Es sollte darauf geachtet werden, dass Sessions „sticky" sind, wenn in-memory Cache benutzt wird. „Sticky" Sessions sorgen dafür, dass aufeinanderfolgende Anfragen von einem Client alle zum selben Server geleitet werden.

Was ist Dependency Injection?

Es ist eine Technik, um die Abhängigkeiten zwischen Objekten zu vereinfachen, indem es diese Abhängigkeiten zentral zur Verfügung stellt. So müssen Objekte, falls sie ein anderes Objekt für ihre Initialisierung benötigen, es nicht mehr selbst erzeugen.

e)

Warum ist eine beidseitige Validierung eine gute Idee?

Client-seitige Validierung verhindert eine unnötige Anfrage an den Server, falls eine Maske falsch ausgefüllt wird. Um sicher zu gehen, dass keine internen Fehler auftreten, wird trotzdem noch eine Server-seitige Validierung benötigt.