РОССИЙСКИЕ ИНТЕРНЕТ-ТЕХНОЛОГИИ
ВЫСОКИЕ НАГРУЗКИ

Сервер приложений C++

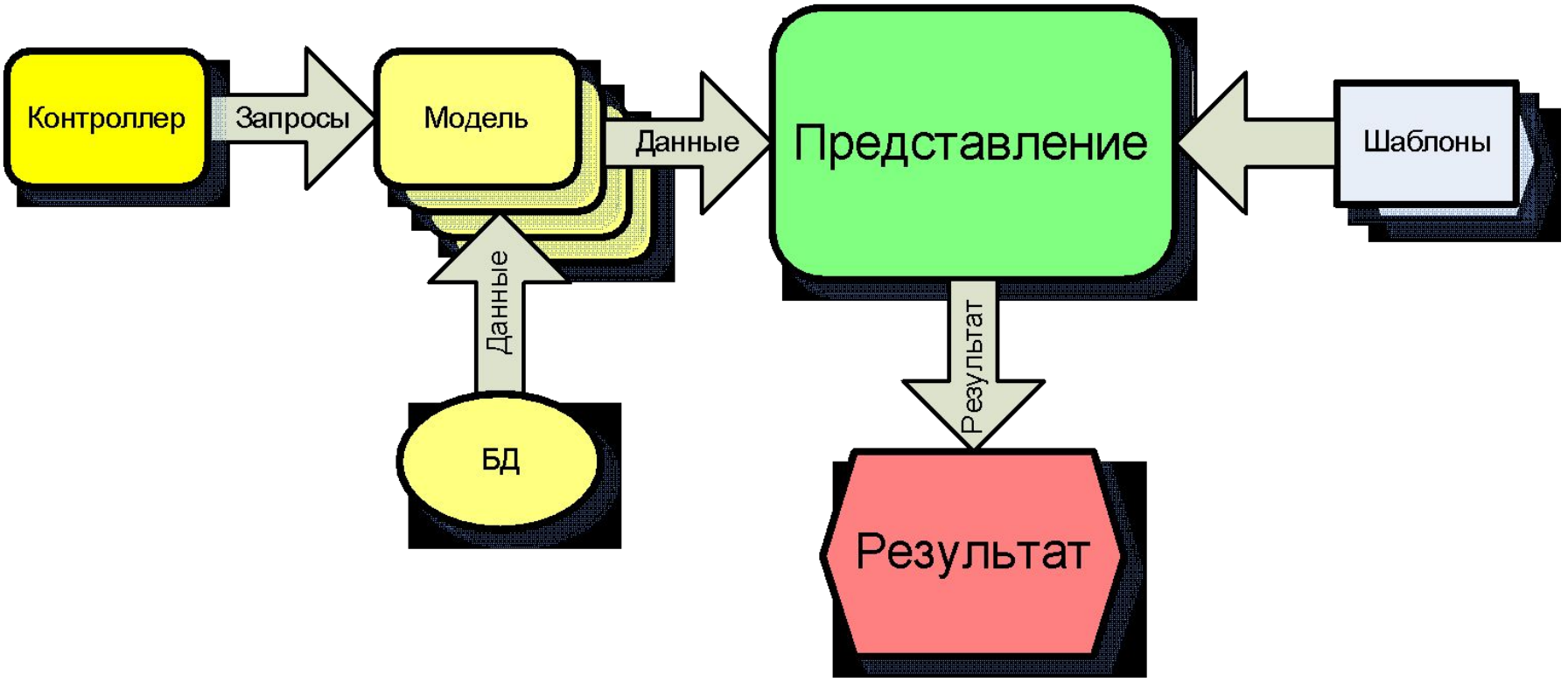
Андрей Шетухин, Илья Космодемьянский
SUP Fabrik

- первые наработки - 2003 год
- нагруженный проект, много запросов, мало памяти, мало CPU
- компьютеры выросли, но выросли и нагрузки; ничего не изменилось
- новые требования: модульность, упрощение API, переносимость

Парадигма MVC и CAS

- что такое MVC и зачем оно нужно?
- наши модификации
- достоинства предложенной схемы
- архитектура CAS

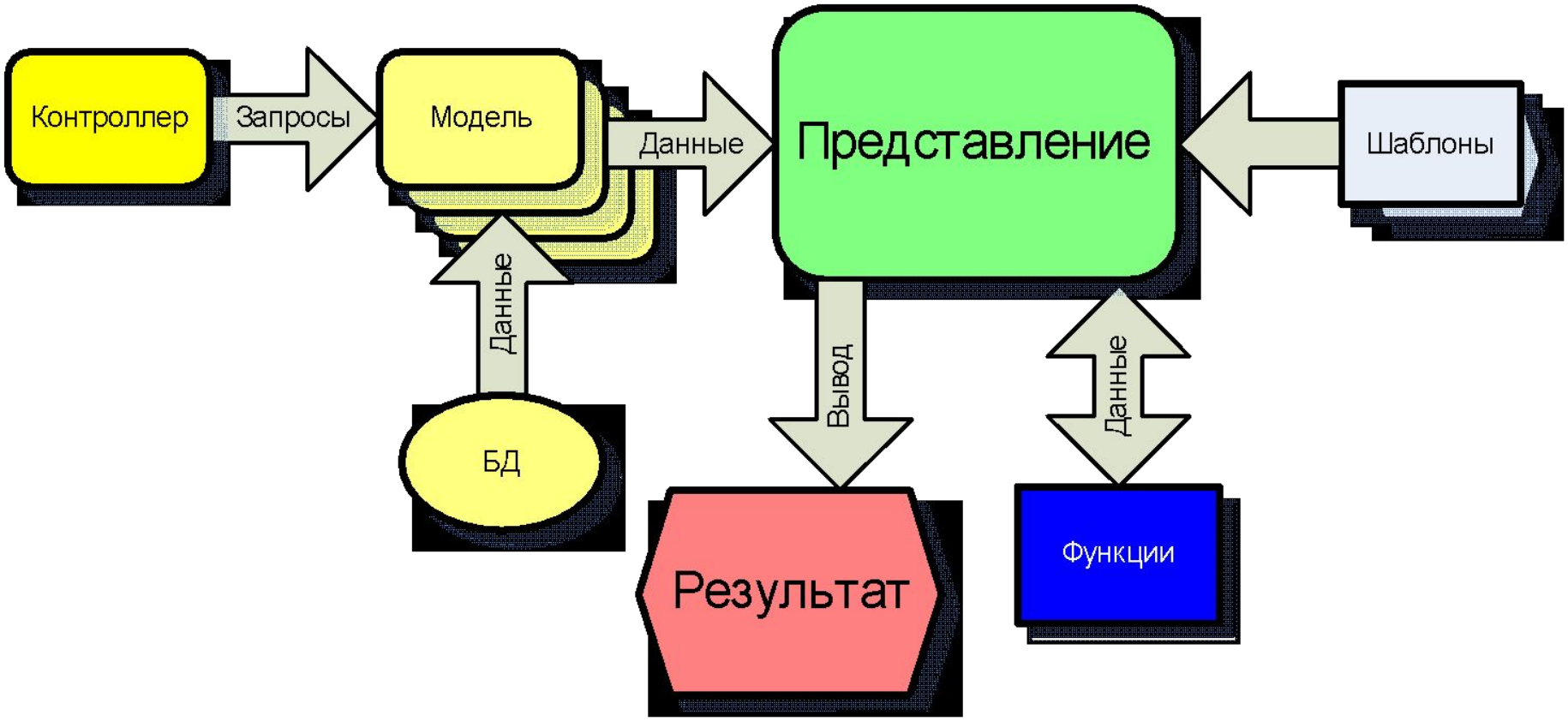
Классическая архитектура MVC



Критика

- плохо работает под большой нагрузкой
- сложность разработки моделей, контроллеров и представлений
- ненужный код в моделях
- проблемы с масштабированием

Модификация MVC

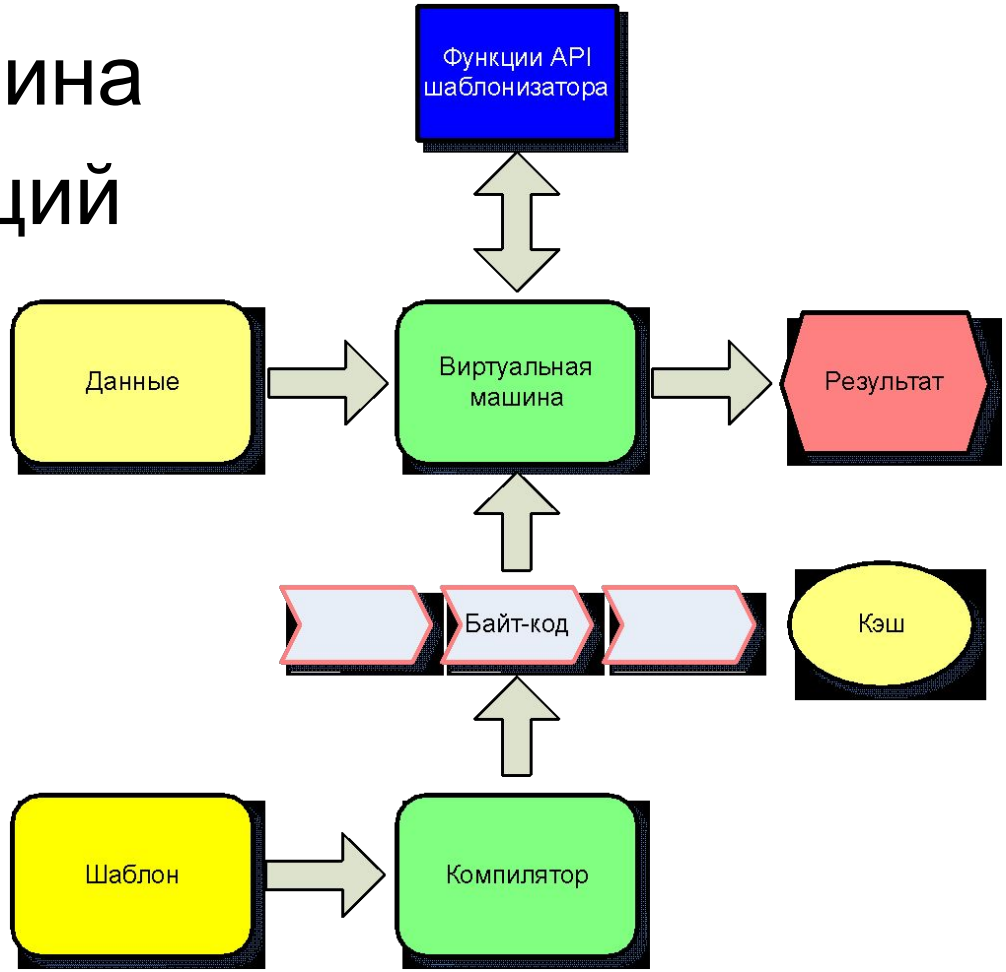


Достоинства

- модели – универсальные
- формирование ответа – только в представлении
- для генерации HTML/JSON/XML кроме шаблона ничего не требуется
- простота внесения правок
- высокая скорость работы

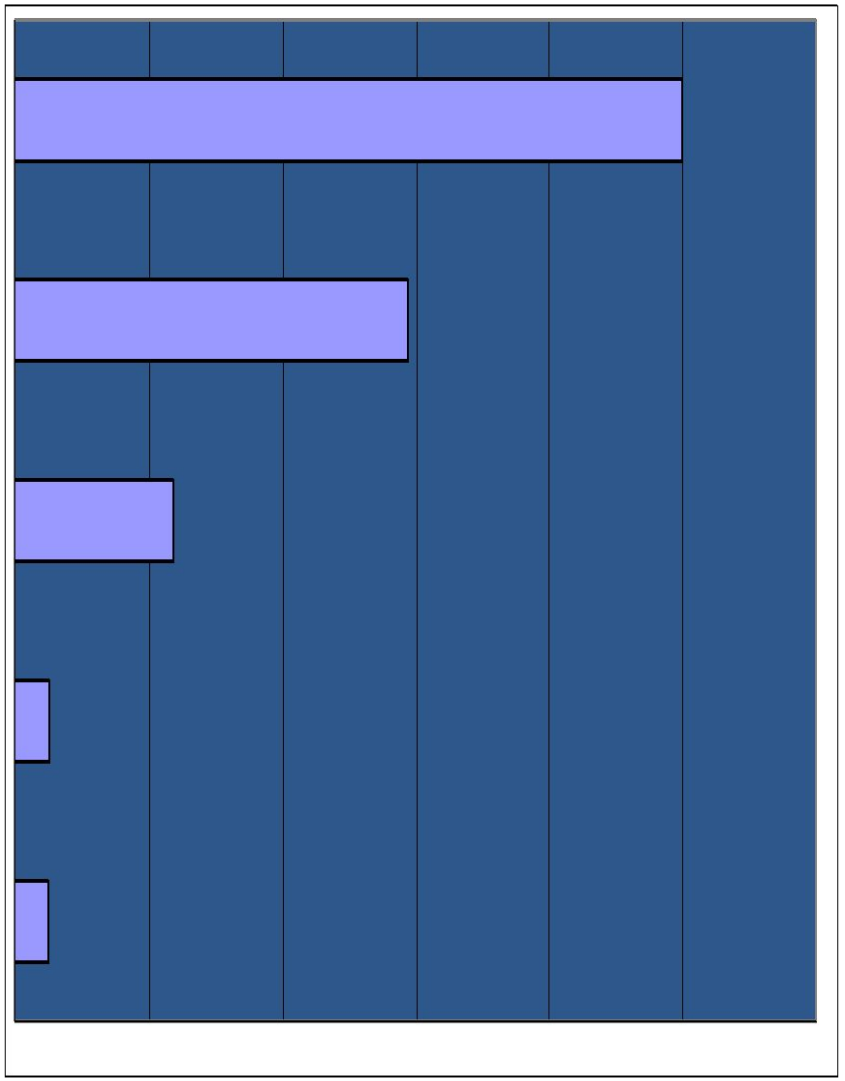
Проект CTPP

- виртуальная машина
- библиотека функций
- компилятор
- коллектор результатов
- кэш байткода



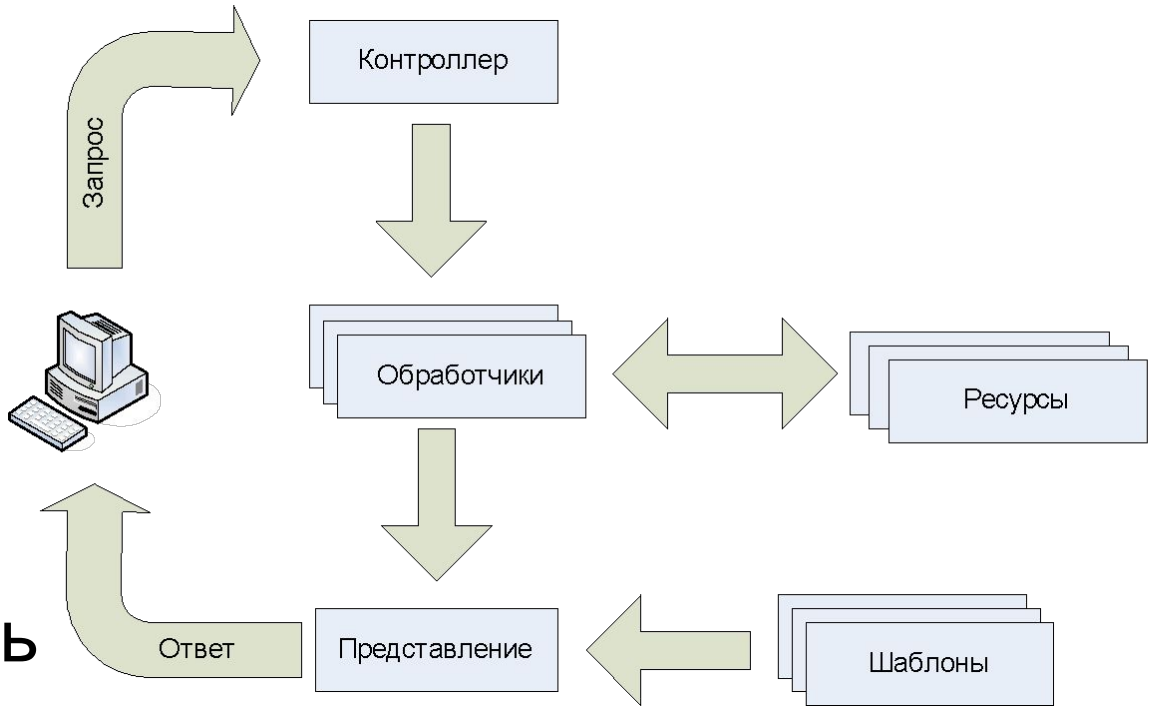
Результаты

- CTPP2
- HTML::Template::JIT
- HTML::Template::Pro
- HTML::Template
- Template::Toolkit



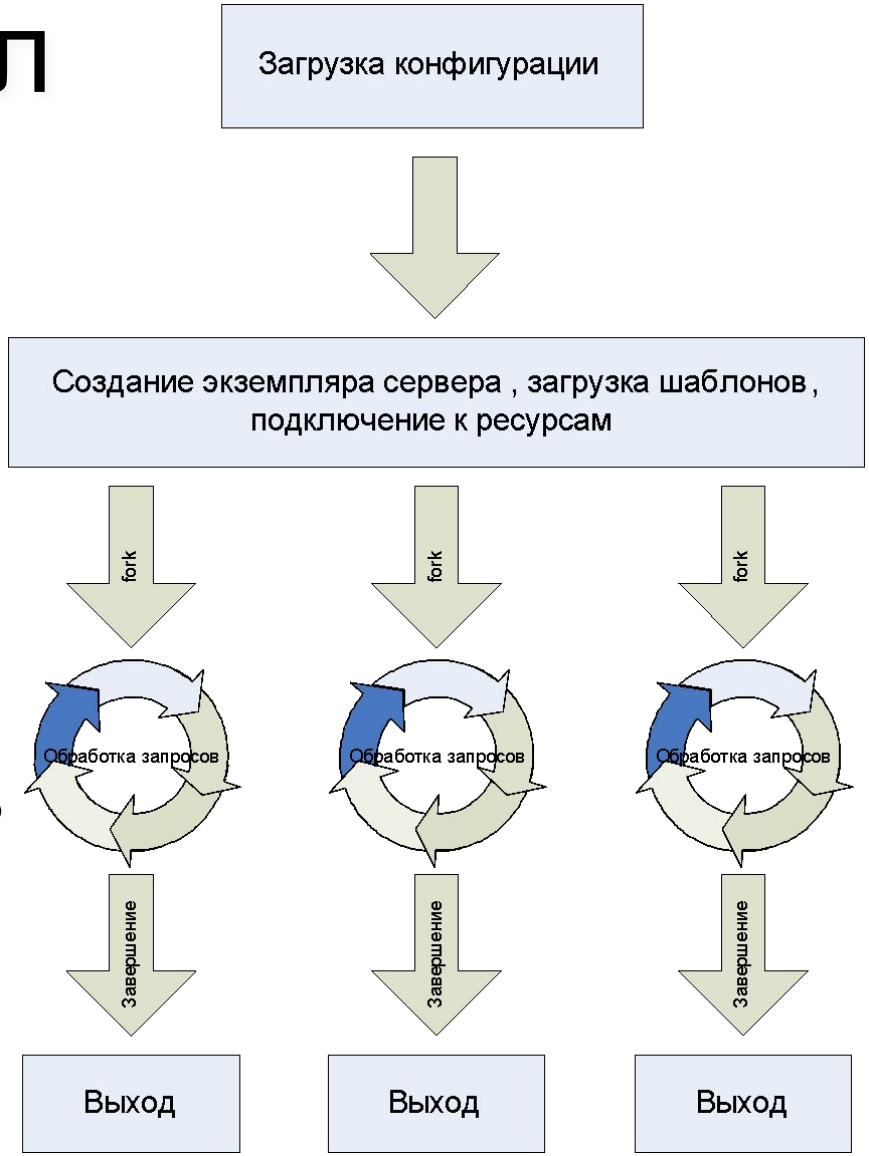
Архитектура CAS

- модульность
- изоляция сущностей
- простота разработки и сопровождения
- расширяемость
- универсальность
- интегрируемость



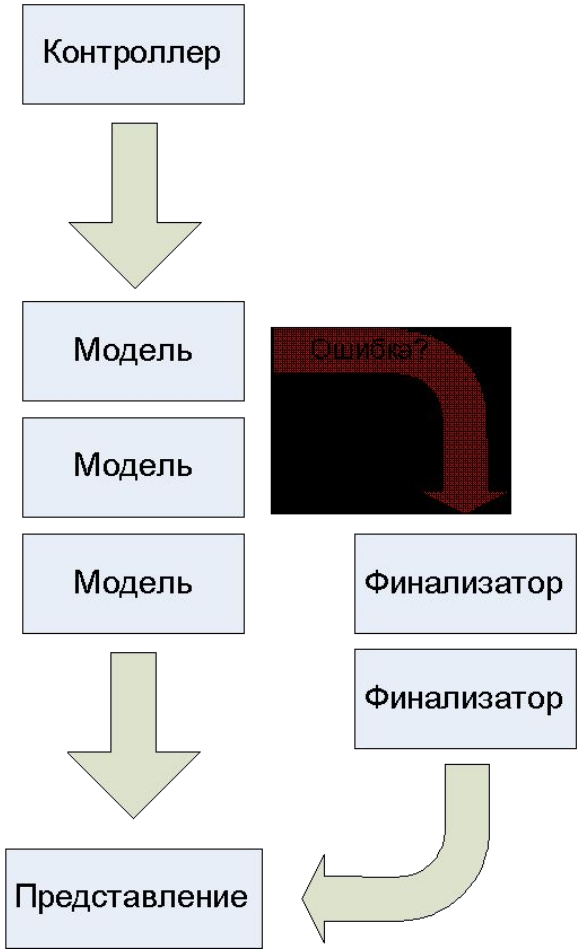
Жизненный цикл

- загрузка конфигурации
- загрузка модулей
- создание сервера приложений
- обработка запросов
- выход



Обработка запроса

- контролллер исполняется первым
- модели запускаются последовательно
- если возникла ошибка – работают финализаторы
- представление формирует данные



От слов – к делу!

- необходимый инструментарий
- пишем “Hello, World!”
- тестируем результаты работы
- пример посложнее – лента новостей
- сравним с mod_perl
- и с PHP – тоже сравним

Инструменты

- компилятор C++
- система сборки cmake
- сервер приложений C++
- 10 минут свободного времени

Hello, World!

- Создаем модуль

```
cas-xt -t handler -g -n Hello

Using templates from directory "/usr/share/cas/xt"
Output directory is "."
Creating [DIR]  Hello
Creating [DIR]  Hello/include
Creating [DIR]  Hello/src
Creating [FILE] Hello/src/Hello.cpp
Creating [FILE] Hello/CMakeLists.txt
```

Hello, World!

- Пишем код

```
INT_32 Hello::Handler(CTPP::CDT & oData, ASRequest &
oRequest, ASResponse & oResponse, ASLogger & oLogger)
{
    // Put your code here
    oData["hello"] = "Hello, World!";
    // 200 OK
    oResponse.SetHTTPCode(200);
    // Header
    oResponse.SetHeader("X-Module", "Hello");
return HANDLER_OK;
}
```

Hello, World!

- Создаем шаблон

```
<html>
<head>
    <title>My first example</title>
</head>
<body>
    <TMPL_var hello>
</body>
</html>
```

Hello, World!

- Проверяем результат

```
lynx -source http://localhost/hello.html

<html>
<head>
        <title>My first example</title>
</head>
<body>
        Hello, World!
</body>
</html>
```

Тоже самое – на mod_perl

```
package CAS::Hello;
use strict;
use Apache::Constants qw(:common);

my $T = new HTML::CTPP2();
my $B = $T -> parse_template('news.tmpl');

sub handler
{
    my $r = shift;
    $r -> content_type('text/html');
    $T -> param({hello => 'Hello, World!'});
    print $T -> output($B);
return OK;
}
1;
```

И на PHP

```php
<?php
  $T = new ctpp();

  $Bytecode = $T -> parse_template("newslist.ct2");

  $T -> emit_params(Array(newslist => $Result));

echo $T -> output($Bytecode);
?>
```

Пример посложнее

- Лента новостей

```
SQL::NonTransaction oNT =
  GetSQLConnector(oResponse,oGlobalPool).NewNonTransaction();


oData["newslist"] =
  NTSQLayerCDT(oNT, "SELECT * FROM news ORDER BY date")
                    << SQL::GetRowMap;
```

Интеграция

- единые шаблоны для всего проекта, независимо от “движка”
- простота миграции между технологиями
- высокая скорость прототипирования и разработки
- поддержка популярных языков и сред: Perl, PHP, Python

Интерфейсы

- Apache 1.3.X
- Apache 2.X
- FastCGI

Платформы и архитектуры

- Linux
- FreeBSD
- Solaris

- i386
- amd64
- UltraSPARC

Развитие проекта

- сервер-сборщик (привет, Mail.ru ;)
- несколько разных View: HTML, JSON, XML
- отдельные части сервера – в виде самостоятельных библиотек
- модули CAS для работы с разнообразными поставщиками данных