1 Apache::RequestRec -- A Perl API for Apache request object

1.1 SYNOPSIS

```
use Apache::RequestRec;
sub handler{
    my $r = shift;

    my $s = $r->server;
    my $dir_config = $r->dir_config;
    ...
}
```

1.2 DESCRIPTION

`Apache::RequestRec` provides the Perl API for Apache request object.

1.3 API

Function arguments (if any) and return values are shown in the function's synopsis.

1.3.1 server()

```
$s = $r->server;
```

Gets the `Apache::Server` object for the server the request `$r` is running under.

1.3.2 dir_config()

dir_config() provides an interface for the per-directory variable specified by the `PerlSetVar` and `PerlAddVar` directives, and also can be manipulated via the `APR::Table` methods.

The keys are case-insensitive.

```
$apr_table = $r->dir_config();
```

dir_config() called in a scalar context without the `$key` argument returns a *HASH* reference blessed into the `APR::Table` class. This object can be manipulated via the `APR::Table` methods. For available methods see the `APR::Table` manpage.

```
@values = $r->dir_config($key);
```

If the `$key` argument is passed in the list context a list of all matching values will be returned. This method is ineffective for big tables, as it does a linear search of the table. Thefore avoid using this way of calling dir_config() unless you know that there could be more than one value for the wanted key and all the values are wanted.

```
$value = $r->dir_config($key);
```

If the `$key` argument is passed in the scalar context only a single value will be returned. Since the table preserves the insertion order, if there is more than one value for the same key, the oldest value assosiated with the desired key is returned. Calling in the scalar context is also much faster, as it'll stop searching the table as soon as the first match happens.

```
$r->dir_config($key => $val);
```

If the `$key` and the `$val` arguments are used, the set() operation will happen: all existing values associated with the key `$key` (and the key itself) will be deleted and `$value` will be placed instead.

```
$r->dir_config($key => undef);
```

If `$val` is *undef* the unset() operation will happen: all existing values associated with the key `$key` (and the key itself) will be deleted.

1.3.3 ap_auth_type()

`$r->ap_auth_type` gets or sets the *ap_auth_type* slot in the request record.

```
$r->ap_auth_type('Basic');
```

or

```
my $auth_type = $r->ap_auth_type;
```

ap_auth_type holds the authentication type that has been negotiated between the client and server during the actual request. Generally, *ap_auth_type* is populated automatically when you call `$r->get_basic_auth_pw` so you don't really need to worry too much about it, but if you want to roll your own authentication mechanism then you will have to populate *ap_auth_type* yourself.

Note that `$r->ap_auth_type` was `$r->connection->auth_type` in the mod_perl 1.0 API.

1.3.4 main()

```
$main_r = $r->main;
```

If the current request is a sub-request, this method returns a blessed reference to the main request structure. If the current request is the main request, then this method returns undef.

To figure out whether you are inside a main request or a sub-request/internal redirect, use `$r->is_initial_req`.

Table of Contents: