COMPUTERWORLD

Tilstandsstyring uden session

Sessions er gode til at håndtere brugerspecifikke oplysninger på et website, men det har nogle svagheder som er mere eller mindre kendte. I denne artikel vil jeg prøve at kigge på nogle af disse svagheder og komme med nogle alternativer. [ASP]

Skrevet den **02. Feb 2009** af **softspot** I kategorien **Programmering / ASP** | ⭐⭐⭐☆☆

Ændringslog
01-02-2009: oprindelig artikel
01-02-2009: tilføjelse af kommentar til kommentar

Målgruppe
Webudviklere med nogen erfaring indenfor programmering og administration af websites og som har nogen kendskab til begreber i IIS 6.0, herunder worker process og recycling, samt webgardens og webfarms.

Introduktion
Artiklen tager udgangspunkt i ASP, men kan med stor sandsynlighed også porteres til andre teknologier/frameworks der skal arbejde sammen i et heterogent miljø (f.eks. et ASP-site, som er på vej over i ASP.NET - eller noget andet - men hvor dette ønskes gjort i faser).

Session-objektet
Gennem arbejdet med websites kommer man før eller siden i kontakt med behovet for at opbevare oplysninger mellem kald til serveren. Valget til at løse denne opgave falder ofte på Session-objektet og det er vældig fint til dette formål.

Nogle af fordelene ved session-objektet er:

```
* det er let at bruge og let tilgængeligt.

* det rydder sig selv op efter noget tid i dvale, man kan endda få besked om
dette
  (Session_OnEnd) - med de rigtige patches på serveren!

* det hjælper med at styre Locale (LCID), dvs. format af diverse oplysninger
  (valuta, dato, tal m.m.).

* det sørger selv for at blive klar når der er behov for det og kan ligeledes
  give besked om dette (Session_OnStart).
```

Et af problemerne med Session-objektet er (iflg. mange "guru'er"), at det er meget pladskrævende i hukommelsen og formegentlig også med hensyn til andre resurser. Dette kan specielt blive kritisk i forbindelse med DOS-angreb (Denial Of Service), hvor ens site bliver bombarderet med forespørgsler med

henblik på at serveren skal opgive at svare pga. overbelastning.

Jeg har ikke selv arbejdet med projekter der har oplevet problemer med hensyn til resurseforbruget fra Session-objekter, men jeg tror da på hvad guru'erne siger... :-)

Problemerne kan bla. være:

```
* urimeligt hukommelses- og resurseforbrug.

* besværliggørelse af skalering, da Session-objekter bliver bundet til en
  bestemt workerprocess, så end ikke webgardening er muligt og da slet ikke
webfarms!

* workerprocess recycling trækker benene væk under sessions (og application-
objektet
  for den sags skyld).

* blokering af samtidighed for sider der benytter sessions (da sessions er
  enkelttrådede pr. bruger). Det betyder at to asp-sider der benytter
  sessions, ikke kan køre samtidigt for den samme bruger!

* der er ikke mulighed for at dele sessions med andre frameworks, så som
ASP.NET,
  da disse bl.a. ikke deler session-format.

* der er sikkert andre mere eller mindre vigtige problemer jeg har glemt...
```

Hvis man kun fokuserer på problemerne, må målet være at man søger en begrænset brug af Session-objektet, eller helt undgår det. Det sidste har jeg set anbefalet mange steder, men jeg har endnu ikke set et seriøst bud på, hvad man så skal gøre i stedet. Det håber jeg, at kunne råde lidt bod på med denne artikel.

En kombinationsløsning

Hvis man vil omkring problemerne med skalering, kunne man implementere en del af den løsning jeg har i tankerne, nemlig at lade session (og application) objekterne være en Write-Through cache for værdier der gemmes i en bagvedliggende database. Filosofien er derefter, at forsøge læsning af session-værdier fra session først og dernæst fra databasen, hvis de ikke findes i session. Efter læsning i databasen, skal værdien lægges i session for at speede læsningen op næste gang. Rammer samme bruger en anden garden eller farm ved næste forespørgsel til sitet, vil værdien blive indlæst i den anden garden eller farm's adresserum og dermed også være tilgængelig der (hvilket så rejser nogle andre problemer, som jeg kommer lidt ind på senere).

Til dette formål ville jeg nok vælge at lave en klasse, som kunne stå for det administrative i forbindelse med opslag i session og evt. i databasen, hvis session ikke indeholder den ønskede/forventede værdi.

Kombinationsløsningen løser dog kun nogle af problemerne med session-objektet, men man bibeholder dog muligheden for at styre Locale (LCID), som bla. påvirker formatet af valuta, dato, tal osv.

De problemer der bliver løst er:

```
* skalerbarheden over gardens og farms

* workerprocess recycling

* åbner muligheden for at integrere med andre frameworks
```

Dog står man stadig tilbage med:

```
* det "urimelige" hukommelses-/resurseforbrug

* samtidighedsblokering
```

Plus at der nu er lagt et større pres på den database sitet benytter.

Den session-fri løsning
Med valget af den sessionfrie løsning, tager man selv det fulde ansvar for tilstandshåndtering, lige fra initiering, over opretholdelse til oprydning.

Hvis jeg lige skal ridse de enkelte faser op, så handler initiering om at undersøge om den aktuelle forespørgsel har en session og hvis ikke skal der

```
* oprettes en nøgle som kan sendes til klienten, så denne kan
  identificeres ved næste forespørgsel (svarer til Session.SessionID).
  Denne nøgle skal også registreres i databasen.

* eventuelt udføres noget brugerdefineret initieringskode (brugeren er
  her den programmør som benytter sig af modellen), som sørger for at bringe
  brugerens tilstand på plads i systemet (det der svarer til Session_OnStart).
```

Efterfølgende forespørgsler skal håndtere følgende:

```
* check om brugerens tilstand stadig er gyldig og evt. oprette ny
  hvis ikke (og rydde den gamle op!). Der skal laves et to-fase check,
  som dels undersøger hukommelsen for tilstand og et der undersøger
  databasen, hvis tilstanden ikke findes i hukommelsen.

* der skal evt. laves et check for "staleness" af cached data, dvs. give
  en mulighed for at angive, hvor længe der skal gå inden data SKAL
  læses fra databasen (Read-Through cache), for at sikre at data er nogenlunde
  up to date. Denne feature skal naturligvis benyttes med omtanke,
  da det kan have alvorlige konsekvenser på databasens tilgængelighed,
  hvis der læses igennem for ofte af for mange brugere på samme tid...

* der skal med jævne mellemrum kontrolleres for om en brugers tilstand er
```

```
forældet og i så fald, skal denne ryddes op. Dette kan give nogle
komplikationer i forhold til gardening og farming, da der jo kan være
mange kopier af samme brugertilstand indlæst i forskellige gardens og farms.
```

Konklusion
Det er ikke alle sites hvorpå erstatningen af session-objektet, efter denne model, vil være optimal. Umiddelbart vil jeg faktisk mene at sitet skal have en vis mængde trafik før jeg vil overveje denne. Dog er det besnærende for mig at man bliver fri for bekymringerne i forbindelse med de features, som IIS 6.0 implementerer. Features som er til for at skabe et mere stabilt miljø for de indlogerede websites (recycling og webgardening). Features som, med den sessionfrie løsning, burde kunne bruges uden videre...

Kommentar til kommentar
Først og fremmest tak fordi du gider tage dig tid til at læse og kommentere på min artikel. Jeg håber du vil holde dine kommentarer konstruktive og saglige, så jeg og andre kan få noget ud af dem. Hvis der kommer kommentarer som giver anledning til ændringer eller udvidelser af artiklen, vil jeg forsøge at flette dem ind i artiklen.

showsource >> sessions opbevares i hukommelsen og er derfor kun læst fra disk i det tilfælde at hukommelsen er virtuel...

Kommentar af showsource d. 01. Feb 2009 | 1

Jeg er overhovedet ikke til asp (windåze), men at server skal læse en fil fra disken, (sessions), især med mange (? 5+ ?) vars, så er det langt hurtigere at bruge en db, hvis ellers tabellen er rigtigt sat sammen.