**1. Mis on Middleware?**

Middleware on tarkvara, mis võimaldab hajutatud võrgus kahe või enama rakenduse või rakenduskomponendi vahel ühte või mitut tüüpi suhtlust või ühenduvust. Lihtsustades rakenduste ühendamist, mis ei ole loodud üksteisega ühenduse loomiseks, ning pakkudes funktsioone nende arukaks ühendamiseks, lihtsustab rakenduste arendamist ja kiirendab turule jõudmise aega.

**2. Missuguseid extensione meetodeid kasutab?**

Run- laiendusmeetod, mis lisab rakenduse päringuahelasse terminali vahevara delegaadi. Lisatakse päringute konveieri lõppu.

Map- hargneb päringute kontuuri antud taotluste vaste põhjal. Kui taotlus algab antud teega siis haru täidetakse. Kasutatakse sissetulevate päringute alusel päringu ümbersuunamiseks erinevatele vahevara komponentidele.

Use- lisab reas määratletud vahevara delegaadi rakenduse päringuahelasse. Saab lisada järgmise parameetri, mis kutsub järgmise päringu delegaadi ette. Täidab ülesandeid enne ja pärast järgmist delegaati, ühendades komponente.

**3. Iseloomusta Middleware käskude loogikat.**

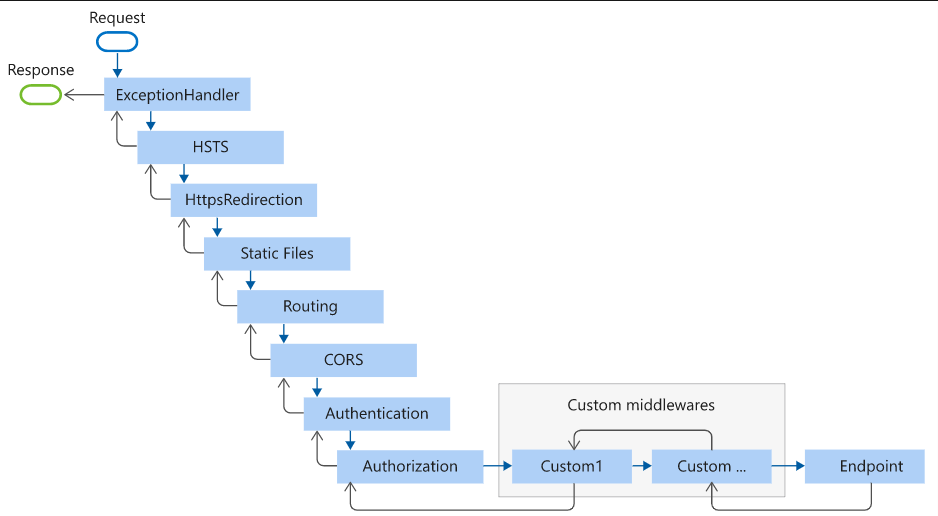
Iga delegaat saab teha toiminguid enne ja pärast järgmist delegaati. Erandite käsitsemise delegaadid tuleks välja kutsuda varakult, et nad saaksid tabada erandeid, mis esinevad konveieri hilisemates etappides.

Lihtsaim võimalik rakendus ASP.NET Core seadistab ühe taotluste delegaadi, kes tegeleb kõigi päringutega. See juhtum ei hõlma tegelikku päringute kontuuri. Selle asemel kutsutakse vastuseks igale HTTP -päringule üks anonüümne funktsioon.

**4. Kuidas kutsutakse delegaati, kui see ei edasta päringut (request) järgmisele delegaadile?**

Kui delegaat ei edasta päringut järgmisele delegaadile, nimetatakse seda short-circuiting the request pipeline. Lühis on sageli soovitav, kuna see väldib tarbetut tööd.

**5. Kirjelda täieliku päringuprotsessi järjestust ASP.NET Core MVC näite puhul.**

****

Vahevara komponendid käivitatakse nende konveierile lisamise järjekorras ning tuleb jälgida, et need lisatakse õiges järjekorras. Vastasel juhul ei pruugi rakendus ootuspäraselt toimida. Esimene konfigureeritud vahevara on taotluse vastu võtnud, muudab seda (vajadusel) ja annab kontrolli järgmisele vahevarale üle. Lõpp-punkt käivitab filtrite jada vastava rakenduse tüübi jaoks (ressursifiltrid, mudeli sidumine, mudeli valideerimine, toimingud filtrid, erandi filtrid, tulemuste filtrid).

**6. Kuidas mõjutab middleware koodi järjestus komponeneti tööd?**

Middleware komponentide järjekord määratakse käivitamisel. Konfigureerimismeetod määrab, millises järjekorras middleware komponente päringutel esile kutsutakse, ja vastupidine järjekord määratakse andmete tagasisaatmise puhul. Järjekord on turvalisuse, jõudluse ja funktsionaalsuse seisukohalt kriitilise tähtsusega.

**7. Mida teeb Developer Exception Page Middleware, UseExceptionHandler, UseHsts, UseHttpsRedirection, UseStaticFile, UseCookiePolicy, UseRouting, UseAuthentication, UseAuthorization, UseSession ja UseEndpoints koos MapRazorPages, UseSpaStaticFiles?**

**Developer Exception Page Middleware**- teatab rakenduse käitustõrkest

**UseExceptionHandler**- püüab erandeid, mis on visatud järgmisesse middleware-i.

**UseHsts**- lisab the Strict-Transport-Security päise.

**UseHttpsRedirection**- suunab HTTP -päringud HTTPS -i.

**UseStaticFile**- tagastab staatilised failid ja lühised edasiseks taotluste töötlemiseks.

**UseCookiePolicy**- vastab rakendusele EL üldise andmekaitsemääruse (GDPR) määrustele.

**UseRouting**- marsuudi taotlused.

**UseAuthentication**- üritab kasutajat autentida enne, kui tal on lubatud juurdepääs turvalistele ressurssidele.

**UseAuthorization**- annab kasutajale juurdepääsu turvalistele ressurssidele.

**UseSession**- loob ja säilitab seansi oleku.

**UseEndpoints koos MapRazorPages**- Razor Pagesi lõpp -punktide lisamiseks päringuahelasse.

**UseSpaStaticFiles**- tavaliselt viimasel kohal, võimaldada kõigil teistel keskklassi esmalt vastata vastavatele päringutele. Lubada kliendipoolse marsruutimisega SPA-sid käitada kõigil marsruutidel, mida serverirakendus ei tunne.

**8. Valige välja viis Built-in Middleware ja iseloomustage neid.**

1. Health Check - Kontrollib rakenduse ASP.NET Core tervist ja selle sõltuvusi, näiteks kontrollib andmebaasi saadavust.
2. URL Rewrite - Toetab URL -ide ümberkirjutamist ja päringute ümbersuunamist. Enne komponente, mis on vajalikud WebSocket'i päringute vastuvõtmiseks.
3. Diagnostics - Mitu eraldi keskklassi, mis pakuvad arendajatele erandilehte, erandite käsitlemist, olekukoodi lehti ja uute rakenduste vaikeveebilehte.
4. Session - Pakub tuge kasutajaseansside haldamiseks enne komponente, mis nõuavad seanssi.
5. Error-handling - Vigade käsitlemisel vahevara korral on meil üks täiendav argument, seejärel rakenduse ja ruuteri taseme vahevara, mis on viga.