KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Tomas Kvietkauskas

Smarty šablonų variklio spartinančiosios atmintinės funkcijos tobulinimas

Magistro darbas

Darbo vadovas
Doc. dr. R. Butkienė

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Tomas Kvietkauskas

Smarty šablonų variklio spartinančiosios atmintinės funkcijos tobulinimas

Magistro darbas

Recenzentas

Doc. dr. A. Ostreika

2008-01-14

Vadovas

Doc. dr. R. Butkienė
2008-01-14

Atliko

IFM-2/4 gr. stud.
Tomas Kvietkauskas

2008-01-14

Kaunas, 2008

Turinys

Summary

The Developement of Smarty Template Engine Caching Function

The main problem in heavy load web applications (built with PHP using template engines according by MVC methods) is waste of temporary information used in scripts execution. In order to enhance performance, temporary information must be cached and kept for the repeated usage.

We can use:

- proxy servers
- compiler‘s information caching
- database query caching
- software-based caching using template engines

Therefore it was analysed the last mentioned method – software-based caching and various template engines, requirements for template engines (MVC) and caching function.

The template engine which had showed the best results was Smarty. Accordingly the Smarty template engine was redeveloped by changing the main class and caching function.

Įvadas

Globaliai augant interneto vartotojų skaičiui atsiranda su tuo susijusios internetinių paslaugų teikimo problemos, kurios verčia ieškoti optimizacijos būdų, kad nenukentėtų paslaugos ir verslas. Didieji informaciniai portalai per parą sulaukia nuo šimtų tūkstančių iki kelių milijonų vartotojų, kitaip sakant nuo vienos iki kelių ar kelios dešimties užklausų per sekundę tenkančių web serveriui ir duomenų bazei.

Daugiausiai serverio resursų sunaudojama generuojant dinaminius puslapius (imant reikiamą informaciją iš duomenų bazės ar trečios šalies šaltinių), todėl norint sumažinti serverio apkrovas naudojami spartinimo būdai, išsaugantys laikinąją informaciją spartinančiojoje atmintinėje (angl. cache). Informacija išsaugojama (dubliuojama) spartinančiojoje atmintinėje iš kurios vėliau (jeigu reikalinga) paimama ir pateikiama vartotojui, neapkraunant duomenų šaltinio – pvz. duomenų bazės.

Galimi laikinosios informacijos išsaugojimo įgyvendinimo variantai:

- įgaliotasis (angl. proxy) serveris
- kompiliatoriaus duomenų išsaugojimas
- duomenų bazės užklausų išsaugojimas
- programiškai paremtas laikinosios informacijos išsaugojimas naudojant šablonų variklius.

Šiame darbe bus analizuojamas pastarasis variantas - programiškai paremtas laikinosios informacijos išsaugojimas naudojant šablonų variklius. Tai pigiausias ir paprasčiausias būdas optimizuoti resursus, ir sumažinti kodo vykdymo laiką. Šis būdas priešingai nei kiti priklauso ir nuo pačio šablonų variklio, informacijos išsaugojimo mechanizmo, ir nuo programuotojo ar sistemos projektuotojo.

Darbo tikslas - išanalizavus šablonų variklius, pagerinti geriausio iš pasirinktų šablonų variklių pagrindinę (vykdomąją) klasę, bei spartinančiosios atmintinės naudojimo mechanizmą.

Tyrimo problema. Problema yra ne techninė įranga (angl. hardware) – kompiuterių greitis ir galingumas auga, problema – prastai optimizuota programinė įranga (angl. software), bei netinkamas laikinosios informacijos panaudojimas. Norint pasiekti geresnių rezultatų reikia tinkamai optimizuoti sistemą, bei informaciją išsaugoti spartinančiojoje atmintinėje. Problemą laikysime išspręsta jeigu atlikus testavimą su

naujai parašyta spartinančiosios atmintinės naudojimo funkcija gausime geresnius rezultatus (rezultatai bus pateikti per trumpesnį laiko tarpą).

Tyrimo sritis – internetinių sistemų laikinosios informacijos saugojimo metodai.

Tyrimo objektas – tinklalapiai, realizuoti PHP kalba pagal MVC [4] modelį.

Tyrimo uždaviniai:

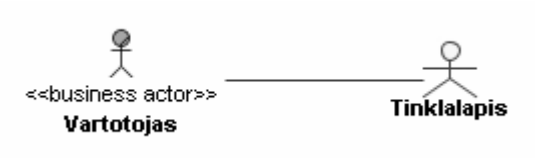
- Išanalizuoti laikinosios informacijos išsaugojimo būdus.
- Išsiaiškinti, kokie veiksniai įtakoja efektyvumą naudojant programiškai paremtą būdą.
- Atlikti techninę rizikos faktorių analizę
- Suformuluoti projekto reikalavimų specifikaciją
- Suprojektuoti naują informacijos išsaugojimo laikinojoje atmintinėje funkciją
- Parengti naujosios funkcijos eksperimentą

1. Analizė

Norint sukurti gerą internetinį projektą, kuris sulauks didelio dėmesio ir didelio lankytojų skaičiaus reikia gerai optimizuoti sistemą. Jeigu projektas sulauks šimtų tūkstančių lankytojų didelių serverio apkrovų išvengti nepavyks, tačiau sumažinti galima. Tai pasiekti galima ne tik atnaujinant serverio techninę įrangą, bet ir laikinai išsaugant reikalingą informaciją spartinančiojoje atmintinėje.

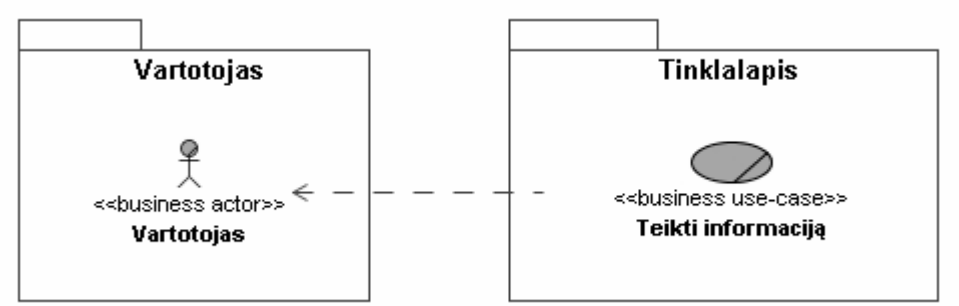
Gera informacijos sistema – tai tokia, kuri surenka, apdoroja, saugo ir platina informaciją padedančią priimti sprendimus, koordinuoja ir kontroliuoja organizacijos veiklą, analizuoja problemas, vykdo vartotojo užklausas ir grąžina rezultatus – informaciją, kurios pageidauja vartotojas. Geresnė bus ta sistema, kuri, vartotojui suformulavus užklausą, rezultatus grąžins per trumpesnį laiko tarpą.

1 paveiksle pavaizduota veiklos kontekstinė diagrama – veiklos sąveikų modelis. Veiklos subjektas – veiklos aktorius (vartotojas), dešinėje – bet koks tinklalapis realizuotas PHP kalba pagal MVC modelį.



1 pav. Veiklos kontekstinė diagrama

Detalesnis veiklos sąveikų modelis pateikiamas 2 paveiksle. Parodytas veiklos panaudojimo atvejis, kuriame dalyvauja išorinis aktorius (vartotojas).

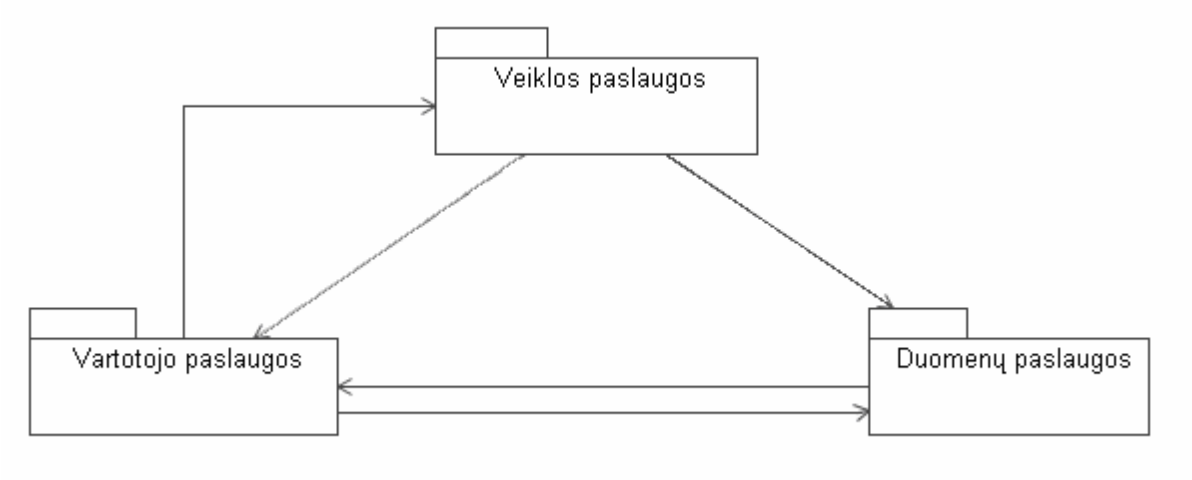


2 pav. Veiklos sąveikų modelis

1.1. MVC modelis

MVC (Model – View – Controller) objektiškai orientuotas programinės įrangos inžinerijos projektavimo modelis [4], naudojamas sudėtingose sistemose, programose,

kurios dirba su dideliu duomenų kiekiu. Tai vienas iš dažniausiai taikomų loginės architektūros variantų leidžiantis atskirti vartotojo interfeisą nuo programos veiklos ir duomenų paslaugų sluoksnio (pav 1):



1 pav. Trijų lygių loginės architektūros principinė schema

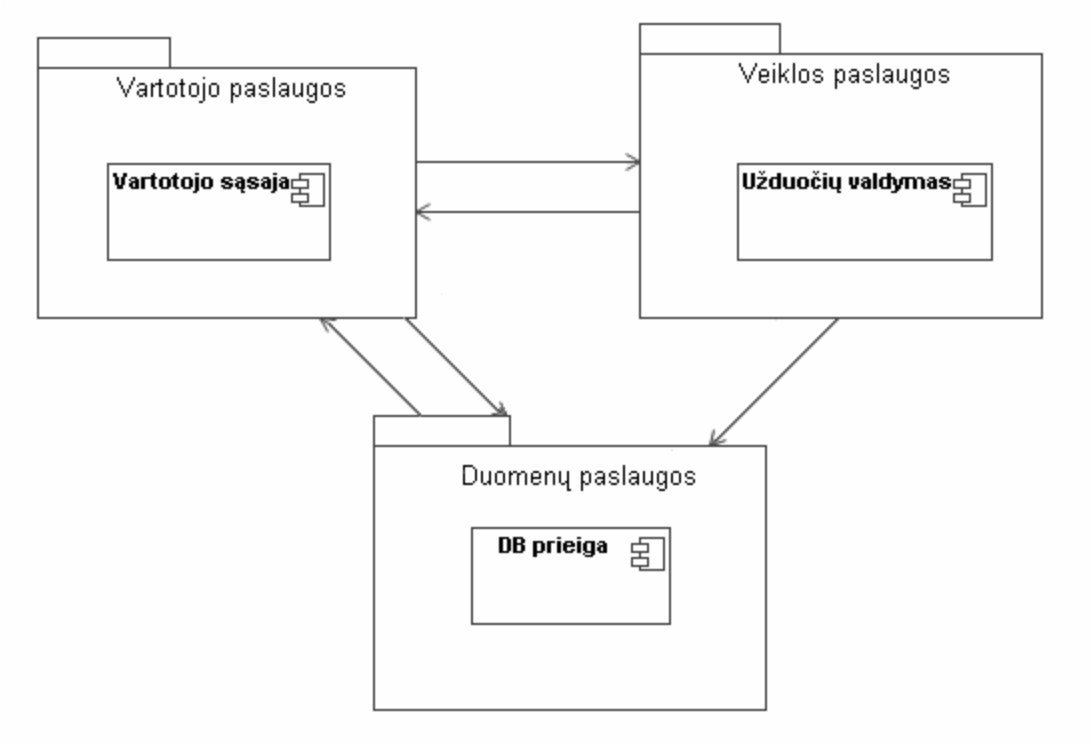
Naudojant PHP programavimo kalbą šį modelį atitinka šie komponentai [3]:

- duomenų paslaugos (duomenų modelis) – PHP kodas.
- veiklos paslaugos – PHP interpretatorius.
- vartotojo paslaugas teikia – vaizdavimo šablonai.

Nenaudojant šio programavimo būdo t.y. programuojant ne pagal MVC modelį duomenų modelis ir vaizdavimas (dizainas) yra sumaišyti - dažniausiai aprašyti viename sluoksnyje, kas apsunkina programavimą ir sumažina pakartotinio panaudojimo (angl. reuse) galimybes. Programuojant pagal MVC modelį duomenų modelis atskiriamas nuo vaizdavimo (dizaino) t.y. programos logika aprašoma viename sluoksnyje, vaizdavimo logika – kitame.

MVC yra trijų sluoksnių architektūra: pirmame – duomenų modelis, antrame – programos logika, trečiame – vaizdavimas.

2 paveiksle pateikiamas trijų lygių architektūros taikymo pavyzdys:

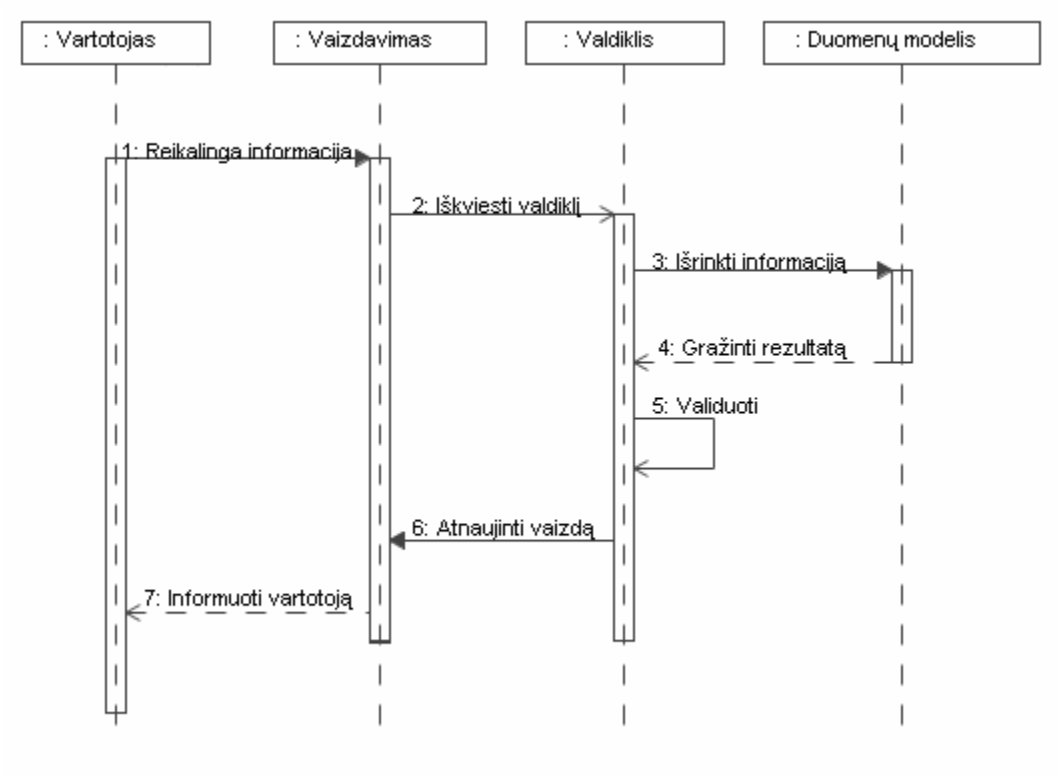


2 pav. MVC modelio taikymo pavyzdys

Toks būdas leidžia programuotojui ir šablonų projektuotojui (techniniam dizaineriui) lygiagrečiai dirbti prie vieno projekto.

3 paveikle pateikiama bendriausia kodo vykdymo sekų diagrama naudojant MVC modelį:

- vartotojas siunčia užklausą į serverį
- vaizdavimas kviečia valdiklį
- valdiklis užkrauna kodą
- duomenų modelis sugeneruoja kodą
- valdiklis užkrautą kodą įvykdo
- atnaujinamas vaizdavimas
- vartotojui grąžinama informacija

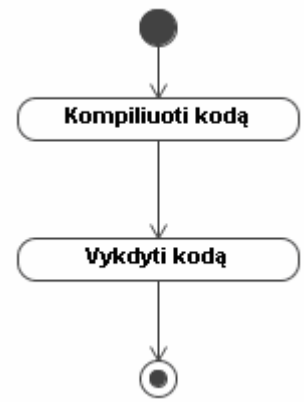


3 pav. MVC modelio sekų diagrama

1.2. Kodo vykdymas ir saugojimas spartinančiojoje atmintinėje

PHP kodo vykdymo mechanizmas veikia taip (4 pav.):

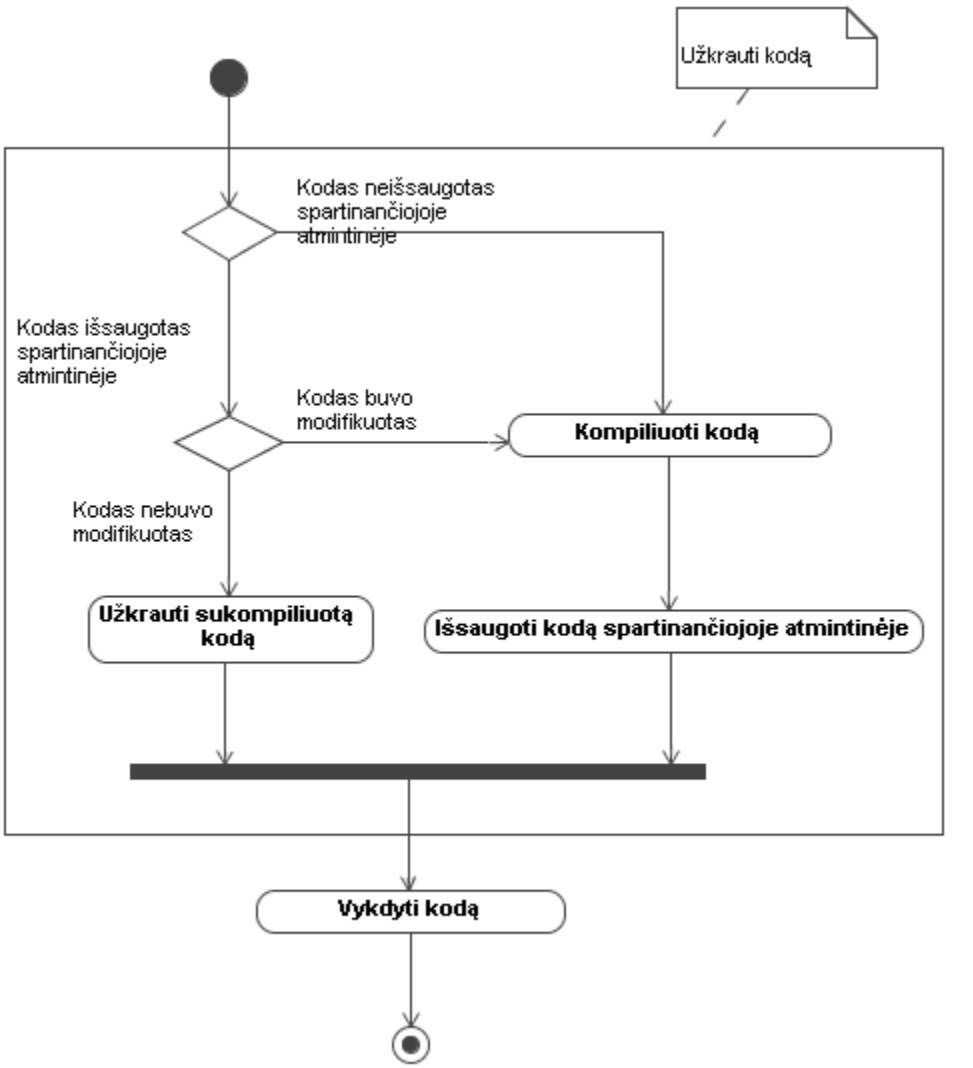
- skaitomas pirminis failas, nagrinėjamas gramatiškai (vykdomos duomenų bazės ar kitos užklausos) ir paverčiamas į interpretatoriui suprantamą kodą.
- PHP interpretatorius (kompiliatorius) vykdo kodą.



4 pav. kodo vykdymas be informacijos saugojimo

Šie du žingsniai vykdomi visą laiką gavus užklausą. Kompiliavimo rezultatai po kodo vykdymo yra prarandami.

Norint sumažinti vykdymo laiką, reikia spartinančiojoje atmintinėje saugoti interpretavimo (vykdymo) rezultatus, kurie vėliau bus panaudoti, kad vėl nereiktų iš naujo interpretuoti PHP kodo (5 pav):



5 pav. kodo vykdymas su informacijos išsaugojimu

PHP kodo vykdymo mechanizmas su informacijos išsaugojimu veikia taip:

- Jeigu sukompiliuotas kodas rastas spartinančiojoje atmintinėje tikrinama ar pirminis kodas nebuvo modifikuotas po paskutinio išsaugojimo. Jei ne - sukompiliuotas kodas užkraunamas iš spartinančiosios atmintinės, jeigu buvo modifikuotas – kompiliuojamas ir įrašomas į spartinančiąją atmintinę.
- Jeigu kodas nebuvo išsaugotas spartinančiojoje atmintinėje jis sukompiliuojamas ir įrašomas į spartinančiąją atmintinę.
- PHP kompiliatorius vykdo kodą.

Panaudos atvejo „Užkrauti kodą“ specifikacija detaliai aprašyta 1-oje lentelėje.

1 Lentelė

Panaudojimo atvejo „Užkrauti kodą“ specifikacija.

PA „**Užkrauti kodą**“	
Prieš sąlyga	Turi būti išpilyti MVC modelio reikalavimai
Sužadinimo sąlyga	Vartotojas siunčia užklausą į serverį. Vaizdavimas iškviečia valdiklį. Valdiklis užkrauna kodą.
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Užkraunamas kodas	
2. Vykdomas kodas	
Po sąlyga:	Valdiklis atnaujina vaizdavimą. Vaizdavimas informaciją perduoda vartotojui.
Alternatyvūs scenarijai	
1. Jei kodas nebuvo išsaugotas spartinančiojoje atmintinėje arba kodas išsaugotas, bet buvo modifikuotas	**1.1**. Kompiliuojamas kodas **1.2** Sukompiliuotas kodas užsaugomas spartinančiojoje atmintinėje

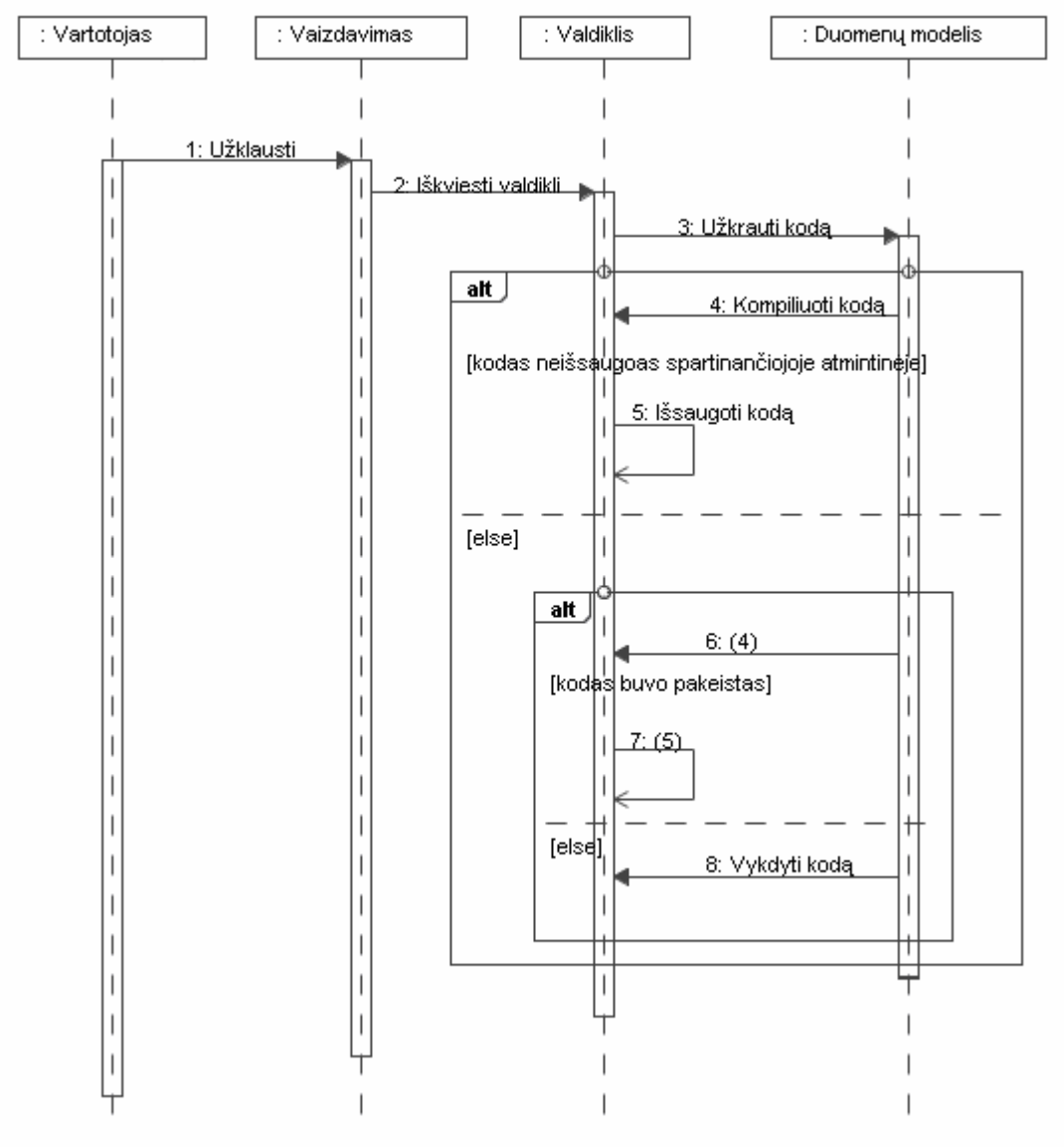
Kompiliavimo rezultatai po kodo vykdymo nėra prarandami, jie išsaugomi spartinančiojoje atmintinėje iš kurios vėliau yra užkraunami ir perduodami dar kartą kompiliatoriui.

Taip iš naujo vykdant PHP kodą panaudojami kompiliavimo rezultatai leidžia taupyti kompiliavimo laiką ir resursus.

6 paveiksle pateikta kodo su informacijos išsaugojimu, realizuoto pagal MVC modelį, vykdymo sekų diagrama:

- vartotojas siunčia užklausą į serverį
- vaizdavimas kviečia valdiklį
- valdiklis užkrauna kodą
- duomenų modelis generuoja kodą
  - jei kodas rastas spartinančioje atmintinėje jis užkraunamas iš spartinančiosios atmintinės

- valdiklis užkrautą kodą vykdo
  - jei kodas nebuvo rastas spartinančioje atmintinėje po vykdymo kodo kopija išsaugojama spartinančiojoje atmintinėje
- atnaujinamas vaizdavimas
- vartotojui grąžinama informacija



6 pav. Kodo vykdymo sekų diagrama

1.3. Architektūros ir galimų įgyvendinimo priemonių variantų analizė

- **Viso sugeneruoto puslapio saugojimas spartinančiojoje atmintinėje**

  Dažniausiai tinklalapiai neturi daug dinaminės informacijos savo puslapiuose. Nekreipiant dėmesio į tai, kad beveik visur yra įdiegtos turinio valdymo sistemos, turinys yra atnaujinamas žymiai rečiau nei jį peržiūri vartotojai. Toks sugeneruotas puslapis saugomas serverio kietajame diske arba serverio atmintyje, arba duomenų bazėje. Kiekvienas iš šių būdų turi savų pliusų ir minusų, todėl juos reikia rinktis pagal galimybes. Toks spartinančiosios atmintinės naudojimo būdas gali leisti išnaudoti *Last-Modified/If-Modified-Since* ir *Etag/If-None-Match* HTTP antraštes. Jis tinka jei tinklalapyje nėra prisijungimo formų, registracijų, balsavimų ar kitų interaktyvių dalykų, arba reikia atsižvelgti į tai kuriant tokią spartinančiosios atmintinės naudojimo sistemą.

- **Puslapio dalių saugojimas spartinančiojoje atmintinėje**

  Kadangi vis daugiau puslapių naudoja registracijas, balsavimus ar kitokius interaktyvius dalykus (informacijos blokus, kurie naudojami daugelyje puslapių, palyginti retai atnaujinami, bet jų generavimui reikia daug resursų), o puslapių generavimas pradeda reikalauti dėl to daugiau resursų, tokiu atveju spartinančiojoje atmintinėje išsaugojamas ne visas puslapis, o tik jo dalis.

  Išsaugojama būtent ta dalis, kuri yra palyginti rečiausiai atnaujinama ar reikalauja daugiausiai resursų generavimui.

  Galima laikinai išsaugoti ne sugeneruotas puslapio dalis, o duomenis (masyvus, objektus) skirtus šiems puslapiams formuoti. Informaciją saugoti vėl įprastoje mums vietoje: atmintinėje, atmintyje, duomenų bazėje.

- **Trečios šalies duomenų išsaugojimas**

  Dažnai tinklalapiuose tenka rodyti informaciją iš trečių šaltinių. Informacija iš įvairių šaltinių (dažniausiai XML formate) gaunama iš trečios šalies, perdirbama į mums reikalingą pavidalą ir parodoma mūsų puslapyje. Šią informaciją reikia išsaugoti dėl greičio ir informacijos stabilumo, bei laiko taupymo.

1.4. Informacijos saugojimo spartinančiojoje atmintinėje būdai

Yra keli pagrindiniai informacijos saugojimo spartinančiojoje atmintinėje būdai:

- **Squid – įgaliotasis (angl. proxy) serveris** [3]

  Squid gali pagreitinti statinių puslapių užkrovimą, naudodamas nuotolinio serverio funkcionalumą, tačiau tai neišeitis dinaminiams puslapiams. Tai gali net pabloginti situaciją. Tai būdas, labai panašus į „Viso puslapio saugojimą spartinančiojoje atmintinėje“. Šiuo atveju nieko nereikia keisti sukurtoje sistemoje. Tam naudojama „reverse proxy“ [18] programinė įranga.

  Įgaliotasis serveris fiziškai yra prieš HTTP serverį ir visos klientų užklausos keliauja per įgaliotąjį serverį. Šis nusprendžia ar siųsti šią užklausą Web serveriui, ar jis pats gali apdoroti (pagal užklausą jis turi turinį, kurį gali pateikti vartotojui). Dažniausiai tokia programinė įranga išsaugo viską: paveiksliukus, JavaScript scenarijus, CSS stilių failus, HTML puslapius, beveik viską, kas keliauja HTTP protokolu nuo serverio iki kliento. Išsaugojimo trukmė ar atnaujinimas priklauso nuo programinės įrangos, jos konfigūracijos ir sistemos.

- **APC (Advanced PHP Cache)** [3]

  APC – pažangus PHP laikinosios atmintinės naudojimas. Tai papildomas PHP modulis, kuris saugo kompiliatoriaus rezultatus ir panaudoja dar kartą, žinoma, prieš tai patikrinus ar kompiliuojamas kodas nebuvo modifikuotas. APC pagreitina kodo vykdymą, tačiau apie jį labai mažai informacijos, nes modulis parašytas dar 2000 metais, beveik neatnaujinamas ir visiškai nebenaudojamas.

- **SQL užklausų išsaugojimas** [3]

  Užklausų išsaugojimos naudojamas ten kur labai daug duomenų bazės užklausų.

  MySQL turi Query Cache [8]. Duomenų išsaugojimą duomenų bazės lygyje valdo pati duomenų bazė, o ne programuotojas ar sistema. Užklausų, užklausų duomenų ar duomenų bazės metaduomenų išsaugojimas atmintyje yra daug greitesnis būdas, nei iš naujo tos pačios informacijos išrinkimas.

- **Turck MMCache** [17]

  Tai dar vienas PHP atviro kodo dinaminio turinio akseleratorius, kuris diegiamas kaip PHP kompiliatoriaus modulis. Daugiau nebeatnaujinamas kūrėjų ir nelabai naudojamas. Paskutinės žinios 2005 metų.

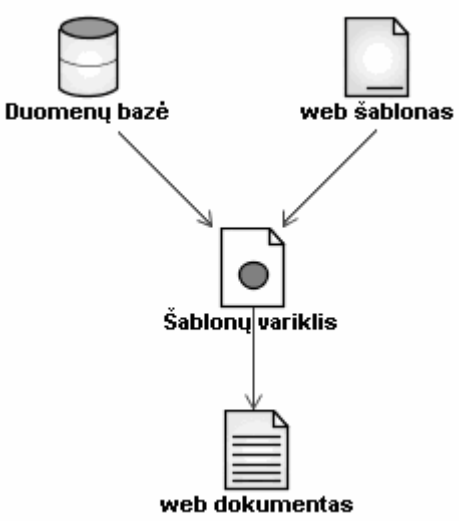
- **MVC modeliu paremtas informacijos saugojimas spartinančiojoje atmintinėje.**

  PHP šablonų varikliai leidžia atskirti PHP kodo logiką nuo vaizdavimo elementų. Korektiškai valdant bei manipuliuojant spartinančiosios atmintinės galimybėmis galimas žymus bet kokios informacinės sistemos, realizuotos šablonų varikliu pagal MVC modelį, pagreitėjimas.

  Šiame darbe bus analizuojamas ir tobulinamas pastarasis variantas – informacijos saugojimas spartinančiojoje atmintinėje naudojant šablonų variklius, nes tai yra pats paprasčiausias ir pigiausias variantas leidžiantis taupyti sistemos resursus ir kodo vykdymo laiką.

1.5. Šablonų ir jų variklių analizė

Šablonų variklis (7 pav.) – tai programinė įranga, kuri atskiria programos logiką nuo vaizdavimo logikos ir dizaino elementų. Kodas gali būti keičiamas nepriklausomai nuo tinklalapio dizaino. Šablonai gali būti vaizduojami daugelyje vietų tinklalapyje. Vienas pakeitimas šablone įtakos visas tinklalapio vietas, kuriose naudojamas tas pats šablonas. Tai leidžia taupyti laiką, nes vietoje pakeitimų keliuose puslapiuose užtenka tai atlikti vienoje vietoje.



7 pav. Šablonų variklio principinė schema

- Programinė logika

  Šiame lygyje vykdomas duomenų apdorojimas, pvz. paėmimas iš duomenų bazės ir informacijos priskyrimas šablonų kintamiesiems.

  Lengva pasakyti, kuri programos dalis ką atlieka. Kodas yra švaresnis, paprastesnis, lengviau skaitomas, galima lengviau pastebėti klaidas. Tai ypač aktualu kai prie vieno kodo dirba keli programuotojai.

  Programos logika gali būti atnaujinta, pakartotinai panaudota (angl. reused) ar net visiškai pakeista nauja be jokios įtakos vaizdavimo logikai.

- Vaizdavimo logika

  Šiame lygmenyje vaizdavimo logika kontroliuoja šablonų failus ir vaizdavimo duomenis. Šablonų duomenys ir išoriniai duomenys yra sujungiami į vieną duomenų sluoksnį.

  Dauguma šablonų variklių naudoja savo šablonų logiką. Pavyzdžiui Smarty šablonų variklio logika aprašoma savo kalba. Kituose varikliuose logika aprašoma PHP kalba.

Toliau trumpai aprašomas šablonų variklių funkcionalumas.

- **Heyes Template Class** [6]

  Viena PHP klasė turinti tik tris pagrindinius šablonų variklio metodus. Turi du operatorius: IF – sąlygos, LOOP – ciklo. Neturi funkcijos, kuri leistų informaciją saugoti spartinančiojoje atmintinėje. Nenaudoja reguliarių išraiškų.

  Versija: 1.4
  Paskutinis atnaujinimas: 2001-07-14
  Autorius: Richard Heyes.

- **Fast Template** [7]

  Kažkada buvęs populiarus CGI šablonų variklis perrašytas PHP kalba. Neturi funkcijos leidžiančios saugoti informaciją spartinančiojoje atmintinėje. Šablonų kintamiesiems naudoja reguliarias išraiškas - {(A-Z0-9_)+}.
  Versija: 1.1.0

Paskutinis atnaujinimas: 1999-05-27

Autorius: originalus Perl CGI::FastTemplate modulis - Jason Moore

- **bTemplate** [9]

  Paprasta PHP klasė, turinti *private* ir *public* metodus, lengvai konfiguruojama klases viduje. Tačiau neturi funkcijos, kuri saugotų tarpinę informaciją spartinančiojoje atmintinėje.

  Versija: 0.3

  Paskutinis atnaujinimas: 2002-11-27

  Autorius: Brian Lozier

- **Savant** [10]

  Nedidelė, tačiau turinti daug failų sistema. Kieviena klasė, modulis, įskiepis atskirame faile. Aiški ir patogi architektūra. Funkcionalumas plečiamas per įskiepius. Tačiau trūkumas - neturi funkcijos, kuri paspartintų veikimą saugodama informaciją spartinančiojoje atmintinėje. Suderinama tik su PHP 5 versija.

  Versija: 2.4.3

  Paskutinis atnaujinimas: 2006-03-05

  Autorius: Paul M. Jones

- **Avan Template** [12]

  Viena klasė, naudojanti PEAR biblioteką. Be kintamųjų pakeitimo palaiko blokus. Neturi funkcijos, kuri saugotų tarpinę informaciją spartinančiojoje atmintinėje.

  Versija: 1.2.2

  Paskutinis atnaujinimas: 2001 metai

  Autorius: Naoki Shima.

- **Template Power** [14]

  Palaiko dinaminius blokus, blokų/failų įtraukimą, nepriskirtų kintamųjų blokų rodymą/slėpimą. Neturi funkcijos leidžiančios saugoti informaciją spartinančiojoje atmintinėje.

  Versija: 3.0.2

  Paskutinis atnaujinimas: 2002 metai

Autorius: R.P.J. Velzeboer

- **phemplate** [16]

  Lietuvškas šablonų variklis palaikantis ciklus, blokus. Neturi funkcijos leidžiančios saugoti informaciją spartinančiojoje atmintinėje.
  Versija: 1.10.1
  Paskutinis atnaujinimas: 2004-04-21
  Autorius: Juozas Šalna

- **ETS - easy template system** [11]

  Dirba su dviem elementais: duomenų medžiu ir šablonais. Integruoti trasavimo (debug) klaidų pranešimai. Gali konstruoti ne tik HTML šablonų failus bet ir SQL užklausas, XML dokumentus. Turi sistemą leidžiančią saugoti informaciją spartinančiojoje atmintinėje.
  Versija: 3.06a
  Paskutinis atnaujinimas: 2004 metai
  Autorius: Franck Marcia

- **Grafx Software's Fast Template** [13]

  Fast Template sistemos modifikacija turinti funkciją, leidžiančią saugoti informaciją spartinančiojoje atmintinėje, trasavimo konsolę ir nepriskirtų dinaminių blokų išmetimą.
  Versija: 1.5.1
  Paskutinis atnaujinimas: 2006-05-01
  Autorius: Grafx Software

- **Cached Fast Template** [15]

  Dar viena Fast Template modifikacija turinti funckciją galinčią saugoti šablonų failus spartinančiojoje atmintinėje. Leidžia saugoti atskiras failo dalis, blokus.
  Versija: 1.0.1
  Paskutinis atnaujinimas: 2000-02-01
  Autorius: Benjamin Kahn

- **Smarty** [1]

  Greitas, lankstus variklis, kompiliuoja šablonus į PHP failus, tada tik juos vykdo. Sukompiuliuotą informaciją gali saugoti spartinančiojoje atmintinėje. Turi aiškią architektūrą, daug įskpiepių. Didžiausia ir galingiausia sistema iš pastarųjų peržiūrėtų. Smarty bando išrasti naują programavimo kalbą ne tik vaizdavimui, bet ir vaizdavimo logikai. Visose kitose sistemose vaizdavimo logika aprašoma PHP failuose.

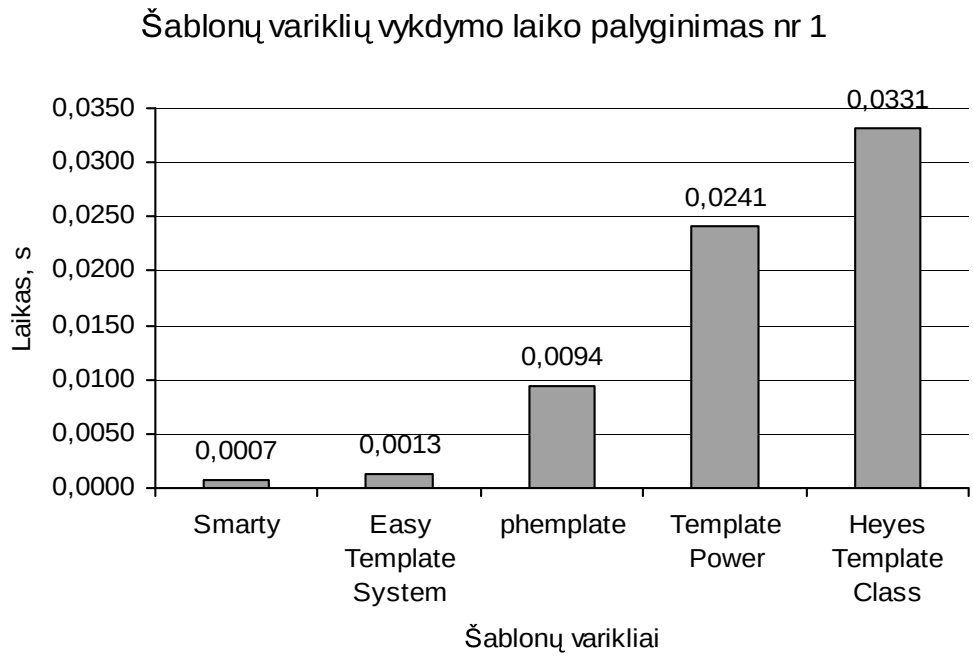
  Versija: 2.6.18

  Paskutinis atnaujinimas: 2007-03-07

  Autorius: The PHP Group

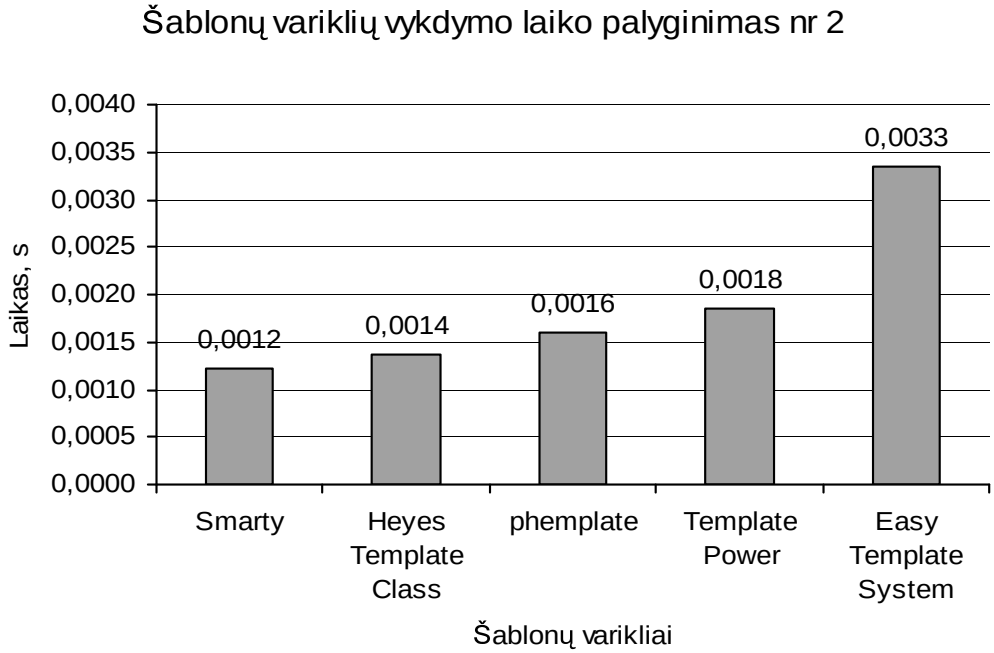
Tai tikrai nevisi šablonų varikliai [5]. Dauguma pasižymi pagrindine savybe – atskirti PHP logiką nuo dizaino elementų (MVC modelis [4]): programos logika (kodas) rašoma viename faile, dizainas (vaizdavimo logika) aprašomas kitame – šablonų faile.

8 paveiksle pateiktas analizuojamų šablonų variklių vykdymo palyginimas. Šiame teste buvo pateiktas ~70kb šablono failas be kintamųjų (ištrauka pateikta 1-ame priede), 100 ciklo įrašų, 10 iteracijų, atitinkančių paprasčiausią kodą, paimantį įrašus iš duomenų bazės ir surašantį duomenis į lentelę.



8 pav. Šablonų variklių vykdymo laiko palyginimas nr 1

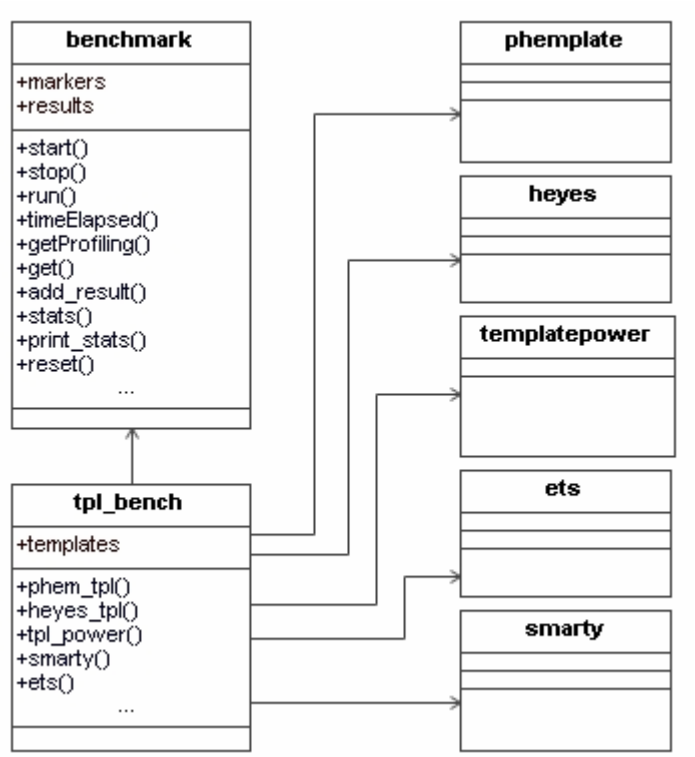
9 paveiksle pateikti antrojo testo rezultatai. Šiame teste buvo skaitomas ~300kb dydžio XML failas (ištrauka pateikta 2-ame priede), turintis apie 3000 elementų (daugiau nei 14000 eilučių) iš kurių išrenkamos 7 reikšmės, surašomos į masyvą ir išvedamos pasinaudojus šablonų failais (pateikta 3-ame priede).



9 pav. Šablonų variklių vykdymo palyginimas nr 2

Testai buvo atlikti naudojant PEAR bandymų (angl. benchmark) klasę *Benchmark_Iterate,* kuri įvykdo duotą kodą ir grąžina vykdymo statistiką. 4 – ame priede pateiktas testavimo išeities kodas.

10 paveiksle pateikta testavimo klasių diagrama:



10 pav. testavimo klasių diagrama

Pagal šiuos rezultatus nustatyta, kad iš analizuojamų šablonų variklių yra geriausias Smarty:

- pasižymi didžiausiu funkcionalumu
- stipriausias palaikymas iš kūrėjų pusės (forumai, wiki, pašto konferencijos, atnaujinimai)
- geriausi testų vykdymo rezultatai

1.6. Tolesnio darbo tikslas

Šio darbo tikslas yra pagerinti t.y. perrašyti (parašyti naują) Smarty šablonų variklio pagrindinę (vykdomąją) klasę, bei spartinančiosios atmintinės naudojimo mechanizmą, ko pasekoje sumažės PHP kodo kiekis, sumažės serverio apkrovos vidurkiai, pagreitės šablonų failų vykdymas ir bet kokios informacinės sistemos veikimas realizuotos Smarty šablonų varikliu. Tai labai aktualu kai vienu metu aptarnaujama daug lankytojų, daug duomenų bazės užklausų. Per tą patį laiką bus galima aptarnauti daugiau vartotojų nei anksčiau arba tas pats, ankstesnis kiekis vartotojų bus aptarnautas greičiau.

1.7. Rizikos faktorių analizė

Didelę įtaką failų vykdymui ir greičiui turi serverio techninė įranga. Daug priklauso nuo to kokiu apkrovimu veikia serveris - procesorius, atmintis, kietieji diskai, kiek užimta spartinančiosios laikmenos, t.y. kokie yra serverio apkrovos vidurkiai, kiek yra optimizuotas pats PHP kodas, kokie įdiegti kompiliatoriaus moduliai naudojantys spartinančiąją atmintinę.

Iš techninės pusės dažniausiai naudojama paskirstyto apkrovimo (ang. load balancing) [19] technologija, paskirstanti apkrovimą keliems kompiuteriams, procesams, kietiems diskams ar kitiems resursams, leidžianti gauti optimalų resursų panaudojimą mažinant darbo laiką.

1.8. Vartotojų analizė

Pagrindinis šablonų variklio tikslas yra atskirti programinę logiką nuo vaizdavimo logikos. Programos logika – PHP kodas yra tvarkomas programuotojų. Vaizdavimo logika, aprašyta vaizdavimo šablonuose, apibrėžia kokį vaizdą matys galutinis vartotojas. Vaizdavimo šablonus tvarko šablonų dizaineriai (techniniai dizaineriai).

Šis sprendimas didžiausią įtaką turės programuotojams ir vartotojams, kurie naudos sistemą realizuotą Smarty šablinų varikliu.

1.9. Analizės išvados

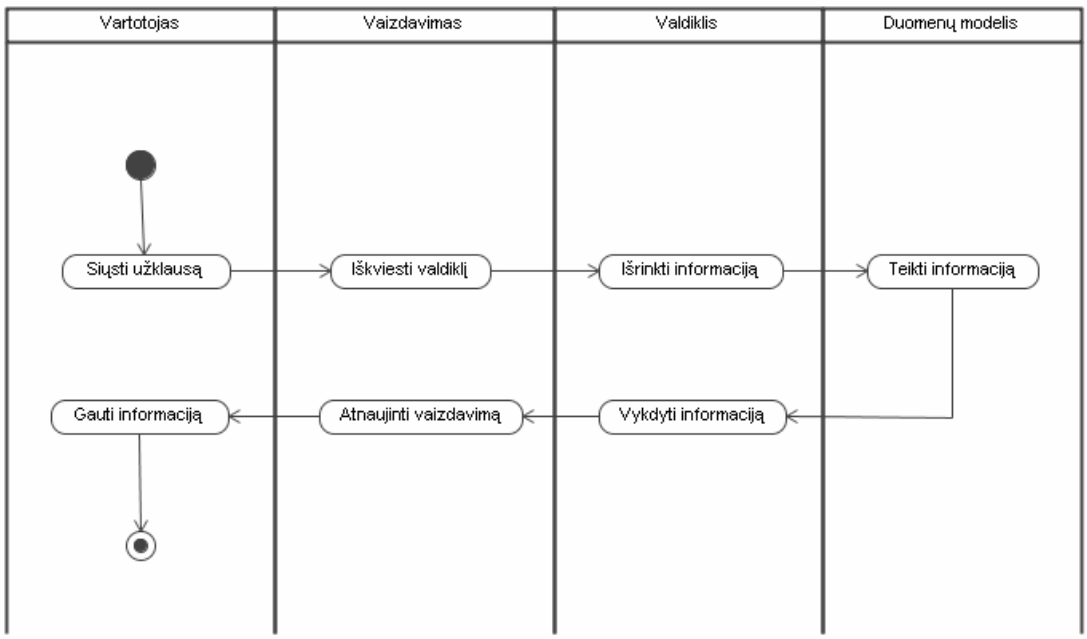
1. Apžvelgus laikinosios informacijos išsaugojimo įgyvendinimo variantus, pastebėta, kad paprasčiausias ir pigiausias yra programiškai paremtas būdas (naudojant šablonų variklius) organizuoti laikinosios informacijos išsaugojimą.
2. Atlikus lyginamąją šablonų variklių analizę, nuspręsta, kad geriausias yra Smarty šablonų variklis, nes testavime parodė geriausius rezultatus, bei turi PHP kūrėjų palaikymą.

2. Reikalavimų specifikacija ir analizė

Išanalizavus galimus įgyvendinimo priemonių variantus, MVC modelį, panašius šablonų variklius nuspręsta tolesniame darbe bandyti išsamiai ištirti ir parodyti Smarty šablonų variklio tobulinimo galimybes, nes pagal atlikto palyginimo testų rezultatus Smarty šablonų variklis parodė geriausius rezultatus, bei yra palaikomas PHP kūrėjų.

2.1. Reikalavimų specifikacija

Reikalavimų etape išskiriami panaudojimo atvejai ir apibrėžiamos sistemos ribos. Panaudojimo atvejų diagramos nerodo veiklos procesų eigos. Veiklos procesus vaizduoja veiklos diagramos. Procesas aukščiausiame abstrakcijos lygmenyje atrodytų taip (pav 11):

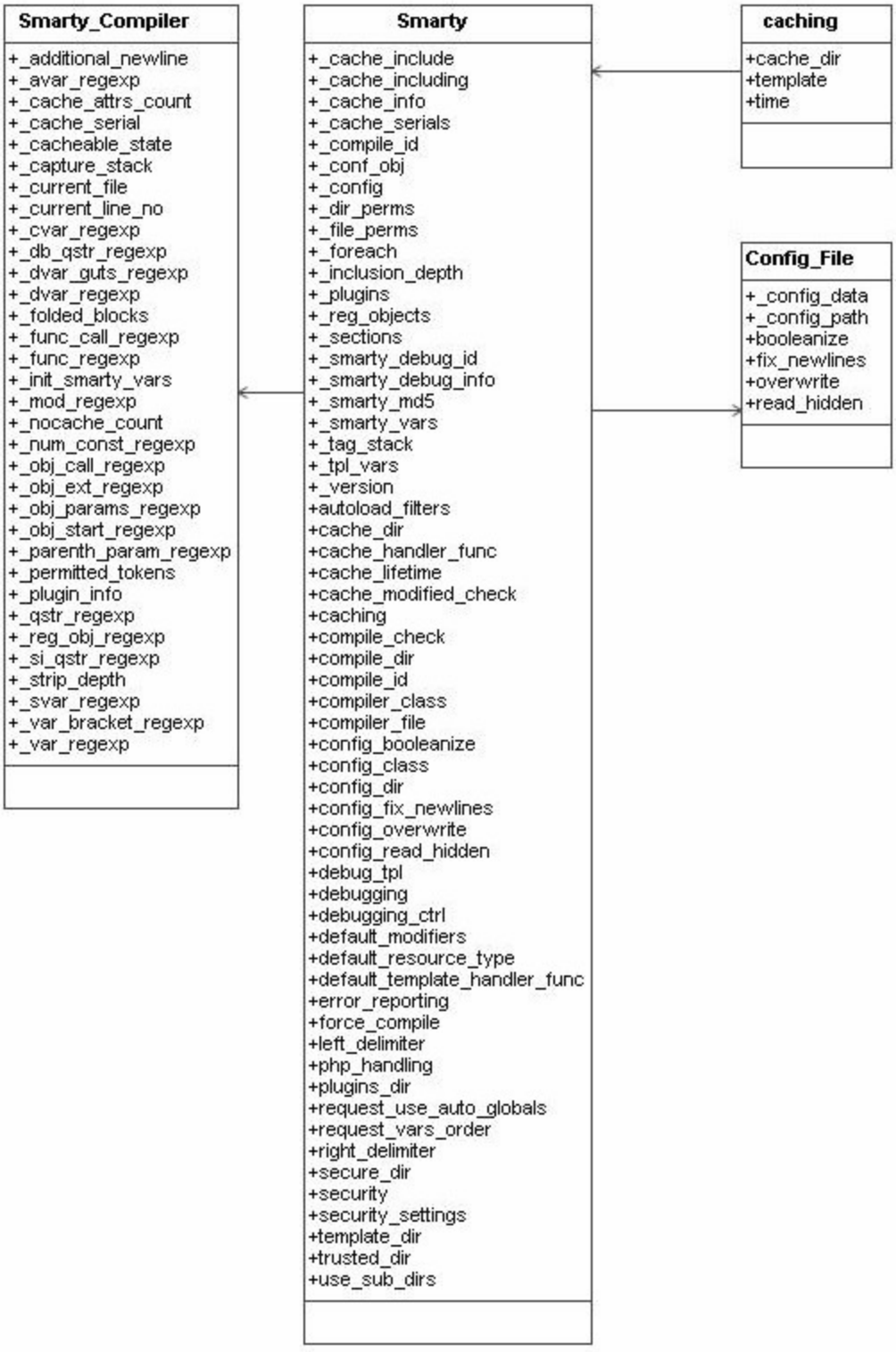


11 pav. proceso modelis

Pagal 11 paveiksle pateiktą proceso modelį turi būti realizuota nauja klasė su metodais: informacijos išsaugojimo spartinančiojoje atmintinėje ir informacijos pateikimo iš spartinančiosios atmintinės.

2.2. Dalykinės srities modelis

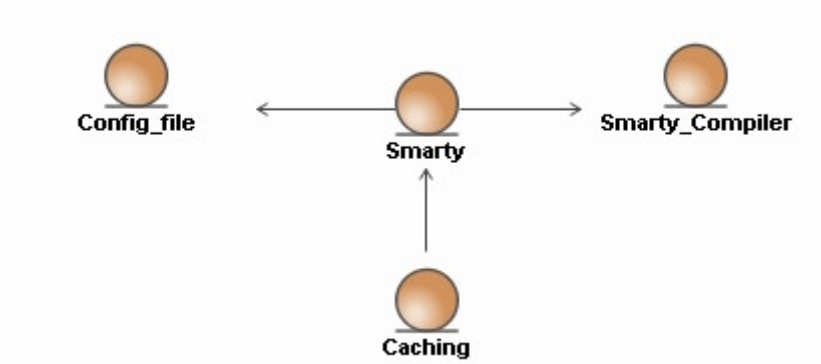
12 paveiksle pateikta projekto klasių diagrama kartu su klasių kintamaisiais.



12 pav. klasių diagrama

2.3. Reikalavimų analizė

Analizės metu panaudojimo atvejui sudaryta analizės diagrama rodo, kokios klasės realizuoja panaudojimo atvejį. Pagal RUP, analizės metu naudojami ribinių (*boundary*), esybių (*entity*) ir valdymo (*control*) klasių stereotipai. Analizės klasių diagrama pateikta 13 paveiksle:



13 pav. Reikalavimų analizės klasių diagrama

2.4. Nefunkciniai reikalavimai

Naujas sukurtas funkcionalumas turi būti langvai integruojamas į šablonų sistemą, lengvai tobulinamas ir atnaujinamas. Turi būti efektyviai naudojami esami resursai, netrikdomas ar kitaip įtakojamas serverio darbas. Išsaugojama informacija negali būti iškraipoma, išsaugojama informacija turi atitikti perduotą informaciją. Informacija turi būti apdorojama greitai ir stabiliai. Naudojant naująjį funkcionalumą kodas turi būti įvykdomas greičiau t.y. informacija vartotojui turi būti grąžinta per trumpesnį laiko tarpą.

3. Projektas

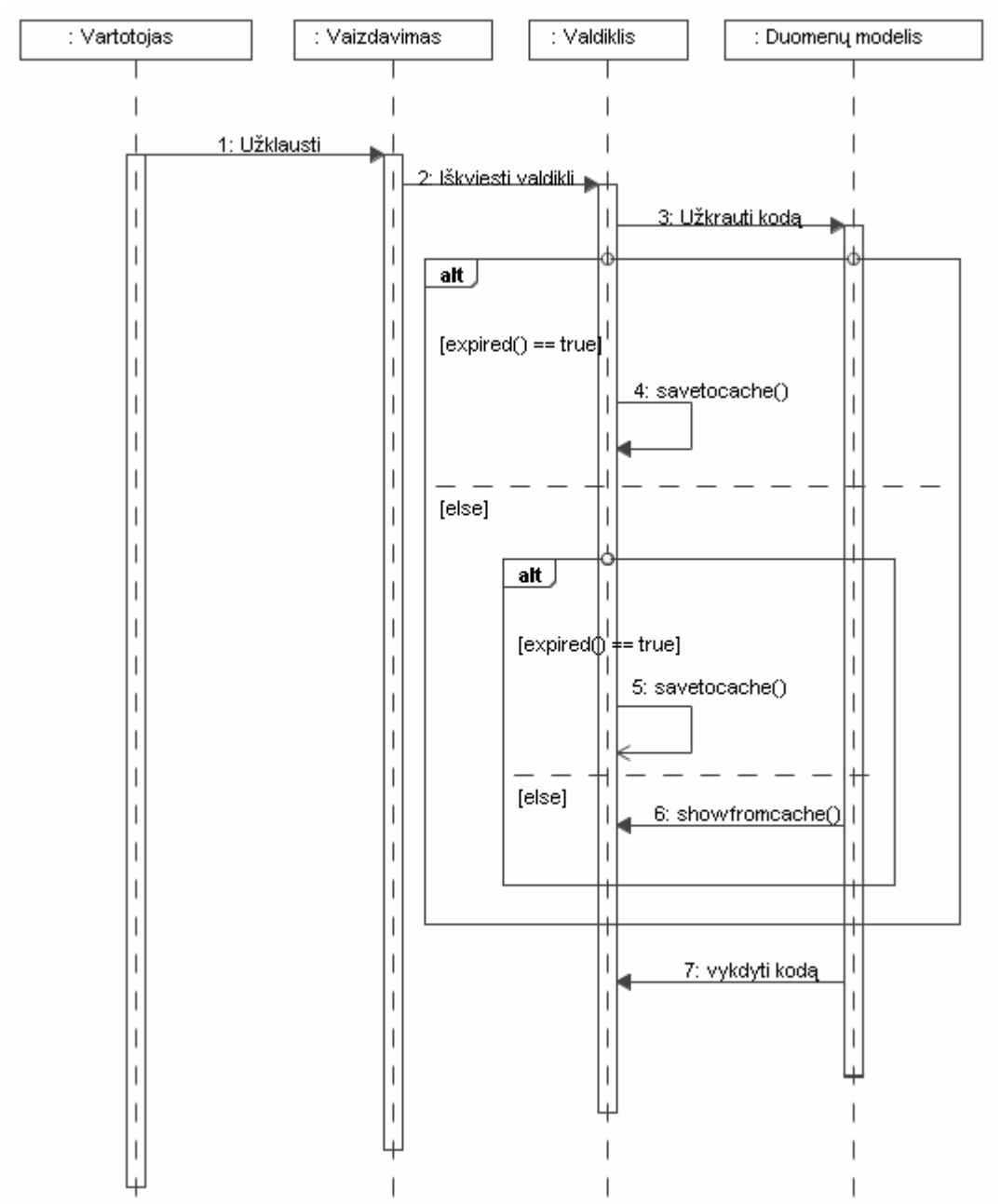
3.1. Sprendimo pagrindimas ir esmės išdėstymas

Senajame projekte saugant informaciją spartinančiojoje atmintinėje naudojamos funkcijos esančios Smarty klasėje, kurios gali būti iškviestos pasinaudojus Smarty klase. Naujajame projekte bus sukurta atskira klasė, kuri iškvies Smarty šablonų sistemos branduolio klasę, kas užtikrins greitesnį sistemos veikimą, bei laikinosios informacijos išsaugojimo mechanizmą.

3.2. Detalus projektas

14 paveiksle pateikta kodo su informacijos išsaugojimu, realizuoto pagal MVC modelį, vykdymo sekų diagrama:

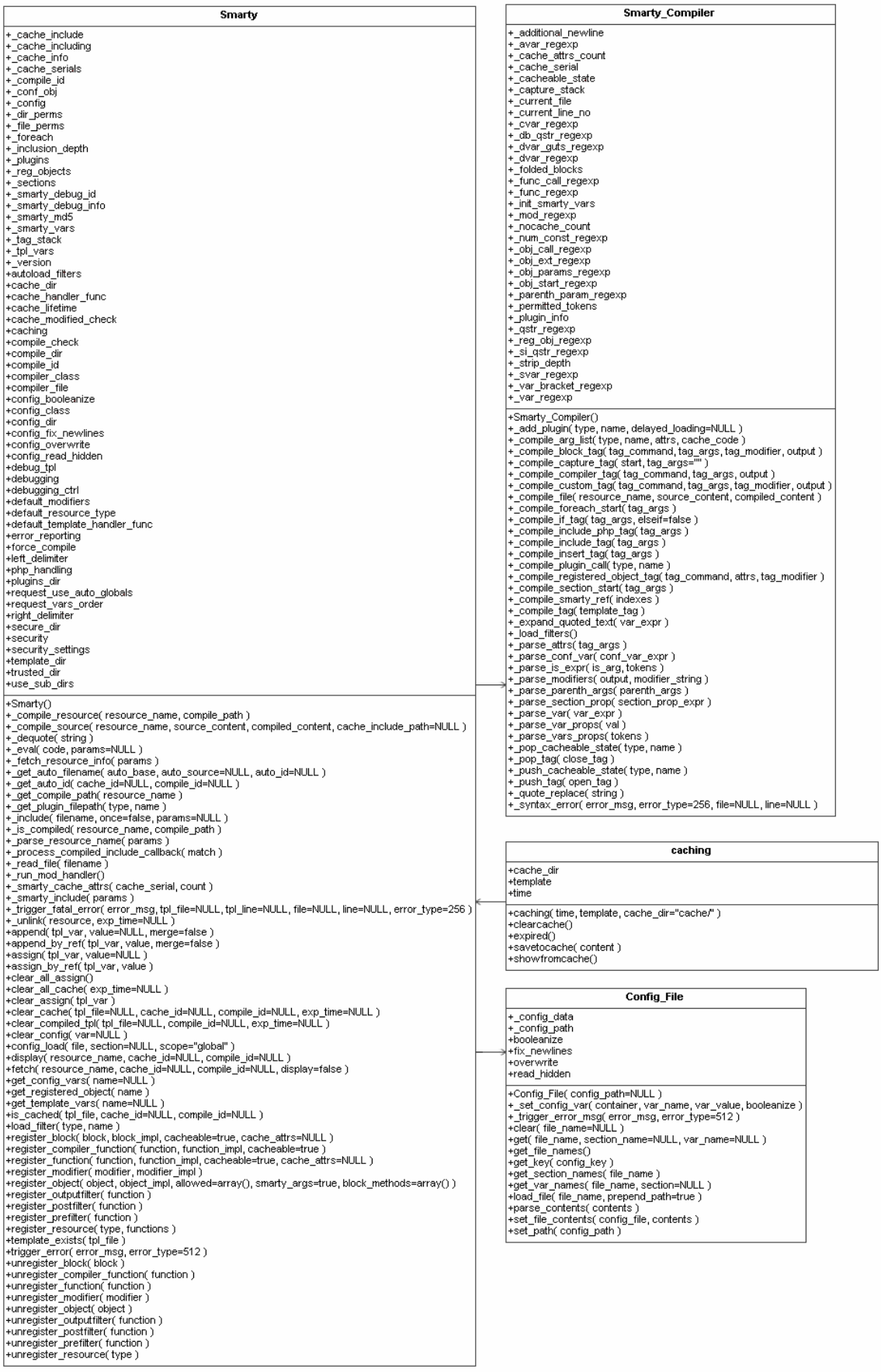
- vartotojas siunčia užklausą į serverį
- vaizdavimas kviečia valdiklį
- valdiklis užkrauna kodą
- duomenų modelis sugeneruoja kodą
  - jei kodas rastas spartinančioje atmintinėje jis užkraunamas iš spartinančiosios atmintinės
- valdiklis užkrautą kodą įvykdo
  - jei kodas nebuvo rastas spartinančioje atmintinėje po įvykdymo kodo kopija išsaugojama spartinančiojoje atmintinėje
- atnaujinamas vaizdavimas
- vartotojui grąžinama informacija



14 pav. Kodo vykdymo sekų diagrama

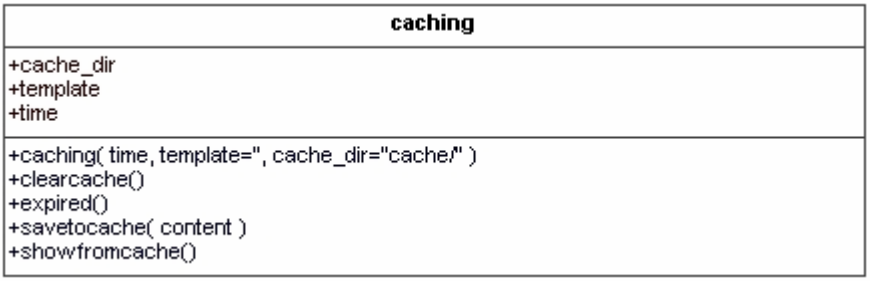
Remiantis šiuo principu realizuotas laikinosios informacijos išsaugojimo spartinančiojoje atmintinėje mechanizmas.

15 paveiksle pateikta projekto klasių diagrama su klasės kintamaisiais ir metodais:



15 pav. projekto klasių diagrama su kintamaisiais ir metodais

16 paveiksle atskirai pateikta projekto naujosios klasės diagrama su kintamaisiais ir metodais:



16 pav. informacijos išsaugojimo spartinančiojoje atmintinėje klasė

- klasės kintamieji:
  - *cache_dir* – spartinančiosios atmintinės katalogas, kuriame išsaugojami failai
  - *template* – išsaugojamo failo vardas
  - *time* – laikas sekundėmis, nurodantis kas kiek laiko bus atnaujinama išsaugojama informacija
- klasės metodai:
  - caching – klasės konstruktorius, kuriame naudojami klasės kintamieji. Kintamojo *cache_dir* nutylėtoji reikšmė yra *cache/*
  - clearcache – metodas išvalantis spartinančiosios atmintinės katalogą
  - expired – metodas, kuris tikrina ar egzistuoja išsaugojamasis failas, ir ar nepraėjo išsaugojimo laikas nustatytas kintamuoju *time*. Jeigu failas neegzistuoja ar praėjo laikas grąžinama *true*, priešingu atveju grąžinama *false*
  - savetocache – metodas išsaugojantis informaciją spartinančiojoje atmintinėje
  - showfromcache – metodas rodantis informaciją iš spartinančiosios atmintinės

3.3. Metodo, išsaugančio informaciją spartinančiojoje atmintinėje, panaudojimas

Paprasčiausio scenarijaus su informacijos išsaugojimu spartinančiojoje atmintinėje (pavaizduoto 5 paveiksle) pavyzdys:

```
01: <?php
02: // įtraukiama caching klasė
03: include "cache.php";
04:
```

```
05: // sukuriamas klasės objektas
06: // saugoma 3600s = 1h
07: // naujas - žodis papildantis išsaugajomo failo vardą;
08: // tai leidžia varijuoti išsaugojamųjų failų vardais
09: // pvz.: kazkas.php-naujas, kazkas.php-senas
10: $cache = new caching(3600, 'naujas');
11:
12: // tikrinama ar egzistuoja išsaugota kopija
13: // jei taip rodoma failo vykdymas baigiamas
14: if ( !$cache->expired() ) {  $cache->showfromcache(); }
15:
16: // priešingu atveju vykdomas failas
17: else {
18: // įjungiamas išvesties buferis
19: ob_start();
20:
21: //
22: // kažkokas kodas, DB užklausos
23: //
24:
25: // kintamajam priskiriamas išvesties buferis
26: $content = ob_get_contents();
27
28: // išvalomas ir išjungiamas išvesties buferis
29: ob_end_flush();
30:
31: // išsaugojamas turinys
32: $cache->savetocache($content);
33: }
34:?>
```

3.4. Standartinis Smarty šablonų variklio kodo vykdymas su informacijos išsaugojimu

Pavyzdinis standartinio Smarty šablonų variklio kodo su informacijos išsaugojimu spartinančiojoje atmintinėje pavyzdys:

```
01: <?php
02: // įtraukiama smarty klasė
03: include "smarty/Smarty.class.php";
04:
05: // sukuriamas objektas
06: $smarty = new Smarty;
07:
08: // išsaugojimo direktorija
09: $smarty->cache_dir = 'smarty_cache';
10:
11: // išsaugojimas įjungtas
12: $smarty->caching = 1;
13:
14: // išsaugojimo trukmė sekundėmis
15: $smarty->cache_lifetime = 3600;
16:
17: // tikrinama ar failas išsaugotas
18: if(!$smarty->is_cached('index.tpl')) {
```

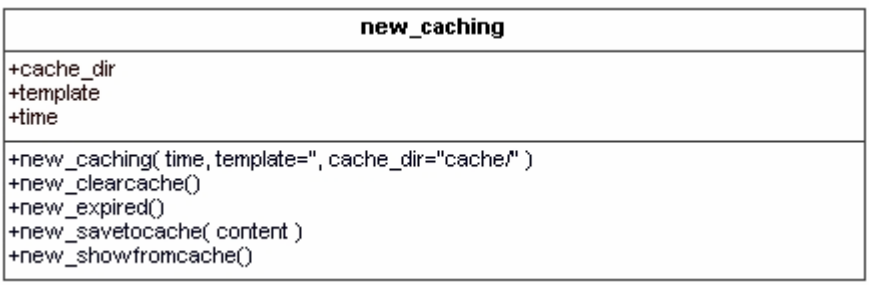
```
19:
20: //
21: //  kažkoks kodas
22: //
23:
24: }
25:
26: // rodomas šablonas
27: $smarty->display('index.tpl');
28:?>
```

Vykdant šį kodą šablonų failai kompiliuojami kartu su php failais. Tikrinama ar spartinančiojoje atmintinėje egzistuoja kopija. Jei kodas rastas jis vykdomas ir rodomas iš spartinančiosios atmintinės. Jei ne – kodas yra vykdomas ir išsaugojama kopija spartinančiojoje atmintinėje.

3.5. Metodo integravimas į Smarty šablonų variklį

Smarty šablonų variklio funkcionalumas plečiamas per įskiepius. Sukurta naujoji klasė *new_caching* (17 pav.), pavadinama *function.new_cache.php* ir išsaugojama šablonų sistemos įskiepių kataloge (*smarty/plugins/*). Kodas pateikiamas 5 – ame priede.

Kad nesidubliuotų pavadinimai, prie naujosios klasės ir metodų pavadinimų pridedamas priešdėlis *new*: *new_caching*



17 pav. integruota informacijos išsaugojimo spartinančiojoje atmintinėje klasė

3.6. Naujojo Smarty šablonų variklio kodo vykdymas su informacijos išsaugojimu

Naujajame vykdymo kode kuriamas naujosios, integruotojos klasės objektas (10 eilutė):

```
01: <?php
02: // įtraukiama smarty klasė
03: include "smarty/Smarty.class.php";
04:
05: // įtraukiama naujoji klasė
06: include "smarty/plugins/function.new_cache.php";
07:
08: // sukuriamas naujosios klasės objektas
```

```
09: // išsaugojimo laikas sekundėmis
10: $smarty = new new_caching(3600);
11:
12: // jeigu kodas rastas spartinančiojoje atmintinėje
13: // kodas rodomas iš jos ir vykdymas baigiamas
14: if ( !$smarty->new_expired() )
15: {
16:$smarty->new_showfromcache();
17: }
18:
19: // priešingu atveju
20: else {
21:// įjungiamas išvesties buferis
22: ob_start();
23:
24://
25:// kažkoks kodas
26://
27:
28:// rodomas šablonas
29:$smarty->display('index.tpl');
30:
31:// priskiriamas išvesties buferio turinys
32:$content = ob_get_contents();
33:
34:// išvalomas ir išjungiamas išvesties buferis
35:ob_end_flush();
36:
37:// turinys išsaugomas spartinančiojoje atmintinėje
38:$smarty->new_savetocache($content);
39: }
40:?>
```

Naudojamas naujasis informacijos išsaugojimo – vaizdavimo mechanizmas, tačiau pagrindinis funkcionalumas yra Smarty branduolio klasės.

Tas užtikrina greitesnį spartinančiosios atmintinės naudojimo mechanizmą ir viso šablonų variklio veikimą.

3.7. Smarty šablonų variklio kodo vykdymas su dvigubu informacijos išsaugojimu

Dvigubas informacijos išsaugojimas – tai abiejų spartinančiosios atmintinės funkcijų panaudojimas viename vykdomajame kode.

```
01: <?php
02: include "cache.php";
03:
04: $cache = new caching(3600, 'naujas');
05:
06: if ( !$cache->expired() ) {  $cache->showfromcache(); }
07: else {
08:
09:include "smarty/Smarty.class.php";
10:
11:ob_start();
12:$smarty = new Smarty;
13:$smarty->cache_dir = 'cache';
```

```
14: $smarty->caching = 0;
15: $smarty->cache_lifetime = 900;
16:
17:
18: if(!$smarty->is_cached('index.tpl')) {
19:             //
20:             // kazkoks kodas
21:             //
22: }
23:
24: $smarty->display('index.tpl');
25: $content = ob_get_contents();
26: ob_end_flush();
27: $cache->savetocache($content);
28:  }
29: ?>
```

4. Eksperimentinis tyrimas

4.1. Reikalavimai

Smarty reikalauja web serverio palaikančio PHP 4.0.6 arba aukštesnę versiją.

4.2. Diegimo aprašymas

Smarty šablonų variklį sudaro PHP bibliotekos (*libs*/ direktorija)

Reikalingi Smarty bibliotekų failai

```
Smarty-v.e.r/
   libs/
      Smarty.class.php
      Smarty_Compiler.class.php
      Config_File.class.php
      debug.tpl
      internals/*.php  (visi failai)
      plugins/*.php    (visi failai)
```

Smarty naudoja PHP konstantą pavadintą *SMARTY_DIR*, kuri nustato pilną kelią iki Smarty bibliotekų (libs/) katalogo. Iš esmės šio kintamojo nereikia nustatinėti jeigu naudojami absoliutūs keliai iki programos kode.

***SMARTY_DIR* konstantos nustatymo pavyzdys**

```
01: <?php
02: // *nix sistemose
03: define('SMARTY_DIR', '/usr/local/lib/Smarty-v.e.r/libs/');
04:
05: // windows sistemose
06: define('SMARTY_DIR', 'c:/webroot/libs/Smarty-v.e.r/libs/');
07:
08: // smarty klasės įtraukimas į kodą
09: require_once(SMARTY_DIR . 'Smarty.class.php');
10: $smarty = new Smarty();
11: ?>
```

Reikalingi katalogai Smarty veikimui

Smarty reikalauja keturių katalogų:

- *templates*/ - šablonų katalogas
- *templates_c*/ - sukompiliuotų šablonų katalogas
- *configs*/ - konfigūracijos failai
- *cache*/ - spartinančiosios atmintinės katalogas

Šių keturių direktorijų keliai sistemoje nustatomi Smarty klasės kintamuosiuose:

- *$templates_dir*
- *$compile_dir*
- *$config_dir*
- *$cache_dir*

Rekomenduojama šių direktorijų kelius nustatyti atskirai kiekvienam projektui, kurie naudoja Smarty šablonų variklį. *templates_c* ir *cache* katalogams būtina nustatyti rašymo teises.

Pavyzdinė failų struktūra

```
/usr/local/lib/Smarty-v.e.r/libs/
        Smarty.class.php
        Smarty_Compiler.class.php
        Config_File.class.php
        debug.tpl
        internals/*.php
        plugins/*.php

/web/www.example.com/
                templates/
                    index.tpl
                templates_c/
                configs/
                cache/
                htdocs/
                    index.php
```

Pavyzdinis vykdomasis failas

```
01:<?php
02:
03: require_once(SMARTY_DIR . 'Smarty.class.php');
04:
05: $smarty = new Smarty();
06:
07: $smarty->template_dir = '/web/www.example.com/templates/';
08: $smarty->compile_dir  = '/web/www.example.com/templates_c/';
09: $smarty->config_dir   = '/web/www.example.com/configs/';
10: $smarty->cache_dir    = '/web/www.example.com/cache/';
11:
12: $smarty->display('index.tpl');
13:
14:?>
```

Smarty funcionalumas yra plečiamas per įskiepius. Nauja spartinančiosios atmintinės valdymo klasė (*function.new_cache.php*) išsaugoma įskiepių kataloge *plugins*.

Priešingai nei prieš tai esančiame failo pavyzdyje, Smarty objektas bus kuriamas naujosios klasės (tačiau naujasis objektas naudoja kai kurias Smarty klasės funkcijas).

Pavyzdinis naujasis vykdomasis failas

```
01: <?php
02:
03: require_once(SMARTY_DIR . 'Smarty.class.php');
04: require_once(SMARTY_DIR . 'plugins/function.new_cache.php');
05:
06: $smarty = new Caching(900);
07:
```

```
08: $smarty->template_dir = '/web/www.example.com/templates/';
09: $smarty->compile_dir  = '/web/www.example.com/templates_c/';
10: $smarty->config_dir   = '/web/www.example.com/configs/';
11: $smarty->cache_dir    = '/web/www.example.com/cache/';
12:
13: $smarty->display('index.tpl');
14:?>
```

4.3. Testavimo modelis bei duomenys, kontrolinis pavyzdys

Testuojami bus trys vykdymo scenarijai aprašyti (4.4, 4.6, 4.7 skyriuose) – standartinis Smarty šablonų variklio scenarijus, scenarijus naudojantis naująjį išsaugojimo mechanizmą ir scenarijus naudojantis dvigubą laikinosios informacijos išsaugojimo mechanizmą. Pirmajame etape kodas bus vykdomas su įjungtu informacijos išsaugojimu, antrajame – be informacijos išsaugojimo spartinančiojoje atmintinėje.

Bus skaitomas ~300kb dydžio XML failas (ištrauka pateikta 2-ame priede), turintis apie 3000 elementų (daugiau nei 14000 eilučių) iš kurių išrenkamos 7 reikšmės, surašomos į masyvą ir išvedamos pasinaudojus šablonų failais (pateikta 3-ame priede).

Testas imituos 1 vartotoją atverčiantį puslapį kas 1 - 2 sekundes, kas iš viso sudarytų apie 30 - 60 puslapio atvertimų per minutę (apie 2500 - 3500 atvertimų per valandą).

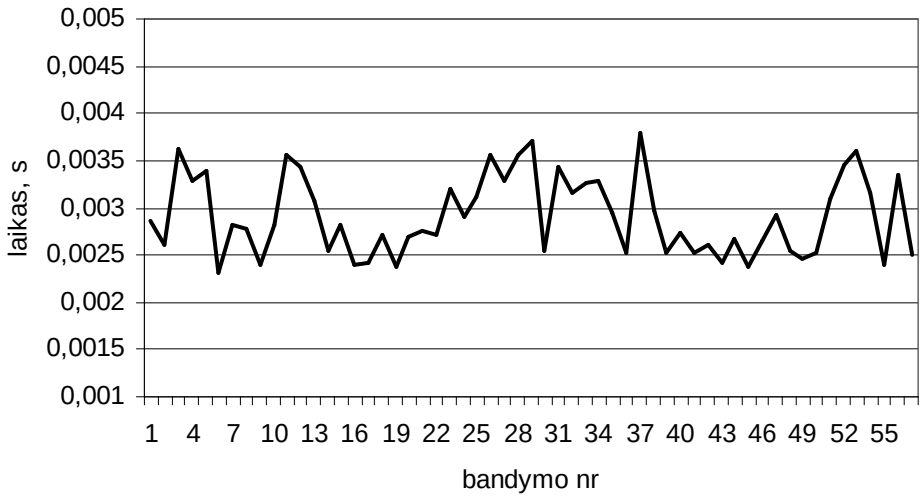
Testas nr 1

Pirmame testo etape naudojamas informacijos išsaugojamas spartinančiojoje atmintinėje.

- Smarty šablonų variklio standartinio scenarijaus vykdymo rezultatai

18 paveiksle pateiktas vykdymo scenarijaus laiko kreivė.

Vidutinis kodo įvykdymo laikas - 0,002914048 s.

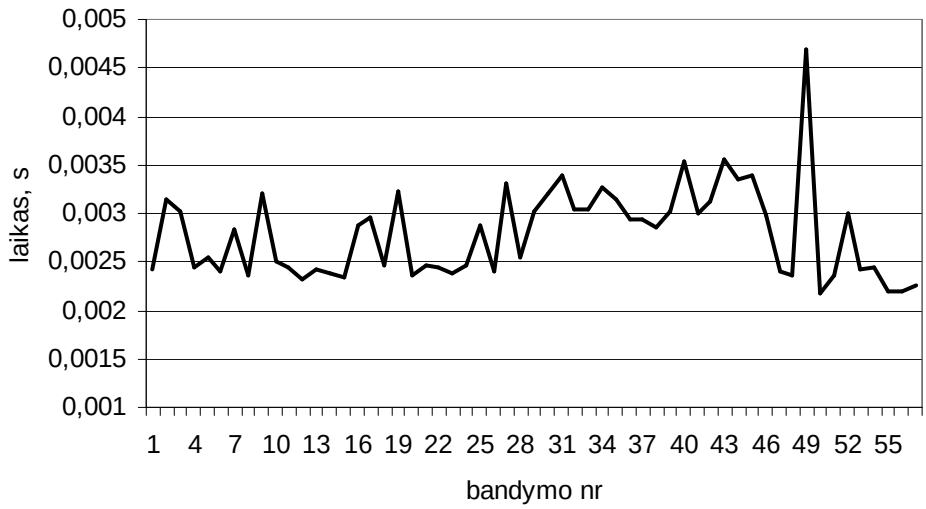


18 pav. standartinio scenarijaus vykdymo laikas

- Smarty šablonų variklio vykdymo scenarijaus rezultatai naudojant naująją funkciją

19 paveiksle pateiktas scenarijaus, naudojančio naująjį informacijos išsaugojimo mechanizmą, vykdymo laiko kreivė.

Vidutinis scenarijaus įvykdymo laikas – 0,002781165 s.

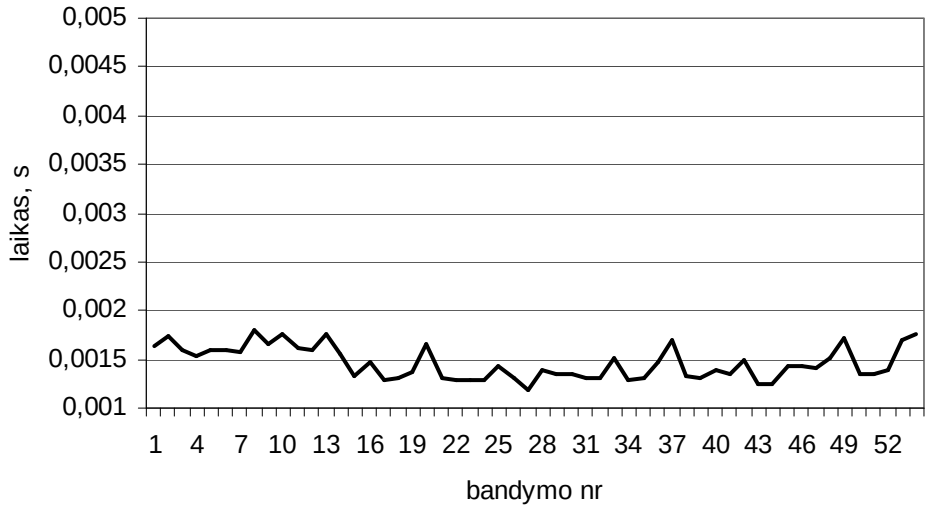


19 pav. scenarijaus naudojančio naująjį mechanizmą vykdymo laiko kreivė

- Smarty šablonų variklio vykdymo scenarijus su dvigubu informacijos išsaugojimu

20 paveiksle pateiktas scenarijaus, naudojančio dvigubą informacijos išsaugojimą, vykdymo laiko kreivė.

Vidutinis scenarijaus įvykdymo laikas - 0,001461378 s.



20 pav. scenarijaus, naudojančio dvigubą informacijos išsaugojimą, vykdymo laiko kreivė

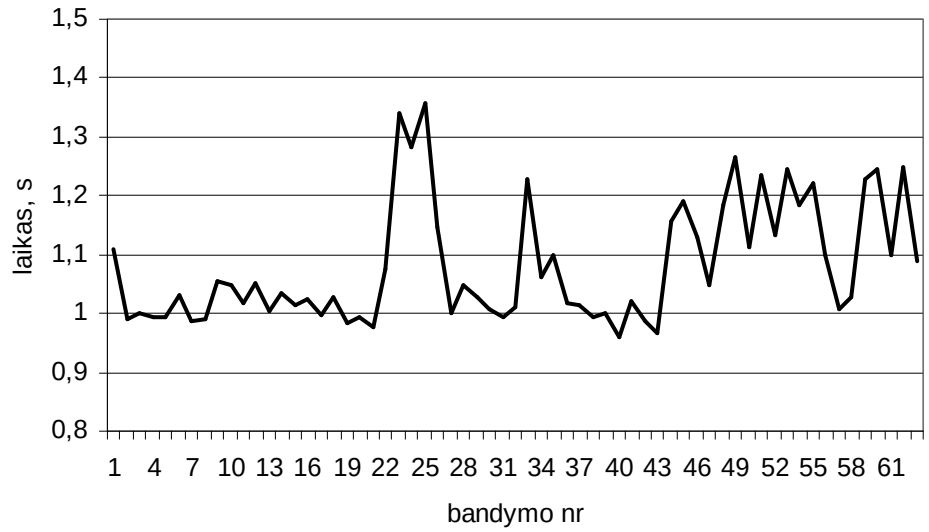
Testas nr 2

Antrame testo etape informacija nesaugojama spartinančiojoje atmintinėje.

- Smarty šablonų variklio standartinio scenarijaus vykdymo rezultatai, nesaugant informacijos spartinančiojoje atmintinėje

21 paveiksle pateiktas vykdymo scenarijaus laiko kreivė.

Vidutinis scenarijaus įvykdymo laikas - 1,081347946 s.

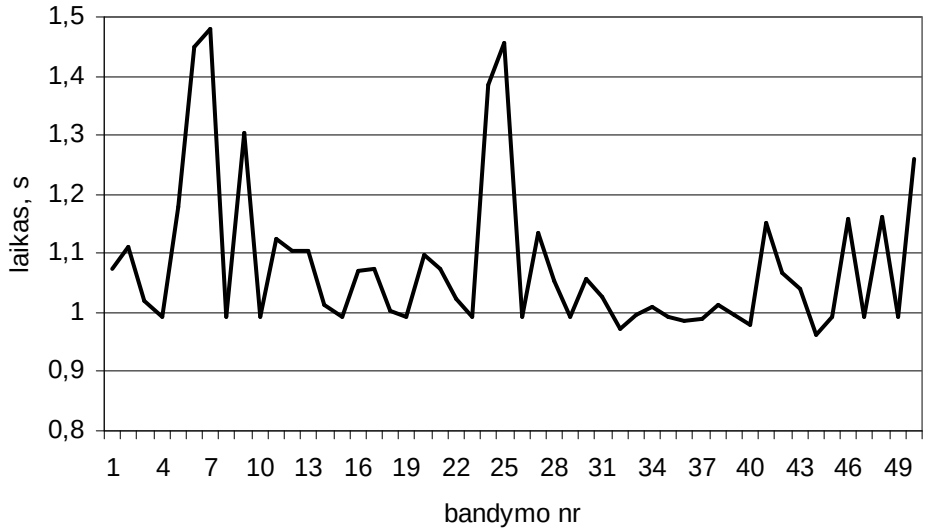


21 pav. standartinio scenarijaus vykdymo laiko kreivė (nesaugant informacijos)

- Smarty šablonų variklio vykdymo scenarijaus rezultatai naudojant naująjį mechanizmą

21 paveiksle pateiktas scenarijaus, naudojančio naująjį mechanizmą, vykdymo laiko kreivė.

Vidutinis scenarijaus įvykdymo laikas – 1,088661402 s.

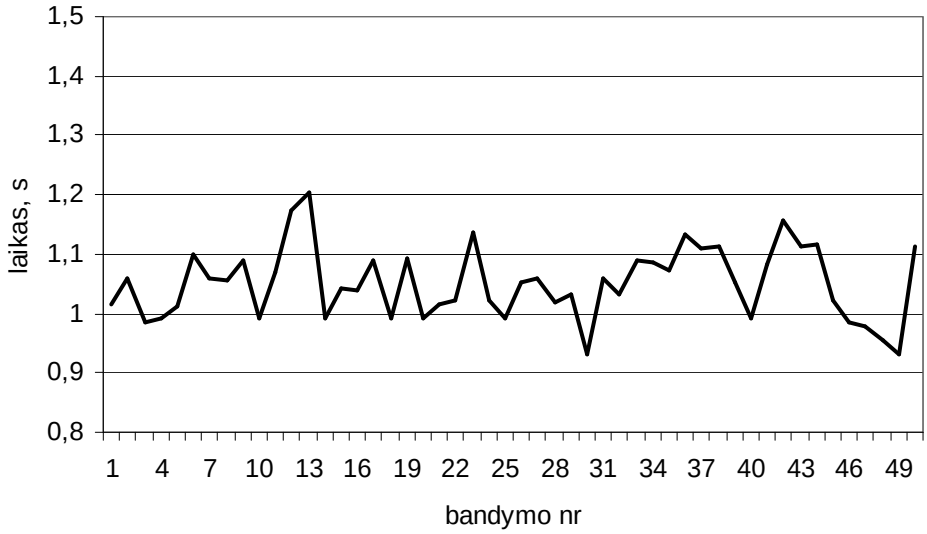


22 pav. scenarijaus, naudojančio naująjį mechanizmą, vykdymo laiko kreivė (nesaugant informacijos)

- Smarty šablonų variklio vykdymo scenarijus naudojantis dvigubą vaizdavimo logiką (be informacijos išsaugojimo)

23 paveiksle pateikta scenarijaus, vykdymo laiko kreivė.

Vidutinis scenarijaus įvykdymo laikas - 1,050525095 s.

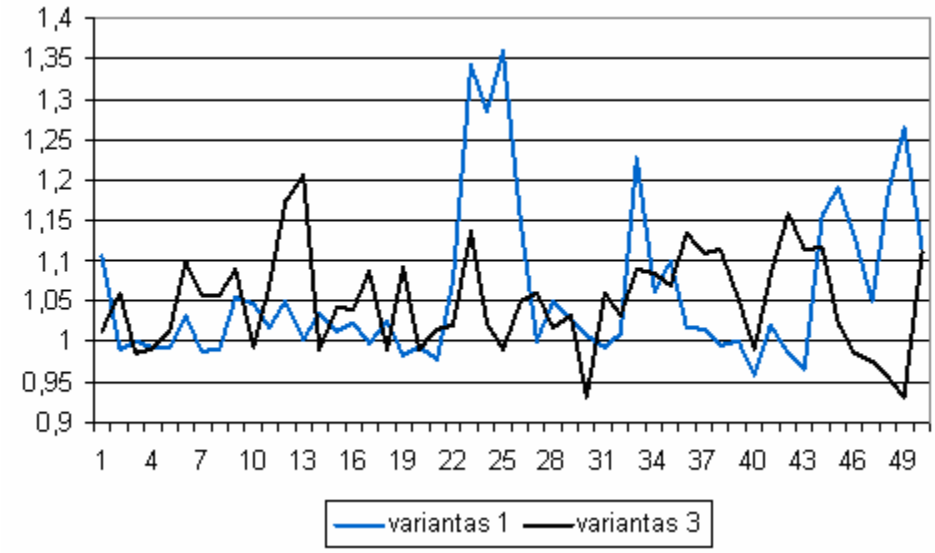


23 pav. scenarijaus, naudojančio dvigubą vaizdavimo logiką, vykdymo laiko kreivė

1-ojo ir 2-ojo testo rezultatai parodė, kad greičiausiai kodą įvykdo scenarijus naudojantis dvigubą informacijos vaizdavimo – išsaugojimo mechanizmą. Standartinis Smarty scenarijus parodė prasčiausius rezultatus.

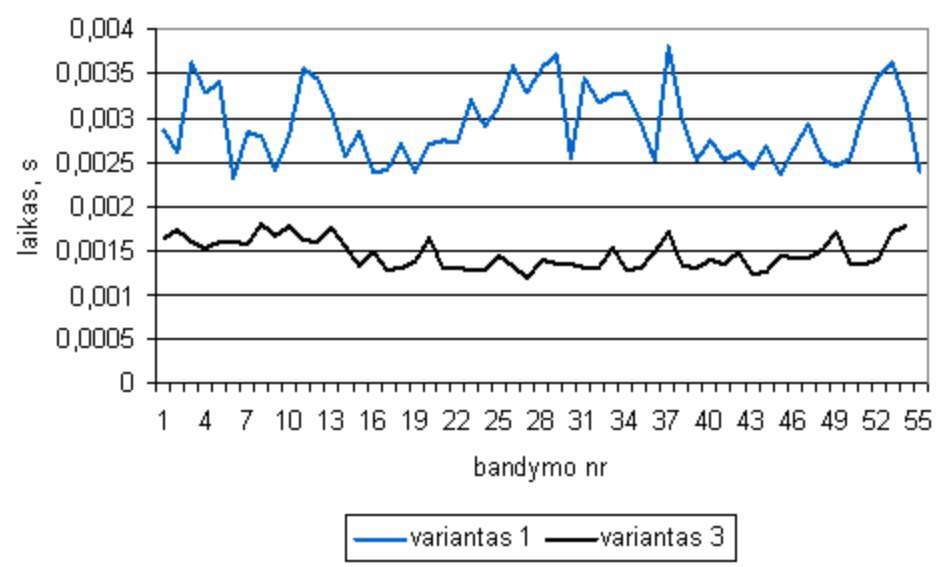
Testų rezultatų apibendrinimas

Nenaudojant informacijos išsaugojimo vykdymo rezultatai yra panašūs. (pav 24). Šuoliai atsiradę testavimo rezultatuose parodo, kad informacijos pateikimas vartotojui taip pat priklauso nuo serverio bet tinklo apkrovimo.



24 pav. vykdymo laiko kreivės išjungus informacijos išsaugojimą

25 paveiksle pateiktas standartinio Smarty kodo vykdymo ir dvigubo vaizdavimo mechanizmo palyginimas. Nenaudojant informacijos išsaugojimo vykdymo rezultatai yra panašūs.



25 pav. vykdymo laiko kreivės naudojant informacijos išsaugojimą

Bandymai parodė, kad naudojant naująją funkciją ir laikinai išsaugant tarpinę informaciją spartinančiojoje atmintinėje galima sutaupyti nuo 5% iki 50% vykdymo ir informacijos pateikimo laiko (priklausomai nuo kodo sudėtingumo ir serverio apkrovimo vidurkių).

Išvados

- Išanalizavus laikinosios informacijos išsaugojimo būdus buvo pasirinktas programiškai paremtas laikinosios informacijos išsaugojimas naudojant šablonų variklius, nes tai yra pigiausias ir paprasčiausias būdas optimizuoti resursus, ir sumažinti kodo vykdymo laiką
- Išsiaiškinta, kad pagrindinis veiksnys įtakojantis programiškai paremtą informacijos išsaugojimo būdą yra techninė įranga.
- Buvo suformuluota projekto reikalavimų specifikacija ir suprojektuotas dalykinės srities modelis.
- Suprojektuota ir realizuota nauja funkcija su informacijos išsaugojimo laikinojoje atmintinėje mechanizmu ir integruota į Smarty šablonų variklį.
- Atliktas naujosios funkcijos pavyzdinio kodo eksperimentas, kuris parodė geresnius rezultatus nei standartinis Smarty kodo vykdymo scenarijus.

Literatūra

1. M. Ohrt, A. Zmievski. Smarty - the compiling PHP template engine. 2006-09-27. [žiūrėta 2006-10-23]. Prieiga per Internetą: http://smarty.php.net/manual/en/
2. M. Achour, F. Betz, A. Dovgal ir kt. PHP manual. 2006-09-19. [žiūrėta 2006-10-23]. Prieiga per Internetą: http://lt.php.net/manual/en/
3. A Kirk. Caching Strategies for Load Reduction on High Traffic Web Applications. [žiūrėta 2006-10-23]. Prieiga per Internetą: http://alexander.kirk.at/papers/caching-strategies/
4. Model-view-controller [žiūrėta 2007-01-14]. Prieiga per Internetą: http://en.wikipedia.org/wiki/Model-view-controller
5. Top 25 PHP template engines. žiūrėta 2007-01-15]. Prieiga per Internetą: http://php5powerprogramming.com/links/top_php_template_engines.php
6. Heyes Template Class. [žiūrėta 2007-01-15]. Prieiga per Internetą: http://www.phpguru.org/static/template.html
7. FastTemplate. [žiūrėta 2007-01-15]. Prieiga per Internetą: http://www.thewebmasters.net/php/FastTemplate.phtml
8. MySQL Query Cache. [žiūrėta 2007-10-08]. Prieiga per Internetą: http://dev.mysql.com/doc/refman/5.0/en/query-cache.html
9. bTemplate. [žiūrėta 2007-01-15]. Prieiga per Internetą: http://www.massassi.com/bTemplate
10. Savant. [žiūrėta 2007-01-15]. Prieiga per Internetą: http://phpsavant.com/yawiki
11. ETS - easy template system. [žiūrėta 2007-01-15]. Prieiga per Internetą: http://ets.sourceforge.net
12. AvanTemplate [žiūrėta 2007-01-15]. Prieiga per Internetą: http://avantemplate.sourceforge.net
13. Grafx Software's Fast Template. [žiūrėta 2007-01-15]. Prieiga per Internetą: http://www.grafxsoftware.com/product.php?id=26
14. TemplatePower. [žiūrėta 2007-01-15]. Prieiga per Internetą: http://templatepower.codocad.com
15. Cached Fast Template. [žiūrėta 2007-01-15]. Prieiga per Internetą: http://xkahn.zoned.net/php/fasttemplate
16. phemplate. [žiūrėta 2007-01-15]. Prieiga per Internetą: http://pukomuko.esu.lt/phemplate

17. Turck MMCache. [žiūrėta 2007-06-16]. Prieiga per Internetą: http://turck-mmcache.sourceforge.net/index_old.html
18. Reverse proxy. [žiūrėta 2007-10-29]. Prieiga per Internetą: http://en.wikipedia.org/wiki/Reverse_proxy
19. Load balancing (computing). [žiūrėta 2007-10-29]. Prieiga per Internetą: http://en.wikipedia.org/wiki/Load_balancing_(computing)

1 Priedas

1-ojo testo šablono failas be kintamųjų.

```
Column selection

You can select text by column using Alt + mouse drag. This feature is not available in
word-wrap mode.

Multiple settings
            You can add or remove file types and specify different options for each file
types. For example, you can set word-wrap feature off in a plain text file and turn it on
automatically when you load an HTML document. You can specify multiple settings in
Settings & Syntax page of the Preferences dialog box. {$var4}
Edit large files
            Unlike Windows' built-in Notepad, it can handle large text files. The file
size is only limited by the amount of free system memory.

Powerful undo/redo
            EditPlus allows multiple undo/redo, so you can safely cancel any typing
error.
Word wrap
            Word-wrap feature helps to edit long lines more conveniently. You can turn
on/off word-wrap feature through Word Wrap command on Document menu.
Line number
            Line numbers can improve readability of HTML documents and program source
codes. You can show or hide line number through Line Number command on Document menu.
Ruler

Ruler improves readability and helps you find the cursor location quickly. You can show
or hide ruler through Ruler command of View menu.
'Drag and drop' editing
            EditPlus supports OLE 'drag and drop' editing which is more efficient than
clipboard commands. You can also drag and drop text between Cliptext window and the
document.
Powerful search and replace{$var5}
            EditPlus supports powerful search and replace command which can handle
Regular Expressions. EditPlus also supports Find in Files command so you can search text
in multiple files. You can also set markers at a specific line and go to the marker
quickly from any part of the document. See Search menu commands for more details.

Spell checker
            EditPlus supports spell checker so you can correct typing errors in your
document easily. Currently, only English dictionary is supported.
Splitter window
            EditPlus allows user to divide the document window into several panes and
edit different part of the file concurrently. To divide current document, run Split
command on Window menu.
Keystroke recording
            You can record keystrokes and playback it later. Use Record Keystrokes
command on Tools menu.

Customizable hotkeys
            You can customize hotkeys of all commands of EditPlus. See Keyboard page of
the Preferences dialog box.
Monitor clipboard
            This option can speed up repeated cut-and-paste operations. Any text that is
copied or cut to clipboard will automatically be appended to current document. See
Monitor Clipboard command.
Column Marker
            Column marker is a vertical line which indicates specific column location.
Column marker can be useful for column-oriented programming languages such as COBOL or
FORTRAN. See Column Marker command.
```

2 Priedas

2-ojo testo trečios šalies duomenų failo pavyzdys.

```
<?xml version="1.0" encoding="UTF-8"?>
<weathers><weather><item><order>1000</order>
<road_id>132</road_id>
<name>Seirijai</name>
<dist>15</dist>
```

```
<time>10-20 21:30</time>
<krit_tipas>nėra</krit_tipas>
<krit_kiekis>0</krit_kiekis>
<visibility>-</visibility>
<air_temp>6</air_temp>
<rasos_temp>5</rasos_temp>
<road_temp>7,6</road_temp>
<uzsal_temp>0</uzsal_temp>
<road_danga> *</road_danga>
<wind_speed_max>2</wind_speed_max>
<wind_speed_avg>1</wind_speed_avg>
<wind_direction>PR</wind_direction>
<image_x>5612</image_x>
<image_y>7195</image_y>
</item><item><order>1001</order>
<road_id>132</road_id>
<name>Seirijai</name>
<dist>15</dist>
<time>10-20 21:00</time>
..
</weather>
</weathers>
```

3 Priedas

2-ojo testo vaizdavimo failas (šablonas).

```
<html>
<head>
<title>test</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</style>
</head>
<body>
<div style="position: absolute; left: 600px; bottom: 170px;">
<h1>Orai</h1>
Vilnius: {$temp.vilnius.temp}° C<br />
Kaunas: {$temp.kaunas.temp}° C<br />
Klaipėda: {$temp.klaipeda.temp}° C<br />
Šiauliai: {$temp.siauliai.temp}° C<br />
Panevėžys: {$temp.panevezys.temp}° C<br />
Alytus: {$temp.alytus.temp}° C<br />
Marijampolė: {$temp.marijampole.temp}° C<br /><br />
</div>

</body>
</html>
```

4 Priedas

1-ojo testo vykdomasis php failas.

```
<html>
<head>
            <title>test </title>
            <META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
</head>
<body>
<?
flush();

set_time_limit(3600);

include_once('benchmark.class.php');
$bench = new benchmark();

include_once('phemplate/phemplate.class.php');
include_once('heyes_template/class.template.inc');
include_once('templatepower2/class.TemplatePower.inc.php');
include_once('smarty/Smarty.class.php');
include_once('ets/ets.php');


$run_2_1 = 10;
```

```
$templates=Array(
             'tpl_power' => "Template Power",
             'phem_tpl' => "phemplate",
             'heyes_tpl' => "heyes class",
             'smarty' => "Smarty",
             'ets' => 'Easy Template System',

);



for ($i = 0; $i < 100; $i++)
{
             $tmp = array(
                          'id' => 20,
                          'name' => "longname$i",
                          'param1' => $i,
                          'param2' => 'eiho',
                          'param3' => 'hello world'
                          );
             $loop[] = $tmp;
}


foreach( $templates as $id => $name )
{
             $bench->run($run_2_1, $id.'_2_1' );
             $bench->add_result($bench->get(), $name );
}

$bench->stats();
$bench->print_stats("TEST :: text 10kb // loop 100 records // $run_2_1 iterations");

$bench->reset();


/*******************************************************/

function phem_tpl_2_1()
{
             global $loop;
             $tpl = new phemplate();

             $tpl->set_file('test', 'phemplate/test_2_1.html');

             // 20 variables
             $tpl->set_loop('list', $loop);
             $tpl->process('', 'test', 1);
}

function heyes_tpl_2_1()
{
             global $loop;

             $tpl = new template();

             $tpl->load_file('test', 'heyes_template/test_2_1.html');


             $tpl->parse_loop('test', 'loop');
             $tpl->parse('test');
}


function tpl_power_2_1()
{
             global $loop;

             $tpl = new TemplatePower( 'templatepower2/test_2_1.html');
             $tpl->prepare();

             $size = count($loop);
             for ($i = 0; $i < $size; $i++)
             {
                          //create a new number_row block
                          $tpl->newBlock("loop_row");
```

```
                        //assign values
                        $tpl->assign("id", $loop[$i]['id']);
                        $tpl->assign("name", $loop[$i]['name']);
                        $tpl->assign("param1", $loop[$i]['param1']);
                        $tpl->assign("param2", $loop[$i]['param2']);
                        $tpl->assign("param3", $loop[$i]['param3']);
                }

                //print the result
                $tpl->getOutputContent();

}

function smarty_2_1()
{
                global $loop;

                $tpl = new Smarty;
                $tpl->template_dir = './smarty/templates';
                $tpl->compile_dir = './smarty/templates_c';
                $tpl->compile_check = false;
                $tpl->caching = 0;


                $tpl->assign($loop);
                $tpl->fetch('test_2_1.html');
}



function ets_2_1()
{
                global $oloop;
                $tpl->oloop = $oloop;
                sprintt($tpl, 'ets/test_2_1.html');
}


?>


</body>
</html>
```

5 Priedas

Į Smarty šablonų variklį integruota spartinančiosios atmintinės klasė

```
<?
class new_caching extends Smarty{
                var $time;
                var $template;
                var $cache_dir;

                function new_caching($time, $template = '', $cache_dir = "cache/") {
                                $this->time = $time;
                                $this->template =
filemtime($_SERVER['SCRIPT_FILENAME'])."".str_replace('/', '-',
$_SERVER['REQUEST_URI'])."-".$template;
                                $this->cache_dir = $cache_dir;
                }

                function new_clearcache() {
                                if ($handle = opendir($this->cache_dir)) {
                                while (($file = readdir($handle)) !== false)
                                if (is_file($this->cache_dir.$file)) unlink($this-
>cache_dir.$file);
                                closedir($handle);
                                }
                }

                function new_expired() {
                                if (!file_exists($this->cache_dir.$this->template)) return
true;
                                else {
```

```
                                        $time = filemtime($this->cache_dir.$this-
>template);
                                        if ($time+$this->time >= time()) return false;
                                        else unlink($this->cache_dir.$this->template);
                                        return true;
                        }
        }

        function new_showfromcache() {
                        if (!file_exists($this->cache_dir.$this->template)) return
false;
                        $handle = fopen($this->cache_dir.$this->template, "r");
                        $contents = fread($handle, filesize($this->cache_dir.$this-
>template));
                        fclose($handle);
                        echo $contents;
                        return true;
        }

        function new_savetocache($content) {
                        $handle = fopen($this->cache_dir.$this->template, "w");
                        $contents = fwrite($handle, $content);
                        fclose($handle);
        }
}
?>
```