

Raspodijeljeni sustavi 1. rok –2015/16.

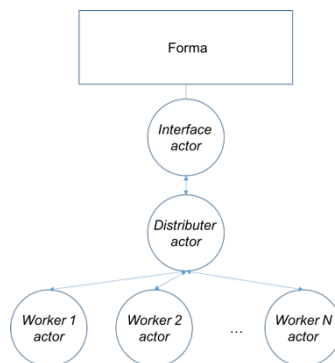
Napomena: U svim zadacima kada odgovarate sa servera koristite Context.Sender.Tell...

Prvi dio

Zadatak 1 . (2 boda)

Kreirajte Akka.NET aplikaciju koja će brojiti koliko puta se neko slovo ponavlja u datotekama mape na disku. U mapi na disku može biti proizvoljan broj datoteka, a sama datoteka može sadržavati proizvoljan broj znakova. Sadržaj datoteke nije ograničen na slova, ali se ostali znakovi ne smiju brojati.

Akka.NET sustav se mora sastojati od 3 vrste actora pri čemu *Interface actor* je taj koji komunicira s formom, *worker actor* je taj koji broji koliko puta se neko slovo ponavlja u **jednoj** datoteci, a *distributer actor* kreira **potreban broj worker-a**, prikuplja rezultate i šalje konačan rezultat *interface actor-u* koji ga ispiše na sučelje forme. Dok god je aplikacija pokrenuta, *interface* i *distributer actor* smiju biti samo jednom kreirani, dok *worker actor*-i moraju biti zaustavljeni nakon što obave svoj zadatak. Na slici 1 je vizualno prikazano kako su *actor*-i posloženi.



Slika 1 Actori u aplikaciji

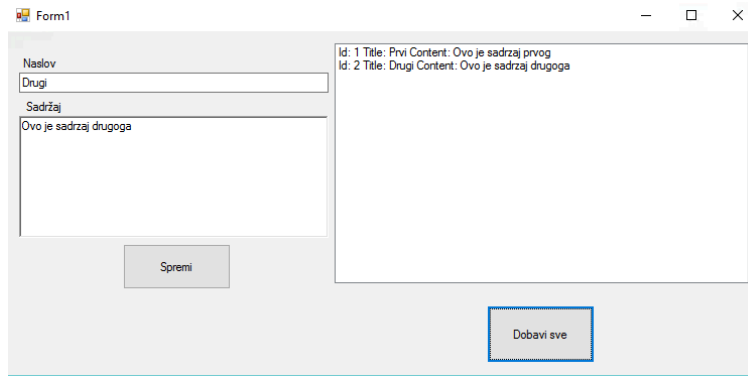
Kada pritisnete botun za pokretanje, Interface actoru se šalje poruka Start, nakon čega on prosljeđuje poruku Distributer actoru. Distributer actor šalje poruku ReadFile svim svojim Workerima. Workeri odgovaraju porukom FileData distributoru, koji nakon što primi od svih Workera odgovor šalje isti **tip** poruke interface actoru.

Kako bi vam se pomoglo, pripremljen je predložak za zadatak u kojem imate napravljene sve potrebne klase za *actor-e* te je napravljen direktorij s datotekama na kojem ćete raditi. Ujedno su vam dane implementirane metode za čitanje svih imena iz direktorija te metoda za čitanje sadržaja jedne datoteke.

Zadatak 2 (3 boda)

Napravite klijent – server Akka.NET (koristeći Akka.Remote) aplikaciju u kojoj server podržava dvije različite poruke: Save i GetAll. Forma klijenta izgleda kao na slici 2. Aplikacija koju radite jest spremanje članaka na server. Klijent unosi naslov članka i sadržaj te pritiskom na dugme "Spremi" šalje serveru poruku Save. Server zapisuje dobiveni sadržaj skupa s ID-om kojega sam dodijeli u datoteku. Cijeli sadržaj s ID-om je spremljen u jedan red, gdje su komponente Id, naslov i sadržaj odvojene posebnim znakom «#» (pretpostavljamo da se znak neće pojaviti u naslovu i sadržaju). Prvi članak koji se sprema mora imati ID 1. Pritiskom na dugme "Dobavi sve" klijent šalje poruku GetAll, a server klijentu sve članke spremljene u datoteku. Prije slanja članaka klijentu server mora iz datoteke raspoznati ID, naslov i sadržaj te kreirati objekt AllArticles. Klijent po primitku AllArticles ispisuje u listbox vrijednosti kao što je prikazano na slici 2.

Ime i prezime: _____



Slika 2 Izgled klijenta

Interface actor je taj koji ima pristup formi, a *communication actor* šalje i prima poruke od servera. Vodite računa o tome da za sadržaj i naslov korisnik ne može unijeti prazan tekst. Vodite računa o tome da oba samo jednom kreirate.

Kako bi vam se pomoglo pripremljeni su predlošci za klijent i server. Predlošci uključuju:

- Stvaranje actor sustava
- Konfiguracije za remote komunikaciju
- Čitanje i pisanje iz datoteke na strani servera (bez razdvajanja po posebnom znaku #)
- Poseban projekt s porukama za komunikaciju servera i klijenta (već uključen u solution)
- Prazne klase za potrebne actore

Drugi dio

Zadatak 3 (2 boda)

Napravite klijent – server Akka.NET aplikaciju u kojoj će server klijentu pružati mogućnost brojanja broja pojedinog slova u poslanom stringu. Klijent kao rezultat dobiva broj ponavljanja pojedinog slova. Napomena: Ne smijete brojati ništa osim slova.

Klijent šalje zahtjev serveru i očekuje rezultat. Zahtjev se šalje porukom Count i to je jedina poruka koju server prepoznaje. Klijent prepoznaje CountAck, Start i Retry. Start – pokreće sustav i prenosi sadržaj textboxa, Retry – posebna poruka za ponovno slanje, a CountAck – potvrda da je odrađen posao na serveru skupa s rezultatom.

Pouzdanost prijenosa poruka osigurajte *at-least-once* semantikom. Napravite da server u 50% slučajeva “zaboravi” poslati potvrdu o izvršenoj operaciji klijentu. Vodite računa o tome da svaku operaciju za koju niste dobili potvrdu o privitku morate poslati ponovno serveru. **Ne morate pamtit sve poruke koje ste poslali na koje čekate odgovor. Dovoljno je ID i ICancelable.**

Dobiveni rezultat se ispisuje u RichTextBox.

Kako bi vam se pomoglo pripremljeni su predlošci za klijent i server. Predlošci uključuju:

- Stvaranje actor sustava
- Konfiguracije za remote komunikaciju
- Poseban projekt s porukama za komunikaciju servera i klijenta (već uključen u solution)

Zadatak 4. (3 boda)

Za ovaj zadatak ste dobili gotov server koji prima poruku za spremanje sadržaja u datoteku i slanje odgovora klijentu. Na serveru ne morate ništa mijenjati.

Prilagodite implementaciju Lamportovog algoritma za međusobno isključivanje na način da kritični odsječak predstavljaju konzola za unos i dobiveni server. Kada dobije pristup kritičnom odsječku,

Ime i prezime: _____

klijent čeka unos s konzole nakon čega šalje poruku serveru. Klijent oslobađa kritični odjeljak tek kada primi odgovor od servera da je posao obavljen.

Poruka koja se šalje je Save i jednostavno prenosi uneseni string.

Klijenata može biti proizvoljan broj.

Modificirajte program i tako da svaki klijent čeka X milisekundi prije nego pošalje zahtjev za kritičnom sekcijom.

Objasnite Lamportov algoritam za međusobno isključivanje.

NAPOMENA: Programi koji u bilo kojem trenutku puknu ili se ne budu mogli uopće pokrenuti će biti ocijenjeni s 0 bodova.