Single Sign-On (SSO)

Cilj

Omoguciti Login, Registraciju, Logout i ostali User management na jednom mjestu za sve ostale stranice nasih projekata. Sto znaci da ukoliko se prijavite (login) na SSO stranici bit ce te prijavljeni i na ostalim stranicama. Isto vrijedi i za ostale gore spomenute funkcionalnosti.

Ideja

Prilikom prijave na SSO stranicu od SSO servera dobijate cookie u kojem se nalazi token koji sluzi za potvrdu identiteta i koji (cookie) se salje u svakom requestu na server. Otvaranjem neke druge stranice (recimo vase) isti cookie se prenosi na nju i koristi se za autentikaciju. O tome kako se prenosi cookie ce pisati kasnije, jer postoji vise opcija.

Kada vasa stranica salje request na vas server salje se i cookie. Vas server vrijednost cookija, tj. token, prosljedjuje na servis SSO servera koji potvrdjuje identitet usera i vraca vasem serveru ID korisnika i role koje ima korisnik ili vas obavijesti da korisnik nije autentificiran.

Na SSO serveru postoje jos servisi za logout i za dobavljanje vise podatka o korisniku (Username, First name i Last name).

Registracija korisnika i dodjeljivanje rola korisnikicima ce se odvijati iskljucivo preko SSO stranice.

** Napomena: SSO stranica i SSO server su razlicite stvari. Prvo se odnosi na frontend, a drugo na backend. Servisi koje SSO server pruza ce biti WCF servisi.*

Dijeljenje cookija izmedju stranica

Varijanta 1: Ista (pod)domena

Sve stranice se nalaze pod istom domenom, razlikuju se samo po pocetnim putanjama. Primjer:

www.domena.com/sso
www.domena.com/tim1

www.domena.com/tim2

Kako su stranice pod istom domenom sve mogu bez problema pristupati cookiju.

Potrebno je samo konfigurirati server (apache, nginx, mozda i IIS (ali IIS ne znam kako)) tako da zavisno od pocetne putanje (sso, tim1 ili tim2) posluzuje razlicite aplikacije.

Prednosti: Relativno lagano za realizirati. Pogodno za development jer je moguce poddomenom localhost sve pokrenuti (jer je samo jedna domena u pitanju).

Varijanta 2: Razlicite poddomene

Svi projekti su poddomene iste domene. Primjer:

sso.domain.com
tim1.domain.com
tim2.domain.com

Cookije je potrebno postaviti za domenu '.domain.com'. Na ovaj nacin ce moci poddomene, nasi projekti, pristupati cookiju bez ikakvih problema. Realizacija je malo teza jer je potrebno napraviti domene i poddomene i vjerojatno je potrebno dodati vrijednosti u neki DNS server. U developmentu moguce je umjesto DNSa promijeniti hosts fileove da neke nase virtuelne (pod)domene resolvaju s localhost i opet omati neki server (apache, nginx) koji ce u zavisnosti od domene posluziti razlicitu aplikaciju.

Varijant 3: Cross domain cookie

Stranice su s razlicitih domena. Primjer:

www.sso.com
www.google.com
www.youtube.com

Ovo je najkompliciranija varijanta i rjesenje za nju jos uvijek u potpunosti ne znamo, ali je moguce. Takodjer je rizicno je da s losom implementacijom napravimo sigurnosne propuste i omogucimo da druge stranice preuzmu cookie, a time i vas identitet. Rjesenje bi najvjerojatnije ukljucivalo da svaka stranica projekta povlaci odredjene (JS) skripte sa SSO stranice preko kojih bi se omogucila razmjena tokena.

Update [2.11]:

- **Odlucena je Varijanta 2 s poddomenama od strane asistenta.**
- **Projekt ce se realizirati u koristeci WCF**

[5.11.]
Funkcionalnosti SSO sistema

SSO stranica funkcionalnosti:

- Login
- Registracija
- Logout
- Role managment
- Change password by Admin
- Change password by User

SSO externi servisi:

- Auth - ovo je onaj sto provjerava je li logovan user i vraca podatke o njemu
- Logout

SSO interni servisi:

- Login
- Registracija
- Logout
- Role management
- Change password by Admin
- Change password by User

Moguci dodatni zahtjevi:

- Izrada .NET middlewarea
- Izrada Java middlewarea

```
Claim {
        user_id: String?,
        token: String,
        created_at: Date,
        valid: Boolean
}
```

Login

Pretpostavka: korisnik nije logiran

Provjerava se da li su username i password tocni. O heshiranju passworda vidjeti vise u *Registracija*. Ukoliko kombinacija username-password nije tocna to se prijavljuje korisniku. Generira se novi token. Kreira se novi Claim objekt koji sadrzi korisnikov ID, hash generiranog tokena, valid=True i created_at postavljen na trenutno vrijeme. Korisniku se vaca request u kojem je server postavio cookie s ovim tokenom (ne sa hashom tokena, vec s tokenom). Za to se koristi Set-Cookie header sa HttpOnly flag-om što će zabraniti pristup ovom cookie-u iz skripti iz web browser-a.
Token je random hex i jedinstven je u bazi. Tako da nakon generiranja treba provjeriti postoji li vec u bazi token sa istim hash-om iako je mala sansa za to.

// Mozda bi trebalo i heshirati token?
Mogao bi se hashirani token smjestiti u bazu. Razlog ovome je isto kao za password, a to je da ako neko dodje u uvid baze da ne moze iskoristiti token da se prijavi. Ovo ce usporiti login i svaku provjeru da li je korisnik autentificiran.

Logout

Pretpostavka: korisnik je logiran, tj. posjeduje validan token

Pronalazi se u bazi objekt Claim s vrijednoscu hasha ovog tokena i postavlja se vrijednost valid=False.

Registracija

Preduvjet: Korisnik ima Admin rolu. (Novog korisnika moze dodati jedino admin)

Ulazni podaci: Username, Ime, Prezime, Password, Email.

Provjerava se da li u bazi postoji korisnik s istim username-om ili s email-om. Ukoliko postoji prijavljuje se greska.
Generira se slucajni salt. Password se hashira sa slucajnim saltom. U bazu se spremaju poslani podaci o korisniku. Takodjer se sprema i generirani salt i naravno umjesto passworda se sprema njegov hash.

// http://security.stackexchange.com/questions/8015/what-should-be-used-as-a-salt

Auth (externi)

Ovo je WCF servis. Salje se token koji je dobijen putem cookija. U bazi se provjerava da li postoji Claim s vrijednoscu ovog tokena (ili hashom tokena ukoliko hashiramo). Ukoliko je pronadjen provjerava se da li je valid=True. Ukoliko ne postoji claim ili valid nije true servis vraca da korisnik nije autentificiran.
U suprotnom se dohvataju podaci korisnika ciji je ID pod poljem user_id i njegove role i servis potvrdjuje da je korisnik autentificiran i vraca te podatke.

Logout (externi)

Takodjer WCF servis. Salje se token koji je dobijen putem cookija. U bazi se provjerava da li postoji Claim s vrijednoscu ovog tokena (ili hashom tokena ukoliko hashiramo). Ukoliko je pronadjen provjerava se da li je valid=True. Ukoliko ne postoji claim ili valid nije true servis vraca da korisnik nije autentificiran.
U suprotnom postavlja valid=False.

Role Management

- **Dodavanje/mijenjanje uloge korisniku**
    - Preduvjet: korisnik ima Admin rolu.
    - Request : POST
      {
        - UserId
        - RoleIds []

      }
    - Response 201 OK
      UserInfoVM
      {
        - Username
        - UserId
        - Roles []
        - FirstName
        - LastName

      }

Preporuke

- Login preko hash-a omoguciti
- Importovati entitete iz baze
- Ubaciti salt + hash passw
- Ako ima neki passw sa istim hash-om u bazi - pobrinuti se za to nekako (U hash da ulazi i username) ili da se Salt generira za svakog usera posebno i smjesta i on u bazu http://security.stackexchange.com/questions/8015/what-should-be-used-as-a-salt
-

Database update:

- **Dodati polje salt: String u user tabelu**
- **Dodati polje banned: Boolean u user tabelu**
- **Dodati Claim tabelu**

```
Claim {
        user_id: String? (isti kao kod User tabele),
        token: String,
        created_at: Date,
        valid: Boolean
}
```