1 APR::Brigade - Perl API for manipulating APR Bucket Brigades
1.1 Synopsis

use APR::Brigade();

$bb = APR::Brigade->new($r->pool, $c->bucket_alloc);
$ba = $bb->bucket_alloc();
$pool = $bb->pool;

$bb->insert_head($b);
$bb->insert_tail($b);

$b_first = $bb->first;
$b_last = $bb->last;

$b_prev = $bb->prev($b_last);
$b_next = $bb->next($b);

$bb2 = APR::Brigade->new($r->pool, $c->bucket_alloc);
$bb1->concat($bb2);

$len = $bb->flatten($data);
$len = $bb2->flatten($data, $wanted);

$len = $bb->length;
$bb3 = $bb->split($b_last);

last if $bb->is_empty();
$bb->cleanup();
$bb->destroy();

1.2 Description

APR::Brigade allows you to create, manipulate and delete APR bucket brigades.

1.3 API

APR::Brigade provides the following functions and/or methods:

1.3.1 cleanup

Empty out an entire bucket brigade:

$bb->cleanup;

- obj: $bb (APR::Brigade object)
  - The brigade to cleanup

- ret: no return value
- since: 2.0.00
This method destroys all of the buckets within the bucket brigade’s bucket list. This is similar to `destroy()` except that it does not deregister the brigade’s `pool()` cleanup function.

Generally, you should use `destroy()` This function can be useful in situations where you have a single brigade that you wish to reuse many times by destroying all of the buckets in the brigade and putting new buckets into it later.

### 1.3.2 `concat`

Concatenate brigade `$bb2` onto the end of brigade `$bb1`, leaving brigade `$bb2` empty:

```
$bb1->concat($bb2);
```

- **obj**: `$bb1` (**APR::Brigade object**)
  The brigade to concatenate to.
- **arg1**: `$bb2` (**APR::Brigade object**)
  The brigade to concatenate and empty afterwards.

- **ret**: no return value
- **since**: 2.0.00

### 1.3.3 `destroy`

Destroy an entire bucket brigade, includes all of the buckets within the bucket brigade’s bucket list.

```
$bb->destroy();
```

- **obj**: `$bb` (**APR::Brigade object**)
  The bucket brigade to destroy.

- **ret**: no return value
- **excpt**: `APR::Error`
- **since**: 2.0.00

### 1.3.4 `is_empty`

Test whether the bucket brigade is empty

```
$ret = $bb->is_empty();
```

- **obj**: `$bb` (**APR::Brigade object**)
- **ret**: `$ret` (boolean)
- **since**: 2.0.00
1.3.5 first

Return the first bucket in a brigade

```perl
$b_first = $bb->first;
```

- **obj**: `$bb` ([APR::Brigade object](#))
- **ret**: `$b_first` ([APR::Bucket object](#))

The first bucket in the bucket brigade `$bb`.

`undef` is returned if there are no buckets in `$bb`.

- **since**: 2.0.00

1.3.6 flatten

Get the data from buckets in the bucket brigade as one string

```perl
$len = $bb->flatten($buffer);
$len = $bb->flatten($buffer, $wanted);
```

- **obj**: `$bb` ([APR::Brigade object](#))
- **arg1**: `$buffer` (SCALAR)
  
  The buffer to fill. All previous data will be lost.

- **opt arg2**: `$wanted` (number)
  
  If no argument is passed then all data will be returned. If `$wanted` is specified -- that number or less bytes will be returned.

- **ret**: `$len` (number)
  
  How many bytes were actually read.

  `$buffer` gets populated with the string that is read. It will contain an empty string if there was nothing to read.

- **since**: 2.0.00
- **except**: [APR::Error](#)

1.3.7 insert_head

Insert a list of buckets at the front of a brigade
$bb->insert_head($b);

- **obj**: $bb (APR::Brigade object)
  Brigade to insert into

- **arg1**: $b (APR::Bucket object)
  the bucket to insert. More buckets could be attached to that bucket.

- **ret**: no return value
- **since**: 2.0.00

### 1.3.8 `insert_tail`

Insert a list of buckets at the end of a brigade

$bb->insert_tail($b);

- **obj**: $bb (APR::Brigade object)
  Brigade to insert into

- **arg1**: $b (APR::Bucket object)
  the bucket to insert. More buckets could be attached to that bucket.

- **ret**: no return value
- **since**: 2.0.00

### 1.3.9 `last`

Return the last bucket in the brigade

$b_last = $bb->last;

- **obj**: $bb (APR::Brigade object)
- **ret**: $b_last (APR::Bucket object)
  The last bucket in the bucket brigade $bb.

  *undef* is returned if there are no buckets in $bb.

- **since**: 2.0.00
1.3.10 length

Return the total length of the data in the brigade (not the number of buckets)

\[ \text{$len = $bb->length;} \]

- obj: $bb (APR::Brigade object)
- ret: $len (number)
- since: 2.0.00

1.3.11 new

my $nbb = APR::Brigade->new($p, $bucket_alloc);
my $nbb = $bb->new($p, $bucket_alloc);

- obj: $bb (APR::Brigade object or class)
- arg1: $p (APR::Pool object)
- arg2: $bucket_alloc (APR::BucketAlloc object)
- ref: $nbb (APR::Brigade object)

a newly created bucket brigade object

- since: 2.0.00

Example:

Create a new bucket brigade, using the request object’s pool:

use Apache2::Connection ();
use Apache2::RequestRec ();
use APR::Brigade ();
my $bb = APR::Brigade->new($r->pool, $r->connection->bucket_alloc);

1.3.12 bucket_alloc

Get the bucket allocator associated with this brigade.

my $ba = $bb->bucket_alloc();

- obj: $bb (APR::Brigade object or class)
- ref: $ba (APR::BucketAlloc object)
- since: 2.0.00

1.3.13 next

Return the next bucket in a brigade
$b\_next = $bb->next($b);

- **obj:** $bb (APR::Brigade object)
- **arg1:** $b (APR::Bucket object)

The bucket after which the next bucket $b\_next is located

- **ret:** $b\_next (APR::Bucket object)

The next bucket after bucket $b.

**undef** is returned if there is no next bucket (i.e. $b is the last bucket).

- **since:** 2.0.00

### 1.3.14 pool

The pool the brigade is associated with.

$pool = $bb->pool;

- **obj:** $bb (APR::Brigade object)
- **ret:** $pool (APR::Pool object)
- **since:** 2.0.00

The data is not allocated out of the pool, but a cleanup is registered with this pool. If the brigade is destroyed by some mechanism other than pool destruction, the destroying function is responsible for killing the registered cleanup.

### 1.3.15 prev

Return the previous bucket in the brigade

$b\_prev = $bb->prev($b);

- **obj:** $bb (APR::Brigade object)
- **arg1:** $b (APR::Bucket object)

The bucket located after bucket $b\_prev

- **ret:** $b\_prev (APR::Bucket object)

The bucket located before bucket $b.

**undef** is returned if there is no previous bucket (i.e. $b is the first bucket).

- **since:** 2.0.00
1.3.16 split

Split a bucket brigade into two, such that the given bucket is the first in the new bucket brigade.

```perl
$bb2 = $bb->split($b);
```

- **obj:** $bb (**APR::Brigade** object)
  
The brigade to split

- **arg1:** $b (**APR::Bucket** object)
  
The first bucket of the new brigade

- **ret:** $bb2 (**APR::Brigade** object)
  
The new brigade.

- **since:** 2.0.00

This function is useful when a filter wants to pass only the initial part of a brigade to the next filter.

Example:

Create a bucket brigade with three buckets, and split it into two brigade such that the second brigade will have the last two buckets.

```perl
my $bb1 = APR::Brigade->new($r->pool, $c->bucket_alloc);
my $ba  = $c->bucket_alloc();
$bb1->insert_tail(APR::Bucket->new($ba, "1"));
$bb1->insert_tail(APR::Bucket->new($ba, "2"));
$bb1->insert_tail(APR::Bucket->new($ba, "3"));
```

$bb1 now contains buckets "1", "2", "3". Now do the split at the second bucket:

```perl
my $b = $bb1->first;  # 1
$b = $bb1->next($b);  # 2
my $bb2 = $bb1->split($b);
```

Now $bb1 contains bucket "1". $bb2 contains buckets: "2", "3"

1.4 See Also

mod_perl 2.0 documentation.
1.5 Copyright

mod_perl 2.0 and its core modules are copyrighted under The Apache Software License, Version 2.0.

1.6 Authors

The mod_perl development team and numerous contributors.
Table of Contents:

1 APR::Brigade - Perl API for manipulating APR Bucket Brigades ........................................ 1
  1.1 Synopsis ............................................................................................................................ 2
  1.2 Description .......................................................................................................................... 2
  1.3 API ....................................................................................................................................... 2
    1.3.1 cleanup .......................................................................................................................... 2
    1.3.2 concat ............................................................................................................................ 3
    1.3.3 destroy .......................................................................................................................... 3
    1.3.4 is_empty ......................................................................................................................... 3
    1.3.5 first .................................................................................................................................. 4
    1.3.6 flatten ............................................................................................................................ 4
    1.3.7 insert_head ..................................................................................................................... 4
    1.3.8 insert_tail ....................................................................................................................... 5
    1.3.9 last ................................................................................................................................... 5
    1.3.10 length ........................................................................................................................... 6
    1.3.11 new .................................................................................................................................. 6
    1.3.12 bucket_alloc .................................................................................................................. 6
    1.3.13 next .................................................................................................................................. 6
    1.3.14 pool .................................................................................................................................. 7
    1.3.15 prev .................................................................................................................................. 7
    1.3.16 split ................................................................................................................................ 8
  1.4 See Also ................................................................................................................................. 8
  1.5 Copyright ............................................................................................................................... 9
  1.6 Authors .................................................................................................................................. 9

15 Feb 2014