Оглавление

Назначение

Коротко — платформа для вебсайтов.

Система управления контентом и платформа для разработки веб сайтов.

Как система управления контентом предоставляет следующие возможности:

- управление перечнем путей ресурсов сайта
- управление содержимым ресурсов (страниц)

Как платформа для разработки предоставляет следующие возможности:

- исполнение кода javaScript на стороне сервера
- поддержка базы данных
- формирование xml ориентированного ответа
- формирование произвольного ответа

Общие концепции

Ядро системы представляет собой монолитное бинарное приложение операционной системы. После запуска начинает слушать заданный tcp – порт, принимает на нем входящие соединения, вычитывает из них http – запрос, формирует http ответ, отдает его и продолжает слушать свой порт. Возможно сопряжение с веб сервером на базе fast cgi.

В системе можно выделить следующие сущности

- статический ресурс
- точка
  - свойство
  - контроллер
  - фильтр
  - статик
- контекстные данные
- интерфейс к бд
- плагин

Статический ресурс представляет собой файл в выделенной папке (в дереве вложеных папок) статических ресурсов.

Точка является основным элементов для построения структуры сайта. Точка имеет набор именованых дочерних точек (дерево), из имен которых строятся пути, идентифицирующие эти точки.

Для внедрения контроля, в обработку запроса по заданному пути предлагаются транзитные и собственные фильтры. Транзитные фильтры стабатывают для всех точек, касающихся запрошенного пути. Собственные фильтры срабатывают для точек, на которых запрошенный путь заканчивается. Фильтр может иметь 3 функции* для инициализации, деинициализации и обработки проходящего запроса.

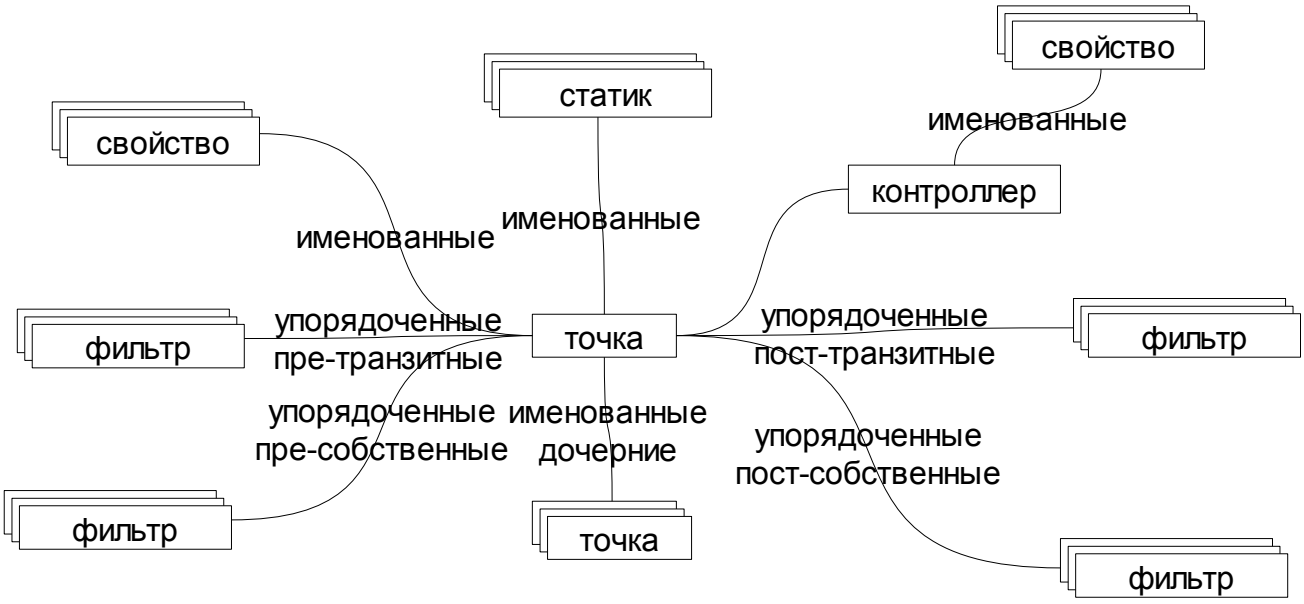
Для непосредственной обработки запроса и генерации ответа предлагаются контроллеры. Один контроллер принадлежит одной точке. Контроллер может иметь 4 функции* для инициализации, деинициализации, обработки запроса и генерации ответа.

Для настройки обработки запроса предлагаются именованные свойства точки и контроллера. Свойство может иметь постоянное значение либо герерировать свой значение функцией*.

Точка может иметь набор собственных именованых статических ресурсов.

* функция на языке JavaScript (TraceMonkey), доступна к редактированию разработчиком сайта.

Образно, точку и другие связанные с ней сущности, можно отобразить так:



Сценарий отработки запроса следующий (упрощенно):

1. ищется статический ресурс по запрошенному пути, если найден то отдается он, конец.
2. запрошенный путь проецируется на дерево точек
    1. выявляется последняя точка в пути
        1. если точка не найдена то 404, конец
        2. если у точки есть статик, соответствующий пути то отдается он, конец.
    2. собираются перечни пре и пост транзитных фильтров, точки которых коснулись пути
    3. собираются перечни пре и пост собственных фильтров, точка которых является последней в пути

3. последовательно отрабатываются пре фильтры транзитные и собственные
4. отрабатывается контроллер последней точки
    1. исполнение функции обработки
        1. исполнение функции отображения
5. последовательно отрабатываются пост фильтры собственные и транзитные
6. конец

Контекстные данные предоставляют возможность создавать некоторые контексты (например контекст «сессия агента» или контекст «сессия пользователя») и размещать в них некоторые данные. Данные для контекстов могут быть привязаны к точке, фильтру, контроллеру а так же, к некоторому глобальному месту. Таким образом, блок контекстных данных идентифицируется парой «место привязки»-«контекст».

Интерфейс к базе данных предоставляет базовые возможности работы с бд: подключить отключить бд, выполнить SQL запрос к бд.

Для унификации типовых решений используется подсистема плагинов. Плагин есть набор функционала и ресурсов. Плагин может

- инсталироваться/деинсталироваться из системы,
- внедрять в систему собственные поддеревья точек
- запускать и останавливать экземпляры своего функционала и предоставлять глобальный доступ к этим экземплярам

Роутер

Роутер исполняет следующие функции:

1. сопряжение с транспортом
2. формирование объекта запроса
3. хранение контекстных данных
4. создание экземпляров
    1. контроллера
    2. фильтра
5. формирование экземпляров из конфига
    1. точки
    2. контроллера
    3. свойства
    4. фильтра
    5. статика
6. хранение корневой точки

7. выдача в ответ запроса статического ресурса
8. обработка запроса на точку
9. поддержка крона
10. поддержка плагинов

Сопряжение с транспортом

Роутер в сторону транспорта предоставляет интерфейс многопоточного обработчика запросов. Многопоточность необходима для корректной обработки запроса по медленному каналу связи с агентом. Многопоточность не допустима при исполнении JavaScript кода в силу различных причин, вдаваться не стану.

Транспорт создает пул нитей (то поток то нить.. Далее будет «поток ввода вывода» поэтому тут нить) исполнения, регистрирует в роутере каждую из них. При наличии входящего соединения транспорт вычитывает заголовок http запроса и преобразует его в набор переменных окружения запроса. Этот набор окружения вместе с потоками ввода и вывода отдает роутеру на обработку в контексте одной нити из пула. Роутер создает объект запроса (который разбирает переменные GET, POST, куки). Роутер производит выдачу статического ресурса если он соответствует пути запроса. Если ресурс по пути запроса не обнаружен среди статических ресурсов то роутер блокирует мутекс, передает запрос в подходящую точку, разблокирует мутекс и отправляет ответ.

Таким образом, вычитывание тела запроса и запись ответа в транспорт производится вне блокировок (параллельно), обработка запроса точкой — в блокировке (последовательно).

Объект роутера доступен как глобальная переменная с именем «router».

Объект запроса

Объект запроса форимруется роутером при старте очередного запроса от транспорта. Несет информацию о запросе агента и предоставляет часть средств для формирвания ответа, доступен как глобальная переменная под именем «request».

Содержит следующие свойства

- env
  ассоциативная карта http заголовков запроса, а так же следующие переменные
  - SERVER_SOFTWARE
  - GATEWAY_INTERFACE
  - SERVER_ADDR
  - SERVER_PORT
  - REQUEST_METHOD
  - SERVER_PROTOCOL
  - REQUEST_URI
  - PATH_INFO

    - REMOTE_ADDR
    - REMOTE_PORT
    - SERVER_NAME
    - CONTENT_TYPE
    - CONTENT_LENGTH
    - HTTP_ACCEPT
    - HTTP_USER_AGENT
    - HTTP_REFERER
    - HTTP_COOKIE
- cookies
  ассоциативная карта куков агента
- paramsGet
  ассоциативная карта параметров GET запроса
- paramsPost
  ассоциативная карта параметров POST запроса, кроме файлов
- filesPost
  ассоциативная карта файлов POST запроса, один файл описывается следующими полями
    - nameClient – имя переданное агентом
    - nameServer — путь в файловой системе сервера, по которому сохранен файл
    - size — размер в байтах
    - contentType – mime тип, заявленный агентом
- params
  собранные вместе параметры GET и POST
- path
  запрошенный путь ресурса
- pathTail
  необработанный остаток пути

Содержит следующие методы

- setStatusCode(code)
  устанавливает статус код http ответа агенту клиента
- setHeader(name, value)
  устанавливает именованый параметр заголовка http ответа агенту клиента
- pushHeader(name, value)
  добавляет очередное значение именованому параметру заголовка http ответа агенту клиента

Тело http ответа формируется многократными вызовами глобальной функции ”print”. Тело ответа накапливается во время обработки запроса и отправляется агенту вместе с заголовками лишь после окончания обработки запроса.

Контекстные данные

Роутер держит менеджер контекстных данных, в сторону разработчика предоставляет 4 метода

- cdStartInstance(key)
  запуск экземпляра контекстных данных. Этот экземпляр представляет собой некий ассоциативный контейнер значний JavaScript, ключами которого являются некие идентификаторы мест. Что это за идентификаторы — в принципе не важно, они используются только менеджером контекстных данных для того чтобы сопоставлять каждое значение своему месту.
- cdStopInstance(key)
  останов
- cdGetInstance(key)
  получение слепка данных заданного экземпляра. Функционал разработчика может вызывать этот метод для того чтобы взять все данные заданного контекста. Предполагается что функционал разработчика не станет менять внутренность этих данных а лишь сохранит их с целью в последствии восстановить контекст вызовом cdSetInstance.
- cdSetInstance(key, value)
  установка слепка данных заданного экземпляра, ранее полученного вызовом cdGetInstance.

Сами контекстные данные располагаются в следующих объектах системы:

- в самом роутере
- в каждой точке
- в каждом контроллере
- в каждом фильтре

назову эти объекты «местами расположения контекстных данных».

Контекстные данные доступны для чтения и записи, доступ к ним осуществляется через выделенный атрибут места с именем «cd» (от Context Data). Например

router.cd.global=220;

тут router – место, global — ключ контекста (чуть ниже поясню кто такой global). Этот код установит значение 220 для контекстных данных экземпляра «global» в месте router. Устанавливаемое значение может быть любым значчением JavaScript, например

router.cd.global=”220”;

router.cd.global={a:220, b:[0,1,15],c:”220”};

router.cd.global=new Date();

Роутер при старте создает экземпляр контекстных данных по имени «global». Содержимое этого экземпляра доступно для чтения и записи на протяжении всего времени работы роутера и может хранить данные между запросами.

В качестве примера, на базе контекстных данных, фильтров и куков агента организована поддержка сессий агента. Выполнено это следующим образом

Создается пре-транзитный фильтр с именем, например, session. Он навешивается на корневую точку структуры. Таким образом, этот фильтр будет срабатывать перед обработкой любого пути на точках. Этому фильтру задаются функции инициализации и обработки. Аналогично создается пост-транзитный фильтр с функциями обработки и деинициализации. Таким образом имеем 4 функции

1. инициализирующая функция
   вызывается при старте структуры
   создает экземпляр контекстных данных с именем, например, session
   router.cdStartInstance('session')
   - функция пре-обработки запроса, срабатывает перед каждым запросом
     смотрит куки, если в них есть ключ 'session' – то берет ранее сохраненный слепок контекстных данных вызовом *router.cdGetInstance('session')* и устанавливает этот слепок вызовом router.cdSetInstance('session', data). Где хранит — в принципе не важно, для примера, пусть хранит в global экземпляре тех же контекстных данных.
2. функция пост-обработки, срабатывает после окончания обработки каждого запроса
   устанавливает куку 'session' дабы в следующий раз распознать начатую сессию,
   получает экземпляр вызовом *router.cdGetInstance('session')* и сохраняет его где нибудь
3. деинициализирующая функция,
   вызывается при останове структуры
   подчищает за собой, вызывает *router.cdStopInstance('session')*

Таким образом, функционал разработчика получает вполне себе нормальную поддержку сессий.

Еще пример, как использовать эти феньки

Пусть есть 2 точки структуры point1 и point2. Разработчик может использовать следующий код для подсчета обращений

```
////////////////////////////////////////////////
//... гдето в функции обработки запроса контроллера точки point1
if(!this.cd.session)
        this.cd.session = 0;
this.cd.session++;
//...
////////////////////////////////////////////////


////////////////////////////////////////////////
//... гдето в функции генерации ответа контроллера точки point2
if(!this.cd.session)
        this.cd.session = {cnt:0};
this.cd.session.cnt++;
var point1 = router.getPoint('/path/to/point1');
```

print(“на контроллер точки 1 было обращений: ”, point1.cd.session, “ а контроллер точки 2 выводил ответ ”, this.cd.session.cnt, “ раз”);

//...

//

Структура

Предназначена для упорядочивания точек ресурсов. Упорядочивает точки в двух разрезах

1. общее дерево родительский — дочерний, каждый узел которого маркируется частью пути, таким образом на точки накладывается путь из урла
2. точка может быть унаследована от другой точки (единожды и не допуская циклов). Наследуются фильтры, контроллер, статики, свойства, дочерние точки.

Структура начинается с некоторой корневой точки, доступной по пути '/'.

Точка

Свойство точки

Контроллер

Свойство контроллера

Фильтр

Статический ресурс

Контекстные данные

База данных

Крон

Плагины