mod_spin Reference Manual

1.0.10

Generated by Doxygen 1.4.7

Mon Sep 4 21:18:50 2006

Contents

1 mod_spin Main Page

Introduction

See also:

Installation News Licence

```
Welcome to mod_spin for Apache 2.x from Rexursive
=================================================

mod_spin is an Apache module that provides the following functionality (in
conjunction with some other modules):

- a simple template language with data replacement capabilities only
- persistent application and session data tracking
- dynamic linking of applications into Apache 2 as shared libraries
- parameters, cookies and multipart/form-data parsing via libapreq2
- simple API for (kind of) MVC controller functionality
- simple API for pooled (or not) access to SQL databases

mod_spin is written in C, just like Apache 2 itself and it uses APR, which
was written to be cross platform, secure and perform well. Generally speaking,
you should see speed improvements when compared to Java, PHP and Perl
solutions, sometimes even by an order of magnitude.

This software exists to enable easy development and deployment of high
performance web applications written in C (or perhaps even other languages) on
Linux (or Unix) systems running Apache 2. It should be particularly easy to do
that on the systems that run RPM packaging system, such as Fedora Core, Red
Hat Enterprise Linux, CentOS and similar distributions. Obviously, other types
of packages can be built too, but RPM support is already in mod_spin.

How does mod_spin work?
=======================

mod_spin is essentially a content handler, meaning, for a specified file
extension(s), mod_spin will read the file, parse it into an Abstract Syntax
```

Tree (AST) and then replace the occurrences of references with values coming from the application (this is where mod_spin is similar to Velocity). There is no predefined file extension for mod_spin templates. I sometimes use ".sm" for "spin macro", but you can use whatever you like, as long as you tell Apache what that is.

At one point I was considering mod_spin as an output filter, but eventually couldn't find enough justification to do so. The code would be much more complex and I wanted to keep things simple. One day, maybe...

The application, a shared library (or .so), is dynamically linked at run-time (i.e. when the request is handled by Apache) and its entry function is called. This function takes one argument, which is a structure containing the context (where the data to be replaced is stored), parsed parameters, session and application information and the current request. It then executes whatever code is appropriate for the current request, most likely based on the URI and the parameters (this completely depends on what the application actually does, of course). This execution results in data structures holding the values that are to be placed into the template. This data is then placed inside the template by traversing the AST and replacing references with values. The end result (a bucket brigade) is given to Apache output filters to push out into the world (and possibly modify the content as well).

Before the application entry function is called, mod_spin takes care of application and session tracking (it relies on cookies for that). It does that in a persistent manner (i.e. the values associated with the application and session are stored in a file). SDBM database functionality, already included in APR, is used to provide hashed searches for values based on specified keys.

What are mod_spin applications?
==========

They are simply shared libraries. You would normally get those as a result of writing, compiling and linking a set C program files. You can, of course, get those as a result of compiling and linking some other language. Keep in mind that mod_spin expects its data in a particular way. IF THAT'S NOT FOLLOWED, MANY THINGS WILL BREAK AND YOU MIGHT CAUSE SECURITY PROBLEMS ON YOUR SYSTEM.

That being said, mod_spin applications are probably not for someone that isn't comfortable with application development in C. If you're looking for a scripting language, mod_spin isn't it. Actually, one of the main reasons for writing mod_spin was that I wanted full access to C Unix API but without the need to hammer (X)HTML out of my code. With mod_spin you can keep your focus on business logic and forget presentation for the most part.

See section "Service function" for all the details related to the entry point into the application.

Why not CSP (C Server Pages)?
==========

C Server Pages are an implementation similar to JSP (Java Server Pages), but unlike JSP, feature C language snippets, not Java, placed into HTML. Such page is then converted into a C program, compiled and then linked into a shared library, which is dynamically linked into Apache at run-time. In essence, it taps directly into the C run-time system, just like mod_spin does. It is probably faster because it does no template processing.

However, just like JSP, it suffers from similar problems. The first one is a confusing mix of C programming language with HTML. This makes it completely unusable (on the presentation level) by non-experts, even if trivial changes to the page are required (i.e. a spelling fix can cause serious functionality problems, even security violations), not to mention that the mixed code is truly unreadable. The second one is the "translate -> compile -> link -> dynamic link -> run" process, which creates further complications and opens up the possibilities for strange run-time errors. And finally, a full C development environment has to be installed on the system running CSP, in case

any of the pages ever get changed. These arguments are more or less the same as the one when a template language such as Velocity is compared to JSP.

mod_spin avoids the above by defining a simple, data replacement only template language and leaves all of the programming logic where it belongs - in the application. At the same time, the application behaves mostly (but not completely) neutral as far as presentation of data is concerned. It is almost irrelevant what the output is going to look like, so most of the time programmers are only busy working on functionality, not looks.

The above arguments, however, don't cut it for everyone (as I have observed in my encounters with other developers), so if you're one of those, mod_spin is probably not for you.

Security concerns
==========

Just like anything else written in C, if you aren't careful, you can shoot yourself in the foot quite effectively. Buffer overflows and similar problems can, however, be avoided if problematic functions aren't used and good programming practices followed. mod_spin makes heavy use of APR, which is an example of an API that was designed from the ground up to be secure.

Even if you're most careful, security problems can happen. It is therefore good to follow guidelines for secure Apache setup. In extreme circumstances (e.g. when you're allowing others to deploy their own applications into Apache running mod_spin), IT IS ADVISABLE TO RUN A SEPARATE INSTANCE OF APACHE, BEHIND THE MAIN SERVER, WITH A SOLE PURPOSE OF RUNNING MOD_SPIN APPLICATIONS. Virtual hosting for multiple clients is one of the examples where such a scenario might be effective. Applying chroot jail, SELinux and/or running different Apache instances under different user IDs on unprivileged ports will go a long way toward ensuring that even if someone breaks in, the potential for damage is minimal.

Here are some very important security implications that may arise from the use of mod_spin. This is in relation to tracking of application and session data and to connection pools. To understand the issues involved, one needs to understand how Apache deals with multiple client connections. One also needs to understand Unix file permission model.

Apache will generally spin up numerous processes or threads in order to handle multiple connections from clients. There is no guarantee that a process/thread that handled one client's connection will handle it again in the future. The process/thread will be assigned at Apache's discretion. So, it is possible, even likely, that a process that handled something related to one client, handles another client next time. If there is anything in the memory space of this process/thread left over from the previous client, it will be completely accessible to the next client. Applications of mod_spin, shared libraries, are linked directly into the running process and they have full access to the memory space of that process.

The above means that an application can fetch any previously opened connection from the pool of connections and use it at will. Depending on how this connection was opened in the first place (by the original client), this will enable reading and/or writing of data that otherwise might not be accessible. IT IS CLEAR THAT THIS IS A SERIOUS SECURITY IMPLICATION.

Generally speaking, you should make sure that mod_spin applications linked into an instance of Apache are all from the same "security realm". For instance, if you're using mod_spin to enable dynamic applications virtually hosted on a single server (machine) for your customers, allowing two different customers to deploy mod_spin application into the same instance of Apache will allow them to read/write each other's databases. This might be accidental or, more seriously, intentional and malicious. YOU SHOULD ABSOLUTELY MAKE SURE THAT SUCH APPLICATIONS ARE DEPLOYED INTO DIFFERENT INSTANCES OF APACHE, RUNNING UNDER DIFFERENT USER ACCOUNTS!

The second problem is connected to this in terms of user accounts and access to files. Session and application data tracking files will be readable and writable by the user Apache processes run as (this is defined in the Apache configuration file). So, in the above scenario with two customers, they would be able to read/write each other's session/application files. Once again, YOU SHOULD ABSOLUTELY MAKE SURE THAT SUCH APPLICATIONS ARE DEPLOYED INTO DIFFERENT INSTANCES OF APACHE, RUNNING UNDER DIFFERENT USER ACCOUNTS!

General recommendation is: if you don't have full control over all applications and you're using session/application persistent store and/or connection pools, you should have separate instances of Apache for each identifiable "security realm".

Stability
=========

If you ever wrote a C program, you know that one of the most dreadful things is the infamous Segmentation Fault (SIGSEGV, Signal 11). It happens when your program tries to dereference a memory location that is invalid, such as NULL. mod_spin makes reasonable effort to ensure the raw data it handles (the template, session and application data, parameters, cookies etc.) is processed in a manner that produces no segfaults. As for the context, the data that your own application prepares, mod_spin doesn't have any control over what's in there. It will take certain precautions against obvious stuff like NULL pointers, but some of the other errors might be complicated to detect and handle. And because mod_spin is a small and lightweight piece of software, it doesn't do any of that. It simply relies on you (yes, that's YOU!) that the data placed in the context is going to be good.

If the data is not good, the code will segfault, bringing down with it the child Apache process inside which it was executing. This is not a big concern from Apache's point of view, as the parent process will fork as many new processes as it needs - however, your server might suffer a denial of service attack because of this. So, make your context data good!

Note that the above scenario is only applicable to the prefork Apache MPM. Other MPM modules might behave in a different way (i.e. more than one thread of Apache can be affected), so keep that in mind when deploying mod_spin under those scenarios.

Memory leaks
============

Apache Portable Runtime uses memory pools for most memory allocation and mod_spin naturally follows. It is a good and fast approach. However, some memory pools may have rather long life cycle (namely per-thread pool of mod_spin and its sub-pools, used for parsed templates). Although the code of mod_spin tries to avoid these long lasting pools whenever possible, it is sometimes unavoidable to have things put into them. Also, the connection pools will be associated with the per-thread pool. This can lead, over time and given huge number of requests or a lot of template changes, to small memory leaks.

That's why mod_spin as of version 0.9.4 has a new configuration parameter SpinClearCount. After defined number of requests handled by the thread, the per-thread pool is destroyed, including all its sub-pools and database connections. This causes templates to be re-parsed and database connections to be reopened, which is a performance penalty, but it might be useful in some pathological corner cases.

Language constructs
===================

The template language of mod_spin has only three commands: a loop and two conditionals. They look like this:

```
#for(${reference})
  some text within the loop and a ${reference.column}
```

```
#end

#if(${reference})
  some text to replace if ${reference} is not NULL
#else
  some text to replace if ${reference} is NULL
#end

#unless(${reference})
  some text to replace if ${reference} is NULL
#else
  some text to replace if ${reference} is not NULL
#end
```

That's it. Everything else is the matter for the application, not the template language.

References, which are case sensitive, placed inside the text will be replaced with their values from the context or nothing if that value is NULL or the reference does not exist. References are never recursively substituted (this may create denial of service or security problems and it is therefore avoided). If such functionality is desired, it belongs in your application.

Data types and loops

You can place two different types of data in the context: single and rows.

Singles are simply character strings. They are pointed to by a char* and limited by the length. Generally, mod_spin does not rely on '\0' being present at the end of the string. However, regular C APIs mostly handle strings that have the ending '\0' character. Therefore, all single data, although being declared as 'size' in length, actually gets a '\0' character at the end (naturally, the space for this character is allocated when the single is created). This is very useful when communicating with regular C APIs, as it saves a lot of copying and memory allocation. If you design your own functions that create single data, you MUST FOLLOW THIS CONVENTION OR YOU'RE SETTING YOURSELF UP FOR A WHOLE HEAP OF BUFFER OVERFLOWS!

Rows are data that looks a lot like something that would be returned from an SQL query: there are named columns and data contains certain number of rows. However, unlike what's returned by SQL queries (i.e. single pieces of data), each actual piece of data can again be either rows or single. This then enables nesting of multiple data dimensions. The nested #for loops are used to spin around such data. That's where the name mod_spin comes from.

Figure 1: Example data - Single

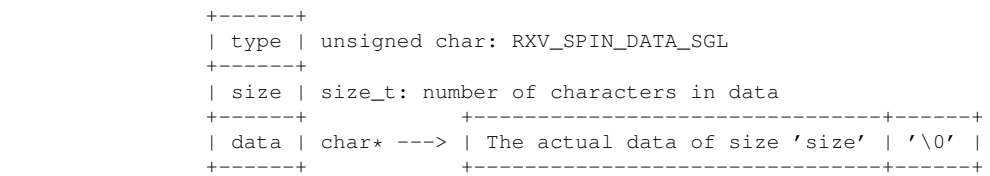
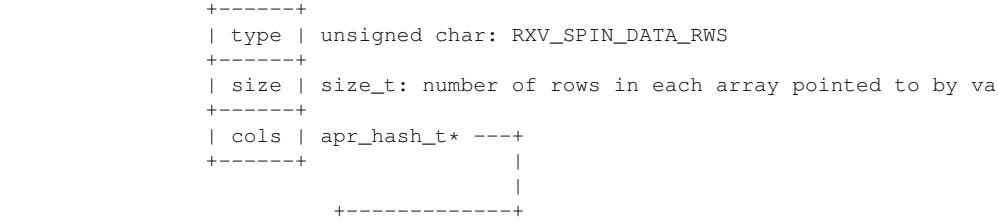
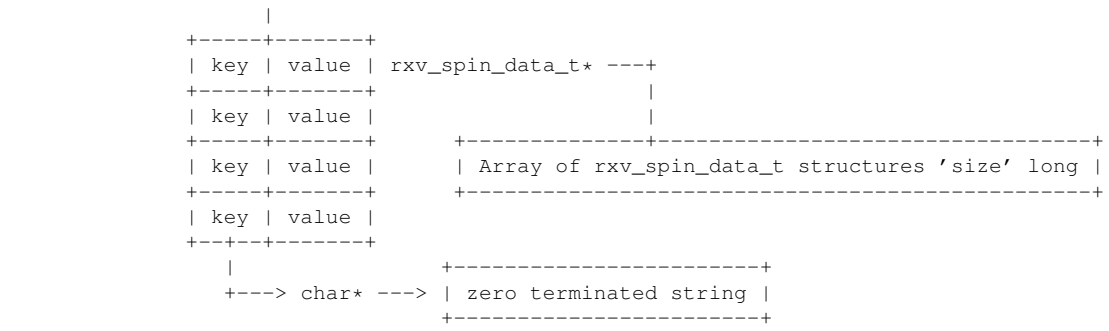


Figure 2: Example data - Rows





The final data type that is replaced into the template is always single. mod_spin doesn't know how to replace full rows because the presentation would be undefined. That's why you have to use #for loops to spin around rows data type to place the singles contained there in their correct places inside the template.

The #for loop won't spin if the data it is supposed to process is NULL. This can happen if the appropriate data for the reference cannot be found in the context, or if the value of it is NULL. The same applies to actual references that are replaced into the template - if the end result is NULL, nothing is replaced.

Metadata
========

Some of the API calls, like rxv_spin_meta_vstr() and rxv_spin_meta() return a pointer to rxv_spin_data_t that has a type of RXV_SPIN_DATA_MTA, or metadata. This data type is never used in the AST and if it is passed into it, it might cause errors. Its only purpose is to facilitate the API itself by making sure lengths of data arrays are stored somewhere, so that the programmer using the API doesn't have to use separate variables to store them. API calls know how to handle metadata, so if you stick to those, you should be fine.

IMPORTANT: Placing metadata into AST can have unpredictable results.

Conditionals
============

The only other command in mod_spin apart from #for loop is the conditional, as shown above. Again, it looks like this:

```
#if(${reference})
  something if ${reference} is not NULL
#else
  something if ${reference} is NULL
#end
```

or the negative variant:

```
#unless(${reference})
  something if ${reference} is NULL
#else
  something if ${reference} is not NULL
#end
```

You can also use:

```
#if(${ref}) something #end
#if(${ref}) something #else#end
#if(${ref})#else something #end
```

and naturally:

```
#unless(${ref}) something #end
#unless(${ref}) something #else#end
#unless(${ref})#else something #end

Loops and impossible references
===============================

Normally, mod_spin template will look something like this:

First text ${ref1}
#for(${ref2})
  replicate some other text and ${ref2.col1}
#end

The ${ref1} will be replaced with the value found in the context, if the data
it points to is a single, or nothing at all if the data it points to is of
type rows. The #for loop will spin around ${ref2} and replicate the enclosing
text for all instances of data that ${ref2} points to. The ${ref2.col1} will
be replaced with the current row value of the column "col1", if ${ref2}
happens to be a data type rows and ${ref2.col1} resolves to a data type
single for the current row.

Now let's examine an example where impossible references are used:

First text ${impossible.reference}
#for(${second.impossible.reference})
  replicate some other text and ${second.impossible.reference.column}
#end

The first reference ${impossible.reference}, can never be found in the context
because there is no #for loop to spin the data in ${impossible} in order to
find ${impossible.reference}. So, when creating the AST, mod_spin will simply
ignore this reference. The reference used to spin the #for loop,
${second.impossible.reference}, is also something that cannot exist, so
mod_spin will ignore the whole loop and never place any of it into AST.

Note that this is different from the first code snippet with, for instance, the
value of ${ref2} being NULL, or not existing at all. The parsed #for loop and
the text it encloses will be placed into the AST, but it won't be replicated
because there is no data to spin the loop around.

The above discussion applies to conditional statements as well. For instance:

First text ${impossible.reference}
#if(${second.impossible.reference})
  replicate some other text and ${second.impossible.reference.column}
#end

The above example would yield exactly the same output as the previous example
with the #for loop. However, if you use the #else, then whatever is placed
within it will be used. For instance:

First text ${impossible.reference}
#if(${second.impossible.reference})
  replicate some other text and ${second.impossible.reference.column}
#else
  this will always be in the output
#end

In the above example, the text placed between #else and #end will always be in
the output, because the #if would never be true, given that the reference is
impossible.

And one example for the #unless, the negative conditional:

#unless(${second.impossible.reference})
  replicate some other text and ${second.impossible.reference.column}
```

```
#end

The above will always end up in the output, because the reference is
impossible.

Service function
================

Service function is the entry function into your application. It is called
BEFORE template processing, so it has the potential to change which template
is going to be processed as well as to decline or do the processing
completely.

The entry function (by default called rxv_spin_service()) takes one argument -
the context. It returns an integer which is similar to what an Apache handler
would return. The meaning is as follows:

OK: Everything was OK, continue with template processing. Note here that by
manipulating filename field within the request_rec structure, you can change
which template is to be processed. Make sure other fields (e.g. finfo) that
are related to filename are properly updated as well.

REDIRECT (e.g. HTTP_TEMPORARY_REDIRECT): Any further processing should not be
done as this request is going to be externally redirected. Note here that the
application HAS TO set the "Location" header in headers_out. Failing that, the
client will have problems.

DECLINED or DONE: The service function either decided it's not something this
handler should do (DECLINED) or has done all the work on behalf of it (DONE).
These are short-circuit return codes that will greatly affect Apache request
processing, so be careful with them.

ANYTHING ELSE: This will result in an internal server error.

If SpinApplication isn't specified, the shared library will not be loaded at
all, but the template will be processed as normal. However, there will be no
data in the context and therefore none will be placed in the final output.

Loading of applications
=======================

Early versions of mod_spin (up to and including 0.9.12) had a very primitive
logic of loading applications (libraries, Dynamic Shared Objects, DSOs). On
each request, the library would be loaded and then unloaded. For small
libraries, this was almost acceptable, but for libraries that pulled in a lot
of dependencies (i.e. other libraries), the performance penalty was severe
(I've measured over 8 times performance degradation in some of the cases, but
it could be even worse). To avoid that, a new system had to be implemented in
0.9.13.

First, some background. All loading of shared libraries is done using
apr_dso_load() call from APR. On Linux (and some other Unix variants), this
translates into dlopen() call, which gives back a handle to a loaded library.
If a process attempts to open the same library again, the same handle will be
given back and a reference count for that library will be increased. In a
multi-threaded environment, this would mean that if multiple threads of
execution attempt to open the same library, they would be given back the same
handle and the reference count would be equal to the number of thread passes
that opened the library. Since Apache 2 could be running in a multi-threaded
configuration (e.g. worker MPM), it is very difficult to control when a library
will be completely unloaded. Something like that would involve introduction of
per-process read/write locks, the code would become much more complicated and
bug-prone. Instead, new version of mod_spin relinquishes the control of
unloading of libraries to Apache itself.

So, mod_spin 0.9.13 and above make sure that each thread keeps cache of loaded
DSOs and that it opens a particular library only once. This is done to avoid
```

registering of a pool cleanup for each call to apr_dso_load(), which would quickly grow private thread pool. Once the new applications are deployed, in order to reliably reload them, the main Apache process has to be given SIGUSR1 signal (i.e. a graceful restart has to be initiated), so that all child processes die and new ones replace them. This will ensure new applications are loaded across the board.

Note that if SpinClearCount other than zero is specified, the private thread pool will be cleaned after specified number of requests served by the thread. Each time the pool is cleaned, the apr_dso_unload() will be called through the pool cleanup functionality. However, dlopen() keeps reference count per process, so relying on this functionality for reloading of new applications is completely unreliable. The only reliable way is to gracefully restart Apache.

Maximum nesting depth
=====================

The combined nesting depth of #for and #if/#unless commands is limited to RXV_SPIN_MAX_DEPTH, as defined in private.h, which is currently 32. Why have such a limit and why is the limit so low?

The limit is there to make the code of mod_spin simple and fast and to avoid logic errors in templates caused by inadvertent use of deep nesting. The limit is low because templates that require nesting depth anywhere near this limit are doing something very, very wrong. The purpose of template language constructs is not to introduce programming logic (in the sense of solving the business problem the application is meant to solve), but to make simple presentation level choices depending on the data generated by the application. I cannot stress enough that ALL business logic should be in the application and application alone.

So, the whole thing is designed on purpose. You are not supposed to have a great variety of commands available in your template language, you should not be able to modify the data from within the template language and you should not give in to the temptation of fixing programming issues inside the template. In my experience, a few nesting levels in the template are quite sufficient for majority of the real world problems. The limit currently set is way above that.

However, if you find that this is not adequate for you, for whatever reason, feel free to modify private.h and recompile.

Presentation issues and the application
=======================================

You'll find that some of the presentation level decisions will be done within your application as well (huh?). When given the choice of placing some presentation level logic into the application as compared to contaminating the template with business logic, I have chosen to go with the former. Cleverly designed application will have a separate part that makes data generated by business logic into a presentation friendly format. For instance, when (X)HTML pages are created, some characters, like ``"'' and ``&'' have special meaning. Business logic won't bother itself with making sure those are escaped. However, the part of your application that makes sure presentation is nice, will. Another example is a list of items on the page that should have rows displayed in alternating colours (this particular problem can be solved by newer version of CSS, but the browsers that support that are still not in widespread use). Business logic, again, won't bother itself with that. Presentation "beautifier" will.

For instance, one might have a boilerplate API calls (similar to what mod_spin provides already, as indicated below) that adds columns to the rows data type with a sole purpose of marking certain things for the template to pick up. One such example would be to add a column "firstrow", which would have all data NULL, except for the first row. Similar can be done for the last row. Again, similar can be done for alternating rows (odd/even).

Then the template can have:

```
#for(${rowsofdata})
  #if(${rowsofdata.firstrow})
    Output this only on the first row
  #end

  #unless(${rowsofdata.firstrow})
    Output this for all rows except for the first one
  #end

  #if(${rowsofdata.oddrow})
    Output this only on the odd row
  #else
    Output this only on the even row
  #end

  #unless(${rowsofdata.lastrow})
    Output this for all rows except for the last one
  #end

  #if(${rowsofdata.lastrow})
    Output this only on the last row
  #end
#end
```

These API calls would then fall into the "beautifier" category. Use your imagination to come up with more...

How do I include other templates?
==========

I find that duplicating functionality is not a good thing. So, I tried to stay away from that. Apache already has mod_include, which can be used as a filter or a handler and provides excellent support for inclusion of other files.

If the files that you're including a not dynamic (at least not very dynamic), you should even consider generating finished files beforehand, using some of the available replacement techniques, such as XSLT. This will be good for the performance of your web server. On my old 1 GHz Athlon system, I have benchmarked Apache 2 and it was capable of delivering around 2,500 static pages per second, each around 10 kB in size (that's 25 MB/s bandwidth). Tomcat behind Apache was able to deliver around 60 dynamic pages per second, of roughly the same size, on the same machine (that's 600 kB/s bandwidth). It is worth an effort to reduce what's dynamic to a minimum.

Template file size
==========

With Apache 2.0.49 and the APR that comes with it, running on Fedora Core 1, apr_off_t, off_t, apr_size_t and size_t are all 32-bit integer values. That means that the maximum template size on this platform is 2 GB (I'm not sure what you'd use such large web pages for, but nevertheless). I'm guessing on 64-bit platforms those values would be 64-bit integers, which would make possible template size much, much larger, but I have not verified that.

Session and application tracking
==========

The simplest way would be to use mod_usertrack, which is part of Apache. This is in fact what mod_spin, up to and including 1.0.4 did. However, the cookie generated this way is very predictable (it is simply a timestamp), so anyone could easily figure it out. That's why mod_spin 1.0.5 and above uses a different approach. It relies on mod_unique_id to provide a unique session identifier, then it produces an MD5 hash of it, using the crypto salt. Both of these (unique id and the hash) are then served to the client in a cookie, usually called SpinSession. Only if both of these are returned back to the server correctly, will mod_spin use this unique id as the session id.

Otherwise, the session simply won't exist. This should make both guessing of session identifiers and denials of service attacks caused by opening of fake sessions significantly more difficult.

As of mod_spin 1.0.5, you must define SpinCookie configuration parameter, or the sessions won't be supported for that application at all.

Each session will have corresponding SDBM files (.dir and .pag) in the SpinWorkspace directory (if defined), named after the session id. Each application will have those as well, named __app.dir and __app.pag. There is nothing special about these files - they are simply a collection of key/value pairs. Through a simple API, you can get values for each key, either on the application level (i.e. shared among multiple sessions) or session level (i.e. private data).

Given those things are just files, the maintenance of stale sessions is easy. Simply define a cron job that goes around and kills whatever is older than you consider a valid session (i.e. has not been accessed for longer than defined). Note that directories for keeping application and session data are considered private (i.e. read/write by owner only) and they cannot be symlinks. The code of mod_spin will refuse to use them if they are not. You also need to make sure that nothing but application and session data is stored in this directory. Otherwise, it may collide with application and session files.

Although basic concepts have been pinched from JSP/Servlet world, applications have a slightly different meaning in mod_spin. Basically, whatever uses the same application database file falls under the "same application" umbrella. You can configure SpinWorkspace per server, virtual host, directory or location. So, applications can cross boundaries freely. Sessions are also following the same rule, so you can have multiple session private data for different definitions of SpinWorkspace.

Application configuration
==========

Each application can (but doesn't have to) have a configuration file. The filename is specified via the SpinAppConfig run-time configuration directive. The file is regular XML and it looks like this:

```
<?xml version="1.0"?>
<!DOCTYPE spin [
  <!ELEMENT spin (prop*)>
  <!ELEMENT prop (#PCDATA)>
  <!ATTLIST prop name CDATA #REQUIRED>
]>
<spin>
  <prop name="spinparameter1">The value associated with spinparameter1</prop>
  <prop name="spinparameter2">The value associated with spinparameter2</prop>
</spin>
```

It is preferred to include the DTD in the document (it is only small) in order to avoid parsing problems.

The configuration is loaded and reloaded automatically by mod_spin. Once the configuration is parsed, the keys and values of the <prop/> tags are placed into the application's SDBM file. Every time this file is opened, the configuration file is checked for modification. If the configuration file is newer, it is parsed again and the keys and values are reloaded into the SDBM file.

Authentication
==========

Apache provides enough authentication mechanisms to not duplicate this functionality in mod_spin. And because Apache's request_rec structure contains all environment variables, the information about the user using the resource

is always available to your applications. At this point in time, I did not feel that keeping user data similar to session and application data was necessary. Things like that mostly belong into the application.

However, you can wrap Apache authentication with the spin_auth application and small amount of your own code. See spin_auth and spin_app applications for all details.

Connection pools
==========

mod_spin has a simple API for accessing SQL relational databases. In order to improve performance of connecting to database (and other) servers, mod_spin uses the popular pool approach. Each connection is identified by the type (of the database) and the connection string, which are specified when the connection is opened. mod_spin creates a hash of all those connections and stores connection structures, which are database specific, as values in this table. Any subsequent attempt to open a connection to the database of the same type and with the same connection string (the keys are case sensitive) will reuse the existing connection. This can dramatically improve performance of applications that frequently use (database) connections.

Each Apache thread will have its own pool of connections (see also the threading discussion that follows). While this is good for performance, it has downsides.

With every thread having its own private connections to the back-end server, the total number of connections can be rather big (i.e. number of threads multiplied by number of connections per thread). Each connection takes memory, CPU cycles and sockets for communication, which, depending on the number of connections, might not be negligible. This alone can overwhelm the machine and can ultimately result in denial of service. That is another reason why it is a good idea to run a separate instance of Apache for heavily loaded applications. Luckily, Apache is fast to start and it doesn't consume a lot of memory (in today's terms), so you can have many instances of it running at once. With this approach, you're turning your Apache server into a transaction processing server.

As of version 1.0.2 of mod_spin, connection pools have been made more generic. Now you can register any type of connection with the connection pool. It will be treated as RXV_SPIN_CONN_FOREIGN type and as long as it has a unique connection string, you should be fine. This can then be used for any type of connection you'd like to keep hanging around for the lifetime of the thread. LDAP and similar services come to mind first.

By all means, this kind of simple database API will not be everyone's cup of tea. There are very nice alternatives (SQL Relay comes to mind first) that solved all of these problems and more. Also, some people prefer to program in a truly cross platform solutions like ODBC. Feel free to completely ignore mod_spin's database API.

Threading
=========

Some Apache 2 MPMs (Multi-Processing Modules), e.g. worker, spin off multiple threads of execution. Also, you might spin off some threads in your application code as well. There are several issues that might be affecting thread safety, most important being connection pool, followed closely by SDBM database handles used for application/session tracking.

SDBM database handles are allocated from each thread when they are needed (i.e. the SDBM files are opened by the thread for the thread). So, as long as you don't spin off any of your threads, you should be fine. If you do spin off threads and want to use the same handles (they are stored in rxv_spin_guts_t), you MUST SYNCHRONISE, or you might experience weird problems in database access, especially in terms of locking (SDBM in not capable of promoting a shared lock into an exclusive lock). This, of course applies to other variables

within all structures as well, so having mutex/rwlock variables outside of the context structure is a must.

The situation with connection pools is slightly more complicated. Each thread will have its own connection pool. The separation is achieved using apr_threadkey_private_* set of functions, which on Linux/Unix map into pthread_key_* functions. However, if you spin off your own threads, you MUST SYNCHRONISE access, or you will experience problems.

mod_spin code does not do any synchronisation of its own (i.e. it is thread unsafe), simply because it makes sure beforehand that all variables are strictly something a single thread can use without worrying about any other threads. This ensures there is no resource competition among threads.

IMPORTANT NOTE: As of MIT Kerberos (krb5-libs) version 1.3.6 and PostgreSQL 7.4.6, the combination is not thread safe. If you use worker (or other thread based) Apache MPM, you can experience segfaults. There are also some memory leaks associated with krb5 library, so if you don't use database pools, you may see Apache child processes slowly growing in size (the memory leaks have been fixed in the upcoming version 1.4 of krb5-libs). A periodical graceful restart should fix that. Generally speaking, a prefork MPM is recommended in these scenarios.

PostgreSQL code
==========

Function conninfo_parse(), located in the file spin/db.c, has been modified from the PostgreSQL 7.4.1 distribution. This function is Copyright (c) PostgreSQL Global Development Group and Regents of the University of California. It is included here under the following licence:

Portions Copyright (c) 1996-2003, PostgreSQL Global Development Group

Portions Copyright (c) 1994, The Regents of the University of California

Permission to use, copy, modify, and distribute this software and its documentation for any purpose, without fee, and without a written agreement is hereby granted, provided that the above copyright notice and this paragraph and the following two paragraphs appear in all copies.

IN NO EVENT SHALL THE UNIVERSITY OF CALIFORNIA BE LIABLE TO ANY PARTY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, ARISING OUT OF THE USE OF THIS SOFTWARE AND ITS DOCUMENTATION, EVEN IF THE UNIVERSITY OF CALIFORNIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

THE UNIVERSITY OF CALIFORNIA SPECIFICALLY DISCLAIMS ANY WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE PROVIDED HEREUNDER IS ON AN "AS IS" BASIS, AND THE UNIVERSITY OF CALIFORNIA HAS NO OBLIGATIONS TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS, OR MODIFICATIONS.

Licensing exceptions
==========

The code of mod_spin is licensed under the terms of the GNU General Public Licence, or GPL for short. However, I have made exceptions in certain files of mod_spin to make it possible to link this code with Apache itself, its modules, as well as dynamically link shared libraries that are mod_spin applications.

It would not be legally possible to link mod_spin against Apache (both dynamically and statically) unless this exception was made. Also, it would not be possible to distribute statically linked Apache that includes mod_spin. This exception takes care of that as well. You MUST OBEY THE GPL for all mod_spin code.

It would also not be legally possible to link any non-GPL licensed mod_spin

```
applications (shared libraries) dynamically, at run-time, into mod_spin.
Because I do not want to attempt to force anyone to use a particular licence
for their own work, you get permission to dynamically link, at run-time, any
of mod_spin applications (shared libraries) with mod_spin. Furthermore, Apache
can have dynamically linked modules that aren't licensed under the GPL, which
would also cause legal problems. The exception makes sure this is OK too.
This dynamic linking has to be through the interface of SpinApplication and
SpinAppEntry or LoadModules run-time configuration directives of Apache, as
provided by mod_spin code or Apache itself. Nothing but dynamic linking of
mod_spin applications and Apache third party modules is covered by this
exception and you MUST OBEY THE GPL for all mod_spin code.

If you modify mod_spin code, you may extend these exceptions to your version of
the file, but you are not obligated to do so.
```

Installation and configuration

See also:

Introduction News Licence

```
Where to get mod_spin?
======================

It is available from here:

ftp://ftp.rexursive.com/pub/mod-spin/

You can also get source and binary RPM packages, built on and for Fedora Core
Linux distribution on x86 architecture. Skeleton application (i.e. an example
application) is available from here:

ftp://ftp.rexursive.com/pub/spinapps/app/

Unless you already have your own mod_spin applications, it can be useful for
testing if the installation of mod_spin actually worked. Source and binary
RPM packages are also available.

What else is required to build
==============================

1. Apache 2: http://httpd.apache.org/. Plus, you have to apply the following
   patch to Apache 2.0.49 and below (later versions have the fix in place):

================================================
diff -ruN httpd-2.0.49-vanilla/server/core.c httpd-2.0.49/server/core.c
--- httpd-2.0.49-vanilla/server/core.c 2004-03-09 09:54:20.000000000 +1100
+++ httpd-2.0.49/server/core.c 2004-03-22 18:56:29.000000000 +1100
@@ -2975,7 +2975,7 @@
     }

     /* Seek the file to 'offset' */
-    if (offset != 0 && rv == APR_SUCCESS) {
+    if (offset >= 0 && rv == APR_SUCCESS) {
         rv = apr_file_seek(fd, APR_SET, &offset);
     }
================================================

2. Apache Portable Runtime and Apache Portable Runtime Utilities - these come
   with Apache 2, so nothing special is required to get them; the home page of
   APR project is here: http://apr.apache.org/

   IMPORTANT NOTE: You should always use the version of APR/APU that comes
   with Apache. If you pick another version, especially with a different major
   number, you may get a nasty surprise.
```

3. Apache Request Library, version 2 (libapreq2); the home page of this project
   is here: http://httpd.apache.org/apreq/. Apache 2 requires version 2 of this
   library/module and will not work with version 1; mod_apreq2 has to be
   installed and configured for mod_spin to work.

   IMPORTANT NOTE: mod_spin 1.0.10 and above is known to work with
   libapreq2-2.06-dev and above (i.e. the multi-env version of this
   library/module).

4. XML C parser and toolkit, libxml2. You can get if from here:
   http://www.xmlsoft.org/. Most Linux distributions already include it.
   Apache Portable Runtime Utilities Library includes some support for XML,
   but libxml2 is a much more mature implementation.

5. A Linux (or Unix) system with Apache 2, APR, APR Util, libxml2 and
   libapreq2 INSTALLED!

6. If you intend to rebuild scanner.c and parser.c files from scanner.l and
   parser.y, respectively, you'll need Flex version 2.5.31 or better, which
   comes with reentrant C scanner support. Versions below that won't work! You
   can get this version of Flex from http://lex.sourceforge.net/. Also,
   having Bison version 1.875 or better is recommended.

How to build and install
========================

There are three options to install mod_spin into Apache 2:

1. As a binary RPM on Fedora Core Linux Distribution
2. As Dynamic Shared Object (DSO) from source
3. By static linking into Apache 2 from source

IMPORTANT: You have to have Apache 2 configured, compiled and installed BEFORE
you attempt this procedure (either as an RPM or from source). You MUST have a
working apxs for options 2 and 3, otherwise nothing will work!

The various config scripts
==========================

Packages that mod_spin depends on come with their own configuration scripts.
Examples include apr-config, apu-config, apreq2-config, xml2-config etc. The
first two will normally reside in the binary path returned by apxs. Others
may be in various places on the system. If you want to use a particular one,
set the PATH variable so that it points to the version you want when you're
running the configure script.

Note on environment variables
=============================

Depending on the system you're running and the features you select, some
environment variables will have to be set, namely CFLAGS, CPPFLAGS and
LDFLAGS. Normally, they are picked up by the above configration scripts.

For instance, if you're compiling in MySQL support on Fedora Core 1, the
libraries will not be in /usr/lib, but in /usr/lib/mysql. Also, the headers
will not be in /usr/include, but in /usr/include/mysql. To make sure configure
finds the libraries and headers, the configure script will attempt to detect
the correct location of header files and library dependencies, by using
mysql_config. If that doesn't work, you'll have to run:

```
CPPFLAGS="$CPPFLAGS -I/usr/include/mysql \
LDFLAGS="$LDFLAGS -L/usr/lib/mysql" ./configure [other options here]
```

This would ensure that the compiler and linker can find the required files.
Some other systems may have those files in a different location, so you'll
have to know certain things about your system before you attempt this.

The same can be said for libxml2, which is a requirement to build mod_spin, as of version 0.9.1. For instance, this library installs its header files in /usr/include/libxml2 on Fedora Core 1. The configure script will attempt to figure that out at the beginning of its run by running xml2-config. Sometimes this won't work as expected. In that case pass relevant -I option on CPPFLAGS variable and -l/-L options in the LDFLAGS variable, similar to the MySQL example above.

The CFLAGS environment variable is used to pass optimisation and other flags to the C compiler. For instance, you might want to see all warnings and compile in debugging support. You would specify:

```
CFLAGS="-Wall -g" ./configure [other options here]
```

The above is relevant to systems that have GNU C Compiler (gcc). If you use some other compiler, you should check its documentations for the options.

Installation of libraries

Default prefix is /usr/local, making the installation of libraries go into /usr/local/lib. If you want rxv_spin library in the system location (i.e. /usr/lib), specify /usr prefix during configuration. Note that this may clash with rxv_spin library installation from the RPM, if you do have one.

WARNING: rxv_spin library (i.e. librxv_spin.* files) are ABSOLUTELY REQUIRED to run both mod_spin and any applications that you may have. If the library isn't installed correctly, things will break! Luckily, all three installation options install libraries automatically and this usually isn't a problem.

Library conflicts

If you have an RPM installed on the system that includes a library (including mod_spin), the location will usually be /usr/lib. When building from source, mod_spin will link against a specific library, possibly located in /usr/local/lib, to avoid versioning conflicts and misterious problems with your module when certain RPMs are removed. Use ldd to verify that you finished mod_spin has correct dependencies.

Macro files for aclocal

Beginning with mod_spin 1.0.9, an installable M4 macro file, mod_spin.m4, is located in the m4 subdirectory of the distribution. It gets installed into the aclocal's ACDIR directory (to verify what that is on your system, run 'aclocal --print-ac-dir'). This macro file is required for all mod_spin applications that want to use some mod_spin specific configure tests and it simplifies writing of your own configure.ac files. All mod_spin macros start with RXV_SPIN_, to distinguish them from other macros. You can see how they are used in various mod_spin applications, including the skeleton application, spin_app.

On most systems there is only one installation of aclocal. Therefore, both packaged installation (via RPM or some other mechanism) and installation from source would attempt to install this file in the system wide location. In all cases, package installed file will be kept, as it is always marked with a comment 'dnl PACKAGE' in the last line of the file. If you want to overwrite this file with the M4 file from the source build, you will have to copy it to the system wide location manually. This discussion only applies if both package based and source based installation are used on the same system.

IMPORTANT: If you install mod_spin using an unprivileged user account, there is a good chance you won't be able to write to system wide aclocal location (usually one needs to be root to do this). Therefore, mod_spin.m4 file IS NOT GOING TO BE INSTALLED. If you still need it for your applications, either

```
convice your system administrator to install the file in the system wide
location, copy the file into the m4 subdirectory of your application or use
a local installation of aclocal.

Session support
===============

As of mod_spin 1.0.5, session identifiers are taken from mod_unique_id and
hashed after concatenating SpinSalt configuration parameter to them. This then
creates a method of protection against session spoofing and hijacking. For
sessions to work at all, YOU MUST set SpinSalt parameter, which cannot be less
than 30 characters long. On purpose, there is no default for this parameter,
because it is supposed to be secret, somewhat random and unique to your server
(or group of servers).

Option 1: From binary RPM
=========================

You have to be using Fedora Core on an x86 machine to do this (or, if you
built your own binary RPM, any distro and architecture you did this for). If
you are, then it's really simple:

rpm -Uvh mod_spin-X.Y.Z-R.i386.rpm

where X.Y.Z is version of mod_spin and R is the RPM release of it. Such
installation will create a configuration file for mod_spin, located here:
/etc/httpd/conf.d/spin.conf. This is just an example file, but it should work
for the simplest of applications. You'll have to edit it to actually deploy
your own mod_spin applications, of course. Once you do that, any further
upgrades of mod_spin shouldn't overwrite the configuration file you fiddled
with, courtesy of RPM. Or you can provide applications' own configuration
files, especially if you're deploying those as RPMs as well.

If you want to develop software for mod_spin and you have an RPM installation,
you have to install the development RPM too (mod_spin 1.0.10 and above):

rpm -Uvh mod_spin-devel-X.Y.Z-R.i386.rpm

Option 2: DSO from source
=========================

From the top directory of mod_spin source do:

./configure [--prefix=/top/dir] [--with-apxs=/path/to/apxs] \
            [--with-pgsql] [--with-mysql] \
            [--with-flex-reentrant=/path/to/flex/installation/directory]
make
make install

Option 3: Static linking from source
====================================

From the top directory of mod_spin source do:

./configure [--prefix=/top/dir] --with-apache=/path/to/apache2/source \
            [--with-apxs=/path/to/apxs] \
            [--with-pgsql] [--with-mysql] \
            [--with-flex-reentrant=/path/to/flex/installation/directory]
make
make install

From the top directory of Apache 2 source do:

If this is the first time you are installing mod_spin into this source tree of
Apache 2, you will have to build appropriate files in modules/spin directory:

./buildconf
```

```
IMPORTANT: If you compiled Apache 2 from this source tree before, you should
run:

make distclean

If you don't, compile or link may fail.

Configure Apache 2, either by using an existing configuration (you can also
edit config.nice and place --enable-spin there):

./config.nice --enable-spin

or by using brand new set of options:

./configure --enable-spin [your options here]

Later, if you don't want mod_spin (why would you ever want to do that ;-), then
specify --disable-spin in your configuration options. Now build Apache 2 and
install it:

make
make install

Uninstall
=========

If you installed from RPM, just do:

rpm -e mod_spin

and:

rpm -e mod_spin-devel

if you installed the development package.

In cases 2 and 3, you can simply run from the top directory of mod_spin source:

make uninstall

If you did the install into Apache 2 source for static linking, you should
run:

./buildconf

from the top directory of the Apache 2 source tree. This should clean up the
configure script and remove mod_spin options from it.

How to build RPMs
=================

You can do it from the tarball by:

rpmbuild -ta mod_spin-X.Y.Z.tar.bz2

The above will build both source and binary RPMs. Or you can build a binary
RPM from the source RPM like this:

rpmbuild --rebuild mod_spin-X.Y.Z-R.src.rpm

Most systems don't come with reentrant flex (this feature is available in
version 2.5.31 and above), so rebuilding the RPMs may fail during the
dependency check phase. However, replacing flex with a reentrant package may
have system wide implications, especially if you compile software that
produces scanners. Therefore, mod_spin, as of version 1.0.10, depends on
flex-reentrant package, which can coexist peacefully alongside flex on your
```

RPM based system. You can get this package from this location (just look for the latest version):

ftp://ftp.rexursive.com/pub/flex/

Although source RPMs are normally versioned according to the distribution they have been built on (for instance, SRPM built on Fedora Core 3 will have FC3 in its name), you can still use them on other distributions. They may not install cleanly due to dependency resolution issue, but most times a rebuild from the source RPM will provide you with a binary package you can use.

Doxygen documentation
=====================

References to external documentation, namely that of Apache Portable Runtime, Apache Portable Runtime Utilities Library and Apache Request Library, are all done to the URLs at various Apache web sites. If you're installing from source, in order to link to local documentation instead of one on the net, use the installdox Perl script, located in the docs/html directory (after you performed make install). Something like this should do the trick (of course, substitute example locations with the ones on your system):

```
./installdox -l apr.tag@/usr/local/doc/apr \
             -l apu.tag@/usr/local/doc/apr-util \
             -l apreq2.tag@/usr/local/apache2/share/libapreq2/docs/html
```

For the RPM installation, the RPM post install script will attempt to detect what documentation you have locally and run installdox for you. If you install some documentation after you put mod_spin RPM on your system, you can either run installdox manually (like in the example above) or you can reinstall the RPM (the --force option does wonders here), which will run the post install script again.

For your convenience, all mod_spin packages contain mod_spin.tag file, to allow easy referencing from your documentation.

Module loading order
====================

The Apache Request Library module must be loaded before mod_spin, or your Apache web server won't be able to resolve some symbols. So, have them like this in your httpd.conf file:

```
LoadModule apreq_module modules/mod_apreq2.so
LoadModule spin_module  modules/mod_spin.so
```

Luckily, when deployed in a standard Red Hat manner via the RPM, where each module gets its own configuration file in conf.d directory, apreq is alphabetically before spin, so it loads first. Phew!

Note on cookies
===============

As of Apache 2.0.48, support for RFC2965 cookies still has problems. It is best to stick with Netscape cookies for now. Also, Apache 2.0.48 requires you to specify CookieName or things won't work. This has been fixed in Apache CVS and it will appear in later Apache versions.

SELinux environments
====================

If you are running Security Enhanced Linux (for instance, Fedora Core 3), you may need to do a bit of extra configuration for mod_spin to work correctly. For instance, if you place application/session database files in /var/tmp/myapp directory, you'll have to make sure that this directory belongs to the correct security context, which is normally root:object_r:httpd_cache_t. To do that,

```
you'd run:

chcon -R -u root -r object_r -t httpd_cache_t /var/tmp/myapp

Alternatively, you can set up a context file like this (myapp.ctx):

/var/tmp/myapp     -d  root:object_r:httpd_cache_t
/var/tmp/myapp/.*      root:object_r:httpd_cache_t

You should then apply this context to the directory:

setfiles myapp.ctx /var/tmp/myapp

Without this, you may have Apache running in a different security context and
being denied access to your application/session database files. Meaning, every
time mod_spin attempts to access /var/tmp/myapp directory, although file
permissions allow it, file access would be denied by policy and the client
would get an internal server error response.

Configuration
=============

Here is an example snippet from httpd.conf (comments after configuration
parameters will cause errors - they are here only to point out the default
settings):

<IfModule mod_spin.c>
    AddHandler spin-template .sm
    AddType text/html .sm
    SpinApplication /usr/local/libexec/spinapps/spinapp.so
    SpinAppEntry rxv_spin_service                    # Default
    SpinAppConfig /usr/local/etc/spinapps/spinapp.xml
    SpinWorkspace /var/tmp/mod_spin
    SpinCookie SpinSession
    SpinSendfile on                                  # Default
    SpinCacheAll off                                 # Default
    SpinClearCount 0                                 # Default
    SpinSalt                                         # No default
</IfModule>

SpinApplication, SpinAppEntry, SpinAppConfig, SpinWorkspace, SpinCookie,
SpinSendfile and SpinCacheAll can be placed in the main server section, virtual
host section, directory or location section. The nearest available will be
used. SpinClearCount and SpinSalt can only be placed in the global
configuration, as it doesn't make sense anywhere else.

SpinApplication is the path to the shared library that is your application.

SpinAppEntry is the name of the entry function in the shared library to call
(rxv_spin_service() by default). Function named in SpinAppEntry will be called
after this library is dynamically linked by mod_spin handler.

SpinAppConfig is the path to the application configuration file. This is a
simple XML file (so that we can use XML parsing facilities easily) which looks
like this:

<?xml version="1.0"?>
<!DOCTYPE spin [
  <!ELEMENT spin (prop*)>
  <!ELEMENT prop (#PCDATA)>
  <!ATTLIST prop name CDATA #REQUIRED>
]>
<spin>
  <prop name="spinparameter1">The value associated with spinparameter1</prop>
  <prop name="spinparameter2">The value associated with spinparameter2</prop>
</spin>
```

Given that parsing of the configuration file is a very rare event (application parameters are kept in the database file after that), the overhead is not big. The DTD of the document is preferably always local, simply because it is very small and makes parsing a predictable event.

SpinWorkspace is a directory location where temporary files for session and application data will go. On Linux (and possibly other systems), you can use /dev/shm (temporary file system support in shared memory) for this, in order to improve performance. If the system is rebooted, you will lose those files, however. This directory must be readable, writable and executable by owner only and it cannot be a symlink.

SpinCookie is the name of the cookie used for session management. This parameter is optional, that is to say, if it isn't defined, sessions will not be supported for that particular configuration section. By convention, this is normally called SpinSession, but it can be called anything you like. As of mod_spin 1.0.5, session cookie handling is entirely done in mod_spin. You need to define SpinSalt for sessions to work.

SpinSendfile enables sendfile() support for pushing the data from the template file onto the socket. Default is on. Disable this for file systems that have problems with sendfile().

SpinCacheAll enables caching of all text for every template. This can improve performance, but it'll use more memory. By default, this switch is turned off and all file chunks 256 bytes in size and over will be put into the brigade as FILE buckets, rather than IMMORTAL buckets.

SpinClearCount is the number of requests that a thread private pool and its sub-pools will persist. The resources associated with those pools are parsed templates, as well as pooled database connections. By setting this value to more than zero (0), those resources will be periodically released.

SpinSalt is the cryptographic salt used to add randomness to the session id hashes. It has to be at least 30 characters long. DO NOT SET THIS TO THE SAME VALUE ON ALL YOUR SERVERS UNLESS YOU'RE RUNNING A CLUSTER of machines that share sessions, as it'll be easier for potential attackers to hijack your sessions. The best thing to do is to make this a "random" string, something that doesn't make any sense in any language.

Strangely enough (as of mod_spin 1.0.6), you can set SpinSalt parameter twice and that is not an error. This functionality exists to enable security savvy administrators to rotate cryptographic salt, in order to make session hijacking even more difficult. An external program, possibly started from cron, can manipulate SpinSalt parameters in the configuration file. The first specified SpinSalt will be considered new and the second one will be considered old, but still valid (other instances of SpinSalt will be ignored). After a graceful restart of Apache, mod_spin will accept sessions encrypted with both the old and the new salt. All outgoing sessions will be encrypted using the new salt. This enables smooth transition between different crypto salts.

News

See also:

Introduction Installation Licence

```
* version 1.0.10 released 2005-08-05

** don't fail if mod_spin.m4 cannot be installed
** adjust to multi-env libapreq2 (2.06 and above)
** use flex-reentrant if available
** context now uses apreq_handle_t
** use --include instead of --cflags for newer mysql_config
** use Bitstream fonts in doxygen.css
```

```
** make GCC 4.0 complain less
** check for ap_regex headers
** fix incorrect AC_PATH_PROG calls

* version 1.0.9 released 2005-04-04

** make it compile and run with Apache 2.1.x and APR 1.1.x
** fake ap_construct_url() in spin.c for testing
** split up M4 macros into separate, installable file
** extend rxv_spin-config with docdir
** process doxygen.conf and mod_spin.spec with configure

* version 1.0.8 released 2005-01-29

** use ap_discard_request_body() in handler

* version 1.0.7 released 2005-01-24

** fix crummy md5b64_validate() code
** get rid of the strcpy() calls
** improve RPM spec file

* version 1.0.6 released 2005-01-13

** fix SpinCookie/SpinSalt documentation
** add salt rotation functionality
** actually use salt in MD5 hashing
** mention SELinux installation

* version 1.0.5 released 2004-12-23

This release has emphasis on security. It is also the first version to use
mod_unique_id for session tracking, rather than mod_usertrack. Note that
cookies are now served directly by mod_spin. A new configuration parameter,
SpinSalt, is used to introduce an unknown into the creation of session id MD5
hashes, thus preventing theft of sessions and (some) denial of service attacks.
Application and session tracking now requires a private directory and mod_spin
refuses to use the files if the directory is not read/write by owner only or if
it's a symlink.

** SECURITY: check session/application store path/permissions
** SECURITY: use mod_unique_id for session IDs
** SECURITY: add MD5 hashes of session IDs
** handle session cookies from within mod_spin
** introduce SpinSalt configuration parameter
** make clearcnt configuration parameter work
** drop SpinBasename configuration parameter
** introduce rxv_spin_ses_idget() function
** introduce rxv_spin_ses_valid() function
** make sessions optional

* version 1.0.4 released 2004-10-12

** provide FC version in the RPM release

* version 1.0.3 released 2004-09-23

** fix bogus installdox logic

* version 1.0.2 released 2004-09-21

Note that in this version there has been a rework of connection pool
functionality. The whole thing has been made more generic, so that other
connection types can be registered with the pool, not just database ones.

The rxv_spin_conn_t has one extra member, the cleanup function, so, you will
HAVE TO RECOMPILE YOUR APPLICATIONS, because this version is obviously BINARY
```

```
INCOMPATIBLE WITH PREVIOUS VERSIONS. Also, the symbol for
rxv_spin_db_pool_create() function no longer exists in the library. It has
been replaced with rxv_spin_cpool_create(). However, compatibility macros have
been defined throughout, so recompilation should go smoothly.

** move connection pools into a separate module
** make rxv_spin_db_pool_* legacy
** make rxv_spin_db_conn and rxv_spin_db_conn_t legacy
** make connection keys case sensitive
** untie connection macros from database functionality
** use request pool where possible to reduce memory pressure
** fix redundant PQclear() call
** add cleanup function to rxv_spin_conn_t
** ship eatdoxygen.c
** avoid copying of context data to reduce memory pressure

* version 1.0.1 released 2004-08-10

** move rxv_spin_service_t back into rxv_spin.h for docs
** fix bogus tag file logic

* version 1.0.0 released 2004-08-04

This is the first 'stable' release of mod_spin. For the most part, I would
prefer to describe it as 'useful', since one can actually have applications
that run mostly correct and perform reasonably well inside the mod_spin
framework. As with any software, I'm sure there are bugs in this code too. How
many, only the time and a lot of use will tell. So, please, if you do find
them, let me know.

Contrary to what many software projects do, I do not have plans for near
future development releases of mod_spin. I'll be focusing more on two things:

1. That the code of mod_spin is stable, secure and performs well.
2. Some useful mod_spin applications.

As I find time, I will experiment with mod_spin inside the current development
version of Apache, 2.1. After all, that's where the future is :-)

** delay XML parser cleanup until the end of the request
** fix possible segfaults with apr_pool_cleanup_register()
** remove rxv_spin_db_clean() function
** remove rxv_spin_db_pool_destroy() function
** remove rxv_spin_db_finish_do() function
** use pool cleanups for database stuff
** remove bogus child cleanups
** rxv_spin_ctx_t: shorthand for rxv_spin_context_t
** improve logging of critical events
** generate Doxygen tag file
** reference external documentation
** change Doxyfile to doxygen.conf and make clearer
** add database type enquiry macros
** move rxv_spin_service_t into private.h
** fix missing tag files

* version 0.9.13 released 2004-06-20

This version features major performance improvements, especially if the
applications you're using are pulling in large or many shared libraries.
However, in order to reliably reload applications when new versions are
deployed, a graceful restart of Apache is now required.

A configuration script, rxv_spin-config, has been introduced in this version,
to allow easy collection of mod_spin installation directory locations. You can
enquire about prefix, bindir, libdir, libexecdir and includedir. This is useful
for configuration of mod_spin applications.
```

```
IMPORTANT: This version (and all future versions) depend on apreq2-config
script being properly installed. As of libapreq2-2.03_04, this is not the case
(unless you installed one of my RPMs). So, you'll have to make sure
apreq2-config is copied into the correct location for mod_spin to build from
source (this is normally /usr/local/apache2/bin or /usr/bin, depending on how
you installed libapreq2).

** fix library installation text
** cache dynamically loaded libraries
** fix linking against librxv_spin
** use apreq2-config
** make rxv_spin-config script

* version 0.9.12 released 2004-06-10

** split into mod_spin and librxv_spin
** fix static linking into Apache
** move RXV_SPIN_MAX_DEPTH into private.h

* version 0.9.11 released 2004-05-28

** size fix for rxv_spin_rows()
** use apr_pcalloc() instead of apr_palloc()/memset()

* version 0.9.10 released 2004-05-27

** reference processing fix

* version 0.9.9 released 2004-05-24

** macro name cleanup

* version 0.9.8 released 2004-05-18

** new function: rxv_spin_single_mem()
** new function: rxv_spin_single_memset()
** more paranoia about NULL pointers
** avoid corner case database connection leak
** delay cleanup until the end of the request

* version 0.9.7 released 2004-05-14

IMPORTANT: As of this release, single data will always have a '\0' character
appended to it. This is useful for passing this data to regular C APIs that
handle strings that are terminated in such manner. If you have your own
functions that create single data, you will have to adjust them to behave like
this or you'll be EXPOSING YOU APPLICATIONS TO BUFFER OVERFLOWS.

In this version a lot of the data manipulation function changed their names.
To avoid incompatibilities, relevant macros have been defined in rxv_spin.h.
However, those compatibility macros may be removed at any time. If you have
code that uses old functions, it is best to rename. The
rxv_spin_column_markeach() function has been replaced by a function that has an
extra parameter, so when changing, take that into account. This new parameter
is offset. The compatibility macro defines it as zero (0), which was the
behaviour of the old function.

** fix wrong URL in spec file
** new function: rxv_spin_meta_parse()
** new function: rxv_spin_meta_hash()
** new function: rxv_spin_rows_select()
** new function: rxv_spin_rows_hash()
** new function: rxv_spin_rows_mark()
** new function: rxv_spin_rows_markeach()
** new function: rxv_spin_single_trim()
** renamed various data functions
** improve rxv_spin_*_markeach() functions
```

```
** make the test file real XHTML
** new function: rxv_spin_str_trim()
** change how singles are stored

* version 0.9.6 released 2000-04-14

** additional APXS check
** do with/without/enable/disable properly
** avoid copying of SQL results
** new function: rxv_spin_resize()
** refine non-pooled database connection cleanup
** new function: rxv_spin_db_clean()
** avoid destroying the brigade, cleanup will do it

* version 0.9.5 released 2004-04-07

** make SpinClearCount global only directive
** make sendfile and text cache flags template specific
** fix incorrect int return from child_init()
** add CFLAGS from apr-config
** improve important program tests
** remove application specific configuration from spin.conf
** expand dynamic linking licensing exception
** adjust RPM spec file

* version 0.9.4 released 2004-04-01

For all the negative thinking folk out there :-), mod_spin now understands
#unless, direct opposite of #if. Seriously, I thought there could be instances
where one would want to check if something isn't there directly, rather than
have the clumsy #if/#else construct. Internally, #unless is implemented using
the #if infrastructure.

Other, rather painful changes, were the name changes to some data manipulation
functions. It is a bit late in the game, but this was simply begging for a
cleanup. Sorry :-(. You can always do a few macro definitions if you want to
stick with the old names for a while in your applications. The argument order
and type did not change at all, which should make things relatively easy to
switch around.

There was also a number of useful functions added to data manipulation
section. There could be more in the future, depending on what I bump into
solving real world problems.

The support for sendfile() is now in by default and should work correctly, but
you'll have to patch you Apache for it to take effect. There is a bug in
server/core.c file of Apache up to 2.0.49, function emulate_sendfile(), that
affects the output when a brigade contains more than one FILE bucket. That was
the reason for FLUSH buckets after each FILE bucket in the previous mod_spin
code. Here is that patch for Apache:

===================================================
diff -ruN httpd-2.0.49-vanilla/server/core.c httpd-2.0.49/server/core.c
--- httpd-2.0.49-vanilla/server/core.c 2004-03-09 09:54:20.000000000 +1100
+++ httpd-2.0.49/server/core.c 2004-03-22 18:56:29.000000000 +1100
@@ -2975,7 +2975,7 @@
     }

     /* Seek the file to 'offset' */
-    if (offset != 0 && rv == APR_SUCCESS) {
+    if (offset >= 0 && rv == APR_SUCCESS) {
         rv = apr_file_seek(fd, APR_SET, &offset);
     }
===================================================

Another important fix is related to keep-alive and pipelined requests. In order
for that to be handled properly, all memory allocation related to the output
```

brigade is now done from the connection pool.

New configuration parameter SpinCacheAll will turn on template file chunks caching even when they are 256 bytes in size and over. This will consume more memory, but might improve performance.

A lot of effort in this version has been put into parsed template cache and the "poolology" of the beast. My tests indicate that mod_spin now behaves pretty good when it comes to performance (I have observed up to 25% better performance using the skeleton application, when compared to previous version of mod_spin), memory leaks and segfaults in that area. As always, non-trivial software has bugs, so if you find some (and you will), let me know.

Unfortunately (or furtunately, depending on your point of view), during mod_spin stress tests, I was able to crash Linux kernel 2.4.22-1.2174.nptl, as provided by Fedora Core 1. A bug report is in (119519). I'm not sure if this is Red Hat specific, my system specific or generic, but it did crash my notebook several times. We'll have to wait for kernel folk to address this one.

** implement #unless, opposite of #if
** new data manipulation API calls
** make exisiting data manipulation function names sane
** better flex and bison build rules
** make the test file XHTML
** new configuration parameter: SpinSendfile
** enable merging of server and directory configurations
** use connection pool for all brigade stuff
** new configuration parameter: SpinCacheAll
** new configuration parameter: SpinClearCount
** make template caching safe
** fix template cache pool usage
** be more portable in configure.ac

* version 0.9.3 released 2004-03-15

Two reasonably big things in this release. The first one related to easy installation - building mod_spin RPMs is now a no-brainer. The second one is a parsed template cache. This has been a long outstanding issue of mod_spin (i.e. all templates were always parsed on every request). Now, parsed templates are remembered per thread and if the modification time and size of the template is identical, no parsing will be done. This can speed up mod_spin up to 30%, as I have observed in my tests. Normally, if you're using database access, the difference in speed will be somewhere in the range of a few percent, but it's nevertheless welcome. With caching of parsed templates memory usage will go up, but given today's memory sizes, I don't see it as a big issue.

Unfortunatly (or fortunately), a small API change occured in this release. It is the order of arguments passed to rxv_spin_db_connect() function (which now makes more sense anyway). But more importantly, the change was warranted due to a semantical change of the function itself, related to the memory leak. It is now required to pass a valid memory pool pointer as the first argument, for all temporary memory allocations.

** RPM build support (not heavily tested)
** create a memory pool per thread to avoid locking
** parsed templates are now cached per thread
** closed serious memory leaks in rxv_spin_db_connect()
** changed the order of arguments for rxv_spin_db_connect()
** improved build system

* version 0.9.2 released 2004-03-08

The API should now be considered relatively stable. All unnecessary variables have been removed from the context and placed into a separate member 'guts', which is used internally by mod_spin and shouldn't be relied upon. This will

```
enable future enhancements to the context, while preserving current structure
layout. There is also a new member 'extra' that can be used for any user data
that needs to be placed into the context.

** add more checks to various functions
** split context stuff into context.c
** split data stuff into data.c
** split application/session stuff into store.c
** fix potential memory leak in XML parsing
** drop session base filename
** fix configuration file reload for missing page file
** split data and context functions in the documentation
** provide context guts to stablise the API
** allow extra data to be placed into the context
** eatdoxygen utility for removing Doxygen comments
** hose Doxygen comments from installed rxv_spin.h file
** build and install manual pages
** remove LaTeX documentation and put mod_spin.pdf in docs
** install HTML documentation

* version 0.9.1 released 2004-03-04

** rxv_spin_app_del(), rxv_spin_ses_del()
** even more clear licensing exceptions
** rxv_spin_ctx_del macro
** build system enhanced (auto find MySQL stuff)
** depend on libxml2
** XML configuration files
** add SpinAppConfig, application configuration file
** fixed incorrect application store linkage to cookies

* version 0.9.0 released 2004-02-27

** first beta version
** more clear licensing exceptions
** rxv_spin_db_escape() function
** rxv_spin_ctx_str_set macro
** don't escape query strings in rxv_sin_db_exec()
** switch to apr_hash_t in rxv_spin_data_t for case sensitivity
** fix ordering of rxv_spin_data structure

* version 0.0.4 released 2004-02-20

** LAST ALPHA RELEASE - FEATURE SET NOW COMPLETE
** MySQL support
** module signature to ServerSignature
** workaround for PACKAGE_* conflicts
** SpinAppEntry parameter
** REDIRECT, DONE and DECLINED handling
** create-spintest.sql
** database connections stored per type in the pool
** avoid use of apr_psprintf() for session filenames

* version 0.0.3 released 2004-01-13

** threading behaviour changed
** changed API

* version 0.0.2 released 2003-10-17

** added tests directory
** changed API

* version 0.0.1 released 2003-10-15

** Alpha quality code
** Most likely builds and works in certain circumstances
```

```
* version 0.0.1 in development 2003-05-26

** Successfuly switched to autoconf/automake/libtool build system
** Needs flex 2.5.31 to build (recursive C scanner feature)
```

Terms of copying, distribution and modification

See also:

Introduction News Installation

```
    GNU GENERAL PUBLIC LICENSE
       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
     59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

    Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.

  We protect your rights with two steps: (1) copyright the software, and
(2) offer you this license which gives you legal permission to copy,
distribute and/or modify the software.

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
software.  If the software is modified by someone else and passed on, we
want its recipients to know that what they have is not the original, so
that any problems introduced by others will not reflect on the original
authors' reputations.

  Finally, any free program is threatened constantly by software
patents.  We wish to avoid the danger that redistributors of a free
program will individually obtain patent licenses, in effect making the
program proprietary.  To prevent this, we have made it clear that any
```

patent must be licensed for everyone's free use or not licensed at all.

  The precise terms and conditions for copying, distribution and
modification follow.

    GNU GENERAL PUBLIC LICENSE
   TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

  0. This License applies to any program or other work which contains
a notice placed by the copyright holder saying it may be distributed
under the terms of this General Public License. The "Program", below,
refers to any such program or work, and a "work based on the Program"
means either the Program or any derivative work under copyright law:
that is to say, a work containing the Program or a portion of it,
either verbatim or with modifications and/or translated into another
language. (Hereinafter, translation is included without limitation in
the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope. The act of
running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the
Program (independent of having been made by running the Program).
Whether that is true depends on what the Program does.

  1. You may copy and distribute verbatim copies of the Program's
source code as you receive it, in any medium, provided that you
conspicuously and appropriately publish on each copy an appropriate
copyright notice and disclaimer of warranty; keep intact all the
notices that refer to this License and to the absence of any warranty;
and give any other recipients of the Program a copy of this License
along with the Program.

You may charge a fee for the physical act of transferring a copy, and
you may at your option offer warranty protection in exchange for a fee.

  2. You may modify your copy or copies of the Program or any portion
of it, thus forming a work based on the Program, and copy and
distribute such modifications or work under the terms of Section 1
above, provided that you also meet all of these conditions:

    a) You must cause the modified files to carry prominent notices
    stating that you changed the files and the date of any change.

    b) You must cause any work that you distribute or publish, that in
    whole or in part contains or is derived from the Program or any
    part thereof, to be licensed as a whole at no charge to all third
    parties under the terms of this License.

    c) If the modified program normally reads commands interactively
    when run, you must cause it, when started running for such
    interactive use in the most ordinary way, to print or display an
    announcement including an appropriate copyright notice and a
    notice that there is no warranty (or else, saying that you provide
    a warranty) and that users may redistribute the program under
    these conditions, and telling the user how to view a copy of this
    License. (Exception: if the Program itself is interactive but
    does not normally print such an announcement, your work based on
    the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If
identifiable sections of that work are not derived from the Program,
and can be reasonably considered independent and separate works in
themselves, then this License, and its terms, do not apply to those
sections when you distribute them as separate works. But when you
distribute the same sections as part of a whole which is a work based
on the Program, the distribution of the whole must be on the terms of

this License, whose permissions for other licensees extend to the
entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest
your rights to work written entirely by you; rather, the intent is to
exercise the right to control the distribution of derivative or
collective works based on the Program.

In addition, mere aggregation of another work not based on the Program
with the Program (or with a work based on the Program) on a volume of
a storage or distribution medium does not bring the other work under
the scope of this License.

  3. You may copy and distribute the Program (or a work based on it,
under Section 2) in object code or executable form under the terms of
Sections 1 and 2 above provided that you also do one of the following:

    a) Accompany it with the complete corresponding machine-readable
    source code, which must be distributed under the terms of Sections
    1 and 2 above on a medium customarily used for software interchange; or,

    b) Accompany it with a written offer, valid for at least three
    years, to give any third party, for a charge no more than your
    cost of physically performing source distribution, a complete
    machine-readable copy of the corresponding source code, to be
    distributed under the terms of Sections 1 and 2 above on a medium
    customarily used for software interchange; or,

    c) Accompany it with the information you received as to the offer
    to distribute corresponding source code. (This alternative is
    allowed only for noncommercial distribution and only if you
    received the program in object code or executable form with such
    an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for
making modifications to it. For an executable work, complete source
code means all the source code for all modules it contains, plus any
associated interface definition files, plus the scripts used to
control compilation and installation of the executable. However, as a
special exception, the source code distributed need not include
anything that is normally distributed (in either source or binary
form) with the major components (compiler, kernel, and so on) of the
operating system on which the executable runs, unless that component
itself accompanies the executable.

If distribution of executable or object code is made by offering
access to copy from a designated place, then offering equivalent
access to copy the source code from the same place counts as
distribution of the source code, even though third parties are not
compelled to copy the source along with the object code.

  4. You may not copy, modify, sublicense, or distribute the Program
except as expressly provided under this License. Any attempt
otherwise to copy, modify, sublicense or distribute the Program is
void, and will automatically terminate your rights under this License.
However, parties who have received copies, or rights, from you under
this License will not have their licenses terminated so long as such
parties remain in full compliance.

  5. You are not required to accept this License, since you have not
signed it. However, nothing else grants you permission to modify or
distribute the Program or its derivative works. These actions are
prohibited by law if you do not accept this License. Therefore, by
modifying or distributing the Program (or any work based on the
Program), you indicate your acceptance of this License to do so, and
all its terms and conditions for copying, distributing or modifying
the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals

of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN
OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES
PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE
PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,
REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE
POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest
to attach them to the start of each source file to most effectively
convey the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate

```
parts of the General Public License.  Of course, the commands you use may
be called something other than 'show w' and 'show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  'Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

2 mod_spin Module Index

2.1 mod_spin Modules

Here is a list of all modules:

3 mod_spin Data Structure Index

3.1 mod_spin Data Structures

Here are the data structures with brief descriptions:

4 mod_spin File Index

4.1 mod_spin File List

Here is a list of all documented files with brief descriptions:

5 mod_spin Module Documentation

5.1 Data manipulation functions

Data Structures

- struct rxv_spin_data

  Data structure.

Defines

- #define RXV_SPIN_DATA_SGL 0x01
- #define RXV_SPIN_DATA_RWS 0x02
- #define RXV_SPIN_DATA_MTA 0xFF
- #define RXV_SPIN_TRIM_LEFT 0x01
- #define RXV_SPIN_TRIM_RIGHT 0x02
- #define rxv_spin_single_trimboth(s) rxv_spin_single_trim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_single_trimleft(s) rxv_spin_single_trim((s),(RXV_SPIN_TRIM_LEFT))
- #define rxv_spin_single_trimright(s) rxv_spin_single_trim((s),(RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_column_del(rows, key) rxv_spin_column_set((rows),(key),NULL)
- #define rxv_spin_str_trimboth(s) rxv_spin_str_trim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_str_trimleft(s) rxv_spin_str_trim((s),(RXV_SPIN_TRIM_LEFT))
- #define rxv_spin_str_trimright(s) rxv_spin_str_trim((s),(RXV_SPIN_TRIM_RIGHT))

Typedefs

- typedef rxv_spin_data rxv_spin_data_t

Functions

- rxv_spin_data_t * rxv_spin_single (apr_pool_t *pool, const char *str)
- char * rxv_spin_single_get (rxv_spin_data_t *single)
- rxv_spin_data_t * rxv_spin_single_set (rxv_spin_data_t *single, const char *str)
- rxv_spin_data_t * rxv_spin_single_mem (apr_pool_t *pool, const char *str, size_t size)
- rxv_spin_data_t * rxv_spin_single_memset (rxv_spin_data_t *single, const char *str, size_t size)
- rxv_spin_data_t * rxv_spin_single_tolower (rxv_spin_data_t *single)
- rxv_spin_data_t * rxv_spin_single_toupper (rxv_spin_data_t *single)

- rxv_spin_data_t * rxv_spin_single_trim (rxv_spin_data_t *single, unsigned char what)
- rxv_spin_data_t * rxv_spin_meta (apr_pool_t *pool,...)
- rxv_spin_data_t * rxv_spin_meta_vstr (apr_pool_t *pool,...)
- rxv_spin_data_t * rxv_spin_meta_parse (apr_pool_t *pool, char *str, const char *sep)
- rxv_spin_data_t * rxv_spin_meta_empty (apr_pool_t *pool, size_t size)
- apr_hash_t * rxv_spin_meta_hash (apr_pool_t *pool, rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_meta_mark (rxv_spin_data_t *data, size_t element)
- rxv_spin_data_t * rxv_spin_meta_markeach (rxv_spin_data_t *data, size_t off, size_t step)
- rxv_spin_data_t * rxv_spin_meta_select (rxv_spin_data_t *data, rxv_spin_data_t *select, apr_hash_t *hash)
- rxv_spin_data_t * rxv_spin_rows (apr_pool_t *pool,...)
- apr_hash_t * rxv_spin_rows_hash (apr_pool_t *pool, rxv_spin_data_t *rows, const char *column)
- rxv_spin_data_t * rxv_spin_rows_mark (rxv_spin_data_t *rows, const char *column, size_t element)
- rxv_spin_data_t * rxv_spin_rows_markeach (rxv_spin_data_t *rows, const char *column, size_t off, size_t step)
- rxv_spin_data_t * rxv_spin_rows_select (rxv_spin_data_t *rows, rxv_spin_data_t *select, const char *column, const char *marker, apr_hash_t *hash)
- rxv_spin_data_t * rxv_spin_column_get (apr_pool_t *pool, rxv_spin_data_t *rows, const char *key)
- rxv_spin_data_t * rxv_spin_column_set (rxv_spin_data_t *rows, const char *key, rxv_spin_data_t *column)
- rxv_spin_data_t * rxv_spin_resize (apr_pool_t *pool, rxv_spin_data_t *data, size_t size)
- rxv_spin_data_t * rxv_spin_copy (apr_pool_t *pool, rxv_spin_data_t *data)
- char * rxv_spin_str_tolower (const char *str)
- char * rxv_spin_str_toupper (const char *str)
- char * rxv_spin_str_trim (char *str, unsigned char what)

5.1.1 Detailed Description

Data manipulation functions (mod_spin API)

5.1.2 Define Documentation

5.1.2.1 #define RXV_SPIN_DATA_SGL 0x01

single data type (size limited string)

5.1.2.2 #define RXV_SPIN_DATA_RWS 0x02

rows data type (named columns, numbered rows)

5.1.2.3 #define RXV_SPIN_DATA_MTA 0xFF

metadata type (not used in rendering)

5.1.2.4 #define RXV_SPIN_TRIM_LEFT 0x01

trim whitespace on the left

5.1.2.5 #define RXV_SPIN_TRIM_RIGHT 0x02

trim whitespace on the right

5.1.2.6 #define rxv_spin_single_trimboth(s) rxv_spin_single_trim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_RIGHT))

trim single on both sides

5.1.2.7 #define rxv_spin_single_trimleft(s) rxv_spin_single_trim((s),(RXV_SPIN_TRIM_LEFT))

trim single on the left side

5.1.2.8 #define rxv_spin_single_trimright(s) rxv_spin_single_trim((s),(RXV_SPIN_TRIM_RIGHT))

trim single on the right side

5.1.2.9 #define rxv_spin_column_del(rows, key) rxv_spin_column_set((rows),(key),NULL)

delete a column from rows

5.1.2.10 #define rxv_spin_str_trimboth(s) rxv_spin_str_trim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_RIGHT))

trim string on both sides

5.1.2.11 #define rxv_spin_str_trimleft(s) rxv_spin_str_trim((s),(RXV_SPIN_TRIM_LEFT))

trim string on the left side

5.1.2.12 #define rxv_spin_str_trimright(s) rxv_spin_str_trim((s),(RXV_SPIN_TRIM_RIGHT))

trim string on the right side

5.1.3 Typedef Documentation

5.1.3.1 typedef struct rxv_spin_data rxv_spin_data_t

data type

5.1.4 Function Documentation

5.1.4.1 rxv_spin_data_t* rxv_spin_single (apr_pool_t * *pool*, const char * *str*)

Create single data from from a nul terminated string. String is not copied - use apr_pstrdup() for that.

Parameters:

pool Pool for allocation of data structure

str String to create data from

Returns:

rxv_spin_data_t pointer, structure allocated from the pool, NULL on error

Example:

```
rxv_spin_single(ctx->pool,"some string");
```

5.1.4.2 char* rxv_spin_single_get (rxv_spin_data_t * *single*)

Get a nul terminated string from the single data. String is not copied, use apr_pstrdup() for that.

Parameters:

single Single data

Returns:

nul terminated string, NULL on error

Example:

```
rxv_spin_single_get(data);
```

Remarks:

You can, of course, completely ignore this function if you know that your data is a single and that it is not NULL. Simply doing data->data will return the same string.

5.1.4.3 rxv_spin_data_t* rxv_spin_single_set (rxv_spin_data_t * *single*, const char * *str*)

Replace data (string) in an already existing single. String is not copied - use apr_pstrdup() for that.

Parameters:

single Already exisiting single

str String to replace within single

Returns:

rxv_spin_data_t pointer, NULL on error

Example:

```
rxv_spin_single_set(single,"some new string");
```

Remarks:

This function is useful for replacing singles while iterating through rows of data. In other words, for data postprocessing.

5.1.4.4 rxv_spin_data_t* rxv_spin_single_mem (apr_pool_t * *pool*, const char * *str*, size_t *size*)

Create single data from from a counted string. Although the string is counted, it has to end with a nul character. This function will check for it and if it isn't there, it'll refuse to create the single. Keep in mind that the memory allocated for the counted string will be size + 1, meaning, if it isn't, the check for that nul character at the end may cause a segfault. String is not copied - use apr_pstrmemdup() for that.

Parameters:

pool Pool for allocation of data structure

str String to create data from

size Size of the string, not counting the ending nul character

Returns:

rxv_spin_data_t pointer, structure allocated from the pool, NULL on error

Example:

```
rxv_spin_single_mem(ctx->pool,"1234567890",10);
```

5.1.4.5 rxv_spin_data_t* rxv_spin_single_memset (rxv_spin_data_t * *single*, const char * *str*, size_t *size*)

Replace data (counted string) in an already existing single. Although the string is counted, it has to end with a nul character. This function will check for it and if it isn't there, it'll refuse to create the single. Keep in mind that the memory allocated for the counted string will be size + 1, meaning, if it isn't, the check for that nul character at the end may cause a segfault. String is not copied - use apr_pstrmemdup() for that.

Parameters:

single Already exisiting single

str String to replace within single

size Size of the string, not counting the ending nul character

Returns:

rxv_spin_data_t pointer, NULL on error

Example:

```
rxv_spin_single_memset(single,"1234567890",10);
```

Remarks:

This function is useful for replacing singles while iterating through rows of data. In other words, for data postprocessing.

5.1.4.6 rxv_spin_data_t* rxv_spin_single_tolower (rxv_spin_data_t * *single*)

Convert a single to lowercase. Single pushed into the function is modified, no copy is made.

Parameters:

single Single data

Returns:

the same single in lowercase, NULL on error

Example:

```
rxv_spin_single_tolower(single);
```

5.1.4.7 rxv_spin_data_t* rxv_spin_single_toupper (rxv_spin_data_t * *single*)

Convert a single to uppercase. Single pushed into the function is modified, no copy is made.

Parameters:

single Single data

Returns:

the same single in uppercase, NULL on error

Example:

```
rxv_spin_single_toupper(single);
```

5.1.4.8 rxv_spin_data_t* rxv_spin_single_trim (rxv_spin_data_t * *single*, unsigned char *what*)

Trim whitespace from a single. No copy is made.

Parameters:

single Single data

what Trim left: RXV_SPIN_TRIM_LEFT, trim right: RXV_SPIN_TRIM_RIGHT

Returns:

the same single trimmed, NULL on error

Example:

```
rxv_spin_single_trim(single,RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_RIGHT);
```

5.1.4.9 rxv_spin_data_t* rxv_spin_meta (apr_pool_t * *pool*, ...)

Create metadata from data. Meta data is an array of rxv_spin_data_t. It is not used to place data into the context, but simply to facilitate the API. Data is not copied.

Parameters:

pool Pool for allocation of data structure

... List of rxv_spin_data_t*, ending with NULL

Returns:

rxv_spin_data_t pointer, structures allocated from the pool, NULL on error

Example:

```
rxv_spin_meta(ctx->pool,
              rxv_spin_rows(ctx->pool,"x",x,"y",y,"z",z,NULL),
              rxv_spin_rows(ctx->pool,"u",u,"v",v,"w",w,NULL),
              NULL);
```

5.1.4.10 rxv_spin_data_t* rxv_spin_meta_vstr (apr_pool_t * *pool*, ...)

Create metadata from from nul terminated strings. Meta data is an array of rxv_spin_data_t. It is not used to place data into the context, but simply to facilitate the API. Strings are not copied - use apr_pstrdup() for that.

Parameters:

pool Pool for allocation of data structure

... List of char*, ending with NULL

Returns:

rxv_spin_data_t pointer, structures allocated from the pool, NULL on error

Example:

```
rxv_spin_meta_vstr(ctx->pool,"first string","second string",NULL);
```

5.1.4.11 rxv_spin_data_t* rxv_spin_meta_parse (apr_pool_t * *pool*, char * *str*, const char * *sep*)

Create metadata by parsing a string. Meta data is an array of rxv_spin_data_t. It is not used to place data into the context, but simply to facilitate the API. String to parse is NOT copied into pool storage before parsing, therefore, if you want to parse constant strings, you MUST copy them using apr_pstrdup(). This function actually chops up the string that is passed into it, so if the string is supposed to be used somewhere else, you may get a nasty surprise after calling this function. It is safer to make a copy.

Parameters:

pool Pool for allocation of data structure

str String to parse

sep Separators to use when parsing

Returns:

rxv_spin_data_t pointer, structures allocated from the pool, NULL on error

Example:

```
rxv_spin_meta_parse(ctx->pool,
                    apr_pstrdup(ctx->pool,"first str,second str/third str"),
                    ",/");
```

5.1.4.12 rxv_spin_data_t* rxv_spin_meta_empty (apr_pool_t * *pool*, size_t *size*)

Create an empty column of data. The size of the column is usually picked up from an existing rows data type. The returned value is metadata and its size is embedded in it.

Parameters:

pool Pool for allocation of data structures

size Size (number of elements) of the column to create

Returns:

rxv_spin_data_t pointer, structures allocated from the pool, NULL on error

Example:

```
rxv_spin_meta_empty(ctx->pool,rows->size);
```

Remarks:

This function is useful for creating boilerplate columns, used for various marker data.

5.1.4.13 apr_hash_t* rxv_spin_meta_hash (apr_pool_t * *pool*, rxv_spin_data_t * *data*)

Create a hash table of metadata. Keys will be strings from singles, values will be pointers to the actual singles. Data other than singles will not be hashed.

Parameters:

pool Pool for all memory allocation

data Meta data to hash

Returns:

a hash table of the data, NULL on error

Example:

```
rxv_spin_meta_hash(ctx->pool,data);
```

5.1.4.14 rxv_spin_data_t* rxv_spin_meta_mark (rxv_spin_data_t * *data*, size_t *element*)

Mark an element in the metadata (associate an empty string to it). This is useful for creating markers for the presentation layer. The data passed into this function is metadata, meaning, the size is embedded in it.

Parameters:

data Data to mark

element The index of the element to mark

Returns:

rxv_spin_data_t pointer to data, NULL on error

Example:

```
rxv_spin_meta_mark(data,rows->size-1);
```

5.1.4.15 rxv_spin_data_t* rxv_spin_meta_markeach (rxv_spin_data_t * *data*, size_t *off*, size_t *step*)

Mark each n-th element in the metadata (associate an empty string to it). This is useful for creating markers for the presentation layer. The data passed into this function metadata, meaning, the size is embedded in it.

Parameters:

data Data to mark

off Offset to start from

step Each n-th element to mark

Returns:

rxv_spin_data_t pointer to data, NULL on error

Example:

```
rxv_spin_meta_markeach(data,1,2);
```

5.1.4.16 rxv_spin_data_t* rxv_spin_meta_select (rxv_spin_data_t * *data*, rxv_spin_data_t * *select*, apr_hash_t * *hash*)

Select (mark) data that matches given data set. Selection data set is either single or metadata. The select operation can use an optional hash of the data that is being matched. Otherwise, the search is linear (which is OK for small data sets).

Parameters:

data Data to select

select Single or metadata to use for selection

hash Optional hash of the data being matched or NULL

Returns:

rxv_spin_data_t pointer to data, NULL on error

Example:

```
rxv_spin_meta_select(data,rxv_spin_str_parse(ctx->pool,"Australia,Japan",","),
                     rxv_spin_meta_hash(ctx->pool,data));
```

5.1.4.17 rxv_spin_data_t* rxv_spin_rows (apr_pool_t * *pool*, ...)

Create rows of data from metadata. The list supplied contains alternating column names, which are strings, and the actual data for the column, in form of metadata. If metadata contains different array lengths, the smallest size will be used. Data is not copied.

Parameters:

pool Pool for allocation of data structure

... List of char* and rxv_spin_data_t*, ending with NULL

Returns:

rxv_spin_data_t pointer, structures allocated from the pool, NULL on error

Example:

```
rxv_spin_rows(ctx->pool,"def",def,"ghi",ghi,"xyz",xyz,NULL);
```

5.1.4.18 apr_hash_t* rxv_spin_rows_hash (apr_pool_t * *pool*, rxv_spin_data_t * *rows*, const char * *column*)

Create a hash table of a column of rows. Keys will be strings from singles, values will be pointers to the actual singles. Data other than singles will not be hashed.

Parameters:

pool Pool for all memory allocation

rows Rows to hash

column Column to hash

Returns:

a hash table of the data, NULL on error

Example:

```
rxv_spin_rows_hash(ctx->pool,rows,"secondcolumn");
```

5.1.4.19 rxv_spin_data_t* rxv_spin_rows_mark (rxv_spin_data_t * *rows*, const char * *column*, size_t *element*)

Mark an element in the column of rows (associate an empty string to it). This is useful for creating markers for the presentation layer.

Parameters:

rows Rows to mark

column Name of the column to mark

element The index of the element to mark

Returns:

rxv_spin_data_t pointer to rows, NULL on error

Example:

```
rxv_spin_rows_mark(rows,"lastrow",rows->size-1);
```

5.1.4.20 rxv_spin_data_t* rxv_spin_rows_markeach (rxv_spin_data_t * *rows*, const char * *column*, size_t *off*, size_t *step*)

Mark each n-th element in the column of rows (associate an empty string to it). This is useful for creating markers for the presentation layer.

Parameters:

rows Rows to mark

column Name of the column to mark

off Offset to start from

step Each n-th element to mark

Returns:

rxv_spin_data_t pointer to rows, NULL on error

Example:

```
rxv_spin_rows_markeach(rows,"evenrow",0,2);
```

5.1.4.21 rxv_spin_data_t* rxv_spin_rows_select (rxv_spin_data_t * *rows*, rxv_spin_data_t * *select*, const char * *column*, const char * *marker*, apr_hash_t * *hash*)

Select (mark) rows of data that match given data set. Selection data set is either single or metadata. The select operation can use an optional hash of the column that is being matched. Otherwise, the search is linear (which is OK for small data sets).

Parameters:

rows Rows of data to select

select Single or metadata to use for selection

column Column used to match the data from select parameter

marker Column used to place markers for selected data

hash Optional hash of the column being matched or NULL

Returns:

rxv_spin_data_t pointer to rows, NULL on error

Example:

```
rxv_spin_rows_select(rows,rxv_spin_str_parse(ctx->pool,"Australia,Japan",","),
                     "country","selected",
                     rxv_spin_rows_hash(ctx->pool,rows,"country"));
```

5.1.4.22 rxv_spin_data_t* rxv_spin_column_get (apr_pool_t * *pool*, rxv_spin_data_t * *rows*, const char * *key*)

Get a column from rows data type. The column is not copied, rather a pointer to the first element in the array is returned inside a metadata type. Being metadata, this column will have its size embedded in it.

Parameters:

pool Pool for allocation of data structure

rows Rows data type

key Name of the column to get

Returns:

rxv_spin_data_t pointer, NULL on error

Example:

```
column=rxv_spin_column_get(rows,"firstfield");
for(i=0;i<rows->size;i++)
  rxv_spin_single_toupper(column+i);
```

5.1.4.23 rxv_spin_data_t* rxv_spin_column_set (rxv_spin_data_t * *rows*, const char * *key*, rxv_spin_data_t * *column*)

Set a column in rows data type. The column passed into this function is metadata. The size of the column has to be equal of greater than the number of rows.

Parameters:

rows Rows data type

key Name of the column to set

column Column to set

Returns:

rxv_spin_data_t pointer to the column, NULL on error

Example:

```
rxv_spin_column_set(rows,"newcolumn",rxv_spin_column(rows->size));
```

Remarks:

This function is also used for column removal, through a macro. In such a case, NULL return value is not necessarily a failure.

5.1.4.24 rxv_spin_data_t* rxv_spin_resize (apr_pool_t * *pool*, rxv_spin_data_t * *data*, size_t *size*)

Resize rows or metadata.

Parameters:

pool Pool for all memory allocation

data Data, rows or metadata type only

size New size

Returns:

resized data, NULL on error

Example:

```
rxv_spin_resize(ctx->pool,data,100);
```

Remarks:

If the requested size is smaller than original size, only the size field will be adjusted. If the requested size is larger, the structures containing pointers to data will be copied to the newly allocated memory space. If you rely on the old data (i.e. before resizing) in any way, you might create big problems, even security related ones.

5.1.4.25 rxv_spin_data_t* rxv_spin_copy (apr_pool_t * *pool*, rxv_spin_data_t * *data*)

Make a copy of data. The result is a full, deep copy (i.e. there is not a single byte in common with the original).

Parameters:

pool Pool for all memory allocation

data Data, any type

Returns:

copy of the data, NULL on error

Example:

```
rxv_spin_copy(ctx->pool,data);
```

Remarks:

If you're using this function on rows or metadata, the copying might take a while, as all memory is going to be allocated from the pool and all data and structures are going to be copied, down to the last string. In such a scenario, only use this function if you are planning on putting the result into context as a new item, or you need to keep the original data for some other reason.

5.1.4.26 char* rxv_spin_str_tolower (const char * *str*)

Convert a string to lowercase. String pushed into the function is modified, no copy is made.

Parameters:

str String to convert, nul terminated

Returns:

the same string in lowercase, NULL on error

Example:

```
rxv_spin_str_tolower(single);
```

5.1.4.27 char* rxv_spin_str_toupper (const char * *str*)

Convert a string to uppercase. String pushed into the function is modified, no copy is made.

Parameters:

str String to convert, nul terminated

Returns:

the same string in uppercase, NULL on error

Example:

```
rxv_spin_str_toupper(single);
```

5.1.4.28 char* rxv_spin_str_trim (char * *str*, unsigned char *what*)

Trim whitespace from a string. No copy is made. You MUST copy the string into pool storage using apr_pstrdup() if you're trimming constant strings.

Parameters:

str String

what Trim left: RXV_SPIN_TRIM_LEFT, trim right: RXV_SPIN_TRIM_RIGHT

Returns:

the same string trimmed, NULL on error

Example:

```
rxv_spin_str_trim(apr_strdup(ctx->pool,"  string with whitespace   ",
                  RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_RIGHT);
```

5.2 Context functions

Data Structures

- struct rxv_spin_context

  Context structure.

Defines

- #define rxv_spin_ctx_strget(ctx, key) rxv_spin_single_get(rxv_spin_ctx_get((ctx),(key)))
- #define rxv_spin_ctx_strset(ctx, key, val) rxv_spin_ctx_set((ctx),(key),rxv_spin_single((ctx) → pool,(val)))
- #define rxv_spin_ctx_del(ctx, key) rxv_spin_ctx_set((ctx),(key),NULL)

Typedefs

- typedef rxv_spin_context rxv_spin_context_t
- typedef rxv_spin_context_t rxv_spin_ctx_t

Functions

- rxv_spin_data_t * rxv_spin_ctx_get (rxv_spin_context_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ctx_set (rxv_spin_context_t *ctx, const char *key, rxv_spin_data_t *value)

5.2.1 Detailed Description

Context functions (mod_spin API)

5.2.2 Define Documentation

5.2.2.1 #define rxv_spin_ctx_strget(ctx, key) rxv_spin_single_get(rxv_spin_ctx_get((ctx),(key)))

get a string from the context by converting from single

5.2.2.2 #define rxv_spin_ctx_strset(ctx, key, val) rxv_spin_ctx_set((ctx),(key),rxv_spin_single((ctx) → pool,(val)))

set a string into context by converting to single

5.2.2.3 #define rxv_spin_ctx_del(ctx, key) rxv_spin_ctx_set((ctx),(key),NULL)

delete a value from context

5.2.3 Typedef Documentation

5.2.3.1 typedef struct rxv_spin_context rxv_spin_context_t

context type

5.2.3.2 typedef rxv_spin_context_t rxv_spin_ctx_t

shorthand for context type

5.2.4 Function Documentation

5.2.4.1 rxv_spin_data_t* rxv_spin_ctx_get (rxv_spin_context_t * *ctx*, const char * *key*)

Retrieve data from the context.

Parameters:

ctx Context

key Unique key by which this data is identified

Returns:

pointer to data (no copying is done), NULL if not found

Example:

```
rxv_spin_ctx_get(ctx,"result");
```

5.2.4.2 rxv_spin_data_t* rxv_spin_ctx_set (rxv_spin_context_t * *ctx*, const char * *key*, rxv_spin_data_t * *value*)

Place data into the context.

Parameters:

ctx Context

key Unique key by which this data is identified

value The actual data

Returns:

pointer to data (no copying is done), NULL on error

Example:

```
rxv_spin_ctx_set(ctx,"result",result->data);
```

Remarks:

This function is also used for context data removal, through a macro. In such a case, NULL return value is not necessarily a failure.

IMPORTANT:

The data set into the context has to have at least the lifetime of the request. Anything less and the pool used to hold the data may get cleaned before the bucket holding the value gets created. In most cases, this will result in a segfault.

IMPORTANT:

Do not put data allocated from the heap (with malloc(), calloc() and friends) into the context unless you set up cleanup functions that will free the allocated space. Although the context itself and the bucket brigade passed to Apache will be cleaned automatically (because it is allocated from the pool and/or has relevant cleanup functions), the heap storage space has to be freed explicitly. The safest option is to always use data allocated from the request pool.

5.3 Connection functions

Data Structures

- struct rxv_spin_conn

  Connection structure.

- struct rxv_spin_cpool

  Connection pool structure.

Defines

- #define RXV_SPIN_CONN_PGSQL 0x01
- #define RXV_SPIN_CONN_MYSQL 0x02
- #define RXV_SPIN_CONN_MINID 0x01
- #define RXV_SPIN_CONN_MAXID 0x3F
- #define RXV_SPIN_CONN_FOREIGN 0x40
- #define RXV_SPIN_CONN_POOLED 0x80
- #define RXV_SPIN_CONN_IS_PGSQL(c) ((((c) → type)&RXV_SPIN_CONN_-MAXID)==RXV_SPIN_CONN_PGSQL)

- #define RXV_SPIN_CONN_IS_MYSQL(c) ((((c) → type)&RXV_SPIN_CONN_-MAXID)==RXV_SPIN_CONN_MYSQL)
- #define RXV_SPIN_CONN_IS_FOREIGN(c) (((c) → type)&RXV_SPIN_CONN_FOREIGN)
- #define RXV_SPIN_CONN_IS_POOLED(c) (((c) → type)&RXV_SPIN_CONN_POOLED)

Typedefs

- typedef rxv_spin_conn rxv_spin_conn_t
- typedef rxv_spin_cpool rxv_spin_cpool_t

Functions

- rxv_spin_cpool_t * rxv_spin_cpool_create (apr_pool_t *pool)
- rxv_spin_conn_t * rxv_spin_cpool_get (apr_pool_t *pool, rxv_spin_cpool_t *cpool, const char *conninfo)
- rxv_spin_conn_t * rxv_spin_cpool_set (rxv_spin_cpool_t *cpool, const char *conninfo, void *conn, apr_status_t(*cleanup)(void *data))
- apr_status_t rxv_spin_conn_close (rxv_spin_conn_t *conn)

5.3.1 Detailed Description

Connection functions (mod_spin API)

5.3.2 Define Documentation

5.3.2.1 #define RXV_SPIN_CONN_PGSQL 0x01

PostgreSQL connection type

5.3.2.2 #define RXV_SPIN_CONN_MYSQL 0x02

MySQL connection type

5.3.2.3 #define RXV_SPIN_CONN_MINID 0x01

lowest known connection type

5.3.2.4 #define RXV_SPIN_CONN_MAXID 0x3F

highest known connection type

5.3.2.5 #define RXV_SPIN_CONN_FOREIGN 0x40

foreign connection type

5.3.2.6 #define RXV_SPIN_CONN_POOLED 0x80

connection pooled flag

5.3.2.7 #define RXV_SPIN_CONN_IS_PGSQL(c) ((((c) → type)&RXV_SPIN_CONN_-MAXID)==RXV_SPIN_CONN_PGSQL)

check if this connection is of PostgreSQL type

5.3.2.8 #define RXV_SPIN_CONN_IS_MYSQL(c) ((((c) → type)&RXV_SPIN_CONN_-MAXID)==RXV_SPIN_CONN_MYSQL)

check if this connection is of MySQL type

5.3.2.9 #define RXV_SPIN_CONN_IS_FOREIGN(c) (((c) → type)&RXV_SPIN_CONN_-FOREIGN)

check if this connection is of MySQL type

5.3.2.10 #define RXV_SPIN_CONN_IS_POOLED(c) (((c) → type)&RXV_SPIN_CONN_-POOLED)

check if this connection is pooled

5.3.3 Typedef Documentation

5.3.3.1 typedef struct rxv_spin_conn rxv_spin_conn_t

connection type

5.3.3.2 typedef struct rxv_spin_cpool rxv_spin_cpool_t

connection pool type

5.3.4 Function Documentation

5.3.4.1 rxv_spin_cpool_t* rxv_spin_cpool_create (apr_pool_t * *pool*)

Create a connection pool. Connection strings used to search for open connections within the pool are treated as case sensitive and they consist of the actual connection string preceded by the type (a single character).

Parameters:

pool Memory pool used for allocation

Returns:

pointer to the new connection pool, NULL on error

Example:

```
cpool=rxv_spin_cpool_create(p);
```

Remarks:

This function is not normally called from the application, even when connection pools are used. Connection pool creation and cleanup is performed automatically by mod_spin. If you decide to manage you own connection pools, you will need this function.

IMPORTANT:

The code of mod_spin calls this function to create one connection pool for each thread of Apache. If you call apr_proc_create() to run a new process from within any mod_spin applications, registered child cleanup function will close all connections and remove them from the pool before the new process is executed inside the forked one. This equally applies to the connection pools created by you.

5.3.4.2 rxv_spin_conn_t* rxv_spin_cpool_get (apr_pool_t * *pool*, rxv_spin_cpool_t * *cpool*, const char * *conninfo*)

Get a registered connection of foreign type from the connection pool.

Parameters:

pool Memory pool for temporary allocations

cpool Connection pool

conninfo Connection string for this connection

Returns:

Pointer to the connection or NULL if not found

Example:

```
rxv_spin_cpool_get(ctx->pool,ctx->cpool,
                   "ldap://ldap.example.com/dc=example,dc=com");
```

5.3.4.3 rxv_spin_conn_t* rxv_spin_cpool_set (rxv_spin_cpool_t * *cpool*, const char * *conninfo*, void * *conn*, apr_status_t(*)(void *data) *cleanup*)

Register a connection of foreign type with the connection pool.

Parameters:

cpool Connection pool

conninfo Connection string for this connection

conn Connection pointer

cleanup Cleanup function to call on pool destruction

Returns:

Pointer to the connection, NULL on error

Example:

```
rxv_spin_cpool_set(ctx->cpool,"ldap://ldap.example.com/dc=example,dc=com",
                   ldapconn,ldap_cleanup);
```

Remarks:

This function will register the connection even if the connection with the same key already exists. This may lead to multiple cleanup functions being registered for the same connection and eventually segfaults. Use rxv_spin_cpool_get() function to verify if the connection was registered before.

IMPORTANT:

The cleanup function MUST remove the connection from the hash table holding all pooled connections. Failing that, the next time such connection is requested, an invalid connection structure will be passed to the application, causing an almost certain segfault. See db_clean() function in mod_spin source (file db.c) for an example.

IMPORTANT:

The lifetime of the connection and all other variables passed into this function has to be at least the lifetime of the connection pool you are registering the connection with. Normally, connection pools have thread lifetime, so if you have shorter lifetime for your connection (e.g. request, connection), you are setting yourself up for segfaults. Also, if the lifetime of the connection is longer than the one of the connection pool, for instance, if it has the lifetime of the process and you are registering it with the regular thread based connection pool, you may experience segfaults if you rely on that connection once the thread exited. If you are managing your own connection pool, same issues apply, except for the fact that you control the lifetime of the connection pool, rather than mod_spin.

5.3.4.4 apr_status_t rxv_spin_conn_close (rxv_spin_conn_t * *conn*)

Close a connection and remove it from the pool.

Parameters:

conn Connection

Returns:

APR_SUCCESS on success, otherwise an error

Example:

```
rxv_spin_conn_close(conn);
```

Remarks:

This function may be useful for explicit termination of pooled or non-pooled connections, especially if you're managing your own connection pools. If pooled connections aren't used, the connection will be closed anyway during the temporary pool cleanup (first argument to rxv_spin_db_connect()).

5.4 Database functions

Data Structures

- struct rxv_spin_db_result

  Database result structure.

Defines

- #define RXV_SPIN_DB_INFO_DB 0x01
- #define RXV_SPIN_DB_INFO_USER 0x02
- #define RXV_SPIN_DB_INFO_PASS 0x03
- #define RXV_SPIN_DB_INFO_HOST 0x04

- #define RXV_SPIN_DB_INFO_PORT 0x05
- #define RXV_SPIN_DB_INFO_TTY 0x06
- #define RXV_SPIN_DB_INFO_OPTIONS 0x07
- #define RXV_SPIN_DB_INFO_ERROR 0x08
- #define rxv_spin_db_db(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_DB)
- #define rxv_spin_db_user(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_USER)
- #define rxv_spin_db_pass(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_PASS)
- #define rxv_spin_db_host(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_HOST)
- #define rxv_spin_db_port(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_PORT)
- #define rxv_spin_db_tty(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_TTY)
- #define rxv_spin_db_options(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_OPTIONS)
- #define rxv_spin_db_error(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_ERROR)

Typedefs

- typedef rxv_spin_db_result rxv_spin_db_result_t

Functions

- rxv_spin_conn_t * rxv_spin_db_connect (apr_pool_t *pool, rxv_spin_cpool_t *cpool, const char *conninfo, unsigned char type)
- rxv_spin_db_result_t * rxv_spin_db_exec (apr_pool_t *pool, rxv_spin_conn_t *conn, const char *query)
- char * rxv_spin_db_info (rxv_spin_conn_t *conn, unsigned char what)
- apr_status_t rxv_spin_db_status (rxv_spin_conn_t *conn)
- char * rxv_spin_db_escape (apr_pool_t *pool, rxv_spin_conn_t *conn, const char *str)
- apr_status_t rxv_spin_db_reset (rxv_spin_conn_t *conn)

5.4.1 Detailed Description

Database functions (mod_spin API)

5.4.2 Define Documentation

5.4.2.1 #define RXV_SPIN_DB_INFO_DB 0x01

database name

5.4.2.2 #define RXV_SPIN_DB_INFO_USER 0x02

database user

5.4.2.3 #define RXV_SPIN_DB_INFO_PASS 0x03

database password

5.4.2.4 #define RXV_SPIN_DB_INFO_HOST 0x04

database host

5.4.2.5 #define RXV_SPIN_DB_INFO_PORT 0x05

database port

5.4.2.6 #define RXV_SPIN_DB_INFO_TTY 0x06

database tty

5.4.2.7 #define RXV_SPIN_DB_INFO_OPTIONS 0x07

database options

5.4.2.8 #define RXV_SPIN_DB_INFO_ERROR 0x08

database error

5.4.2.9 #define rxv_spin_db_db(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_DB)

get name

5.4.2.10 #define rxv_spin_db_user(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_USER)

get user

5.4.2.11 #define rxv_spin_db_pass(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_PASS)

get password

5.4.2.12 #define rxv_spin_db_host(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_HOST)

get host

5.4.2.13 #define rxv_spin_db_port(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_PORT)

get port

5.4.2.14 #define rxv_spin_db_tty(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_TTY)

get tty

5.4.2.15 #define rxv_spin_db_options(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_OPTIONS)

get options

5.4.2.16 #define rxv_spin_db_error(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_ERROR)

get error

5.4.3 Typedef Documentation

5.4.3.1 typedef struct rxv_spin_db_result rxv_spin_db_result_t

database query result type

5.4.4 Function Documentation

5.4.4.1 rxv_spin_conn_t* rxv_spin_db_connect (apr_pool_t * *pool*, rxv_spin_cpool_t * *cpool*, const char * *conninfo*, unsigned char *type*)

Connect to a database and optionally pool the connection.

Parameters:

pool Memory pool used for memory allocations

cpool Connection pool

conninfo Connection string

type Database type

Returns:

pointer to a database connection, NULL on error

Example:

```
rxv_spin_db_connect(ctx->pool,NULL,"dbname=spintest",RXV_SPIN_DB_PGSQL);

rxv_spin_db_connect(ctx->pool,ctx->cpool,"dbname=spintest",RXV_SPIN_DB_MYSQL);
```

Remarks:

If connection pooling isn't used, the cpool parameter has to be set to NULL. The pool parameter has to point to a valid APR memory pool. If pooled connections aren't used, the connection will be closed on pool (the first argument) cleanup. Meaning, if you want custom lifetime of the connection, create your own memory pool and then destroy it when you don't need the database connection any more.

IMPORTANT:

Alywas use relevant macros (such as RXV_SPIN_CONN_IS_PGSQL(c)) to enquire about the type of the connection, because the type is an unsigned char that is a combination of various binary data, including the database type and connection pooled flag.

IMPORTANT:

This function takes connection string in the form defined by PostgreSQL manual. In fact, the code for the string parsing (for databases other than PostgreSQL) has been directly lifted from PostgreSQL version 7.4.1. However, when used with database types other than PostgreSQL, not all keywords are supported. Namely, for use with MySQL, you can only use "host", "port", "dbname", "user" and "password". Anything else will cause errors.

5.4.4.2 rxv_spin_db_result_t* rxv_spin_db_exec (apr_pool_t * *pool*, rxv_spin_conn_t * *conn*, const char * *query*)

Execute a database query of any type.

Parameters:

pool Pool used for memory allocation

conn Database connection

query SQL query to be performed

Returns:

Valid rxv_spin_db_result_t pointer, NULL on error

Example:

```
result=rxv_spin_db_exec(ctx->pool,conn,"select * from spintest");
```

Remarks:

Depending on the nature of the query, result->data may be NULL. Only SELECT and similar queries that return tuples (rows) will have this member non-NULL. To verify if the query finished successfully, use result->status.

5.4.4.3 char* rxv_spin_db_info (rxv_spin_conn_t * *conn*, unsigned char *what*)

Get various info about the database connection.

Parameters:

conn Database connection

what The type of information to get

Returns:

string with the desired info, NULL on error

Example:

```
rxv_spin_db_info(conn);
```

Remarks:

You can use either this function or defined macros to get specific information about this database connection. Note that some database APIs won't provide all information. In that case, an empty string might be returned.

5.4.4.4 apr_status_t rxv_spin_db_status (rxv_spin_conn_t * *conn*)

Get the status of the connection.

Parameters:

conn Database connection

Returns:

APR_SUCCESS if OK, otherwise an error

Example:

```
rxv_spin_db_status(conn);
```

5.4.4.5 char* rxv_spin_db_escape (apr_pool_t * *pool*, rxv_spin_conn_t * *conn*, const char * *str*)

Escape a string for use within an SQL query.

Parameters:

pool Pool used for memory allocation

conn Database connection

str string to be escaped

Returns:

Pointer to escaped string, NULL on error

Example:

```
escstr=rxv_spin_db_escape(ctx->pool,conn,"'A string to escape'");
```

Remarks:

The connection pointer is used to determine database type and therefore use the correct, database specific escaping function. MySQL needs a pointer to the connection in order to determine the correct character set. Therefore the second argument must be a valid connection pointer. The underlying database specific functions have slightly different behaviour, depending on the database being used. Check in the relevant manual for differences.

5.4.4.6 apr_status_t rxv_spin_db_reset (rxv_spin_conn_t * *conn*)

Reset a database connection (i.e. attempt to reconnect).

Parameters:

conn Database connection to reset

Returns:

APR_SUCCESS on success, otherwise an error

Example:

```
rxv_spin_db_reset(conn);
```

Remarks:

This function is not normally called from the application. Reset is performed automatically by mod_spin if the connection is down. However, you may choose to use this function in same rare cases.

5.5 Application and session functions

Defines

- #define rxv_spin_app_strget(ctx, key) rxv_spin_single_get(rxv_spin_app_get((ctx),(key)))
- #define rxv_spin_app_strset(ctx, key, val) rxv_spin_app_set((ctx),(key),rxv_spin_single((ctx) → pool,(val)))
- #define rxv_spin_ses_strget(ctx, key) rxv_spin_single_get(rxv_spin_ses_get((ctx),(key)))
- #define rxv_spin_ses_strset(ctx, key, val) rxv_spin_ses_set((ctx),(key),rxv_spin_single((ctx) → pool,(val)))

Functions

- rxv_spin_data_t * rxv_spin_app_get (rxv_spin_context_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_app_set (rxv_spin_context_t *ctx, const char *key, rxv_spin_data_t *val)
- apr_status_t rxv_spin_app_del (rxv_spin_context_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ses_get (rxv_spin_context_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ses_set (rxv_spin_context_t *ctx, const char *key, rxv_spin_data_t *val)
- apr_status_t rxv_spin_ses_del (rxv_spin_context_t *ctx, const char *key)
- char * rxv_spin_ses_idget (rxv_spin_context_t *ctx)
- int rxv_spin_ses_valid (rxv_spin_context_t *ctx)

5.5.1 Detailed Description

Application and session functions (mod_spin API)

5.5.2 Define Documentation

5.5.2.1 #define rxv_spin_app_strget(ctx, key) rxv_spin_single_get(rxv_spin_app_get((ctx),(key)))

get a string from application instead of single

5.5.2.2 #define rxv_spin_app_strset(ctx, key, val) rxv_spin_app_set((ctx),(key),rxv_spin_single((ctx) → pool,(val)))

set string into application instead of single

5.5.2.3 #define rxv_spin_ses_strget(ctx, key) rxv_spin_single_get(rxv_spin_ses_get((ctx),(key)))

get a string from session instead of single

5.5.2.4 #define rxv_spin_ses_strset(ctx, key, val) rxv_spin_ses_set((ctx),(key),rxv_spin_single((ctx) → pool,(val)))

set string into application instead of single

5.5.3 Function Documentation

5.5.3.1 rxv_spin_data_t* rxv_spin_app_get (rxv_spin_context_t * *ctx*, const char * *key*)

Retrieve a value from the application database.

Parameters:

ctx Context

key Unique key by which this value is identified

Returns:

pointer to a copy of the value, NULL on error

Example:

```
rxv_spin_app_get(ctx,"some application key");
```

5.5.3.2 rxv_spin_data_t* rxv_spin_app_set (rxv_spin_context_t * *ctx*, const char * *key*, rxv_spin_data_t * *val*)

Put value in the application database.

Parameters:

ctx Context

key Unique key by which this value is identified

val Value to be placed in the database (single)

Returns:

pointer to the value, NULL on error

Example:

```
rxv_spin_app_set(ctx,"some application key",
                    rxv_spin_str_single(ctx->pool,"some application value"));
```

5.5.3.3 apr_status_t rxv_spin_app_del (rxv_spin_context_t * *ctx*, const char * *key*)

Delete a record in the application database.

Parameters:

ctx Context

key Unique key by which record is identified

Returns:

APR_SUCCESS on success, otherwise an error

Example:

```
rxv_spin_app_del(ctx,"some application key");
```

Remarks:

It is not an error to delete and non-existent record.

5.5.3.4 rxv_spin_data_t* rxv_spin_ses_get (rxv_spin_context_t * *ctx*, const char * *key*)

Retrieve a value from the session database.

Parameters:

ctx Context

key Unique key by which this value is identified

Returns:

pointer to a copy of the value, NULL on error

Example:

```
rxv_spin_app_get(ctx,"some session key");
```

5.5.3.5 rxv_spin_data_t* rxv_spin_ses_set (rxv_spin_context_t * *ctx*, const char * *key*, rxv_spin_data_t * *val*)

Put value in the session database.

Parameters:

ctx Context

key Unique key by which this value is identified

val Value to be placed in the database (single)

Returns:

pointer to a copy of the value, NULL on error

Example:

```
rxv_spin_app_set(ctx,"some session key",
                    rxv_spin_str_single(ctx->pool,"some session value"));
```

5.5.3.6 apr_status_t rxv_spin_ses_del (rxv_spin_context_t * *ctx*, const char * *key*)

Delete a record in the session database.

Parameters:

ctx Context

key Unique key by which record is identified

Returns:

APR_SUCCESS on success, otherwise an error

Example:

```
rxv_spin_ses_del(ctx,"some session key");
```

Remarks:

It is not an error to delete and non-existent record.

5.5.3.7 char* rxv_spin_ses_idget (rxv_spin_context_t * *ctx*)

Get session id

Parameters:

ctx Context

Returns:

Session id or NULL if no session support exists

Example:

```
rxv_spin_ses_idget(ctx);
```

Remarks:

This function fetches a string representation of the session id, which draws is origin from mod_unique_id. If mod_unique_id isn't present in Apache, this function returns NULL. Note that the fact that one can get a session id does not mean that there is a valid session in existence (that is to say, that the client accepted the session). Use rxv_spin_ses_valid() function to find that out.

5.5.3.8 int rxv_spin_ses_valid (rxv_spin_context_t * *ctx*)

Find out if the session is valid.

Parameters:

ctx Context

Returns:

1 if valid, otherwise 0

Example:

```
rxv_spin_ses_valid(ctx);
```

Remarks:

If the client accepted this session, this function returns 1. This will only be true if the client submitted a valid session id and its relevant hash to mod_spin. Also, the session will not be considered valid if there are path tricks in the session id (this is unlikely to happen, due to hash checking, but nevertheless).

5.6 Service function

Typedefs

- typedef int(*) rxv_spin_service_t (rxv_spin_context_t *context)

5.6.1 Detailed Description

Service function (mod_spin API)

5.6.2 Typedef Documentation

5.6.2.1 typedef int(*) rxv_spin_service_t(rxv_spin_context_t *context)

service function

6 mod_spin Data Structure Documentation

6.1 rxv_spin_conn Struct Reference

Connection structure.

`#include <rxv_spin.h>`

Data Fields

- unsigned char type
- char * cinfo
- union {
  PGconn * pgconn
  MYSQL * myconn
  void * conn
  };
- union {
  rxv_spin_cpool_t * cpool
  apr_pool_t * pool
  };
- apr_status_t(* cleanup)(void *data)

6.1.1 Detailed Description

Connection structure.

6.1.2 Field Documentation

6.1.2.1 unsigned char rxv_spin_conn::type

connection type: PostgreSQL, MySQL (also pooled or not) etc., use the macros to find out

6.1.2.2 char* rxv_spin_conn::cinfo

full connection identification string: type + connect string

6.1.2.3 PGconn* rxv_spin_conn::pgconn

PostgreSQL specific connection

6.1.2.4 MYSQL* rxv_spin_conn::myconn

MySQL specific connection

6.1.2.5 void* rxv_spin_conn::conn

generic or foreign connection

6.1.2.6 union { ... }

unnamed

6.1.2.7 rxv_spin_cpool_t* rxv_spin_conn::cpool

the pool of connections for this connection

6.1.2.8 apr_pool_t* rxv_spin_conn::pool

memory pool we registered the cleanup with, if pooled connections aren't used

6.1.2.9 union { ... }

unnamed

6.1.2.10 apr_status_t(* rxv_spin_conn::cleanup)(void *data)

cleanup function

The documentation for this struct was generated from the following file:

- spin/rxv_spin.h

6.2 rxv_spin_context Struct Reference

Context structure.

`#include <rxv_spin.h>`

Data Fields

- request_rec * r
- apr_pool_t * pool
- rxv_spin_data_t * data
- apreq_handle_t * req
- rxv_spin_cpool_t * cpool
- void * guts
- void * extra

6.2.1 Detailed Description

Context structure.

6.2.2 Field Documentation

6.2.2.1 request_rec* rxv_spin_context::r

raw request

6.2.2.2 apr_pool_t* rxv_spin_context::pool

memory pool used by the context

6.2.2.3 rxv_spin_data_t* rxv_spin_context::data

data produced by the application

6.2.2.4 apreq_handle_t* rxv_spin_context::req

apreq handle for cookies and parameters

6.2.2.5 rxv_spin_cpool_t* rxv_spin_context::cpool

pool of connections

6.2.2.6 void* rxv_spin_context::guts

private data used internally by mod_spin

6.2.2.7 void* rxv_spin_context::extra

data you may want to place in context

The documentation for this struct was generated from the following file:

- spin/rxv_spin.h

6.3 rxv_spin_cpool Struct Reference

Connection pool structure.

`#include <rxv_spin.h>`

Data Fields

- apr_pool_t * pool
- apr_hash_t * conns

6.3.1 Detailed Description

Connection pool structure.

6.3.2 Field Documentation

6.3.2.1 apr_pool_t* rxv_spin_cpool::pool

pool used for all allocations

6.3.2.2 apr_hash_t* rxv_spin_cpool::conns

hash table of connections

The documentation for this struct was generated from the following file:

- spin/rxv_spin.h

6.4 rxv_spin_data Struct Reference

Data structure.

`#include <rxv_spin.h>`

Data Fields

- unsigned char type
- size_t size
- union {
  apr_hash_t * cols
  char * data
  rxv_spin_data_t * meta
  void * both
  };

6.4.1 Detailed Description

Data structure.

6.4.2 Field Documentation

6.4.2.1 unsigned char rxv_spin_data::type

data type

6.4.2.2 size_t rxv_spin_data::size

string size or number of rows

6.4.2.3 apr_hash_t* rxv_spin_data::cols

columns of the data type rows

6.4.2.4 char* rxv_spin_data::data

actual data

6.4.2.5 rxv_spin_data_t* rxv_spin_data::meta

metadata (i.e. pointer to an array)

6.4.2.6 void* rxv_spin_data::both

when we need to test both cols & data at once

6.4.2.7 union { ... }

unnamed

The documentation for this struct was generated from the following file:

- spin/rxv_spin.h

6.5 rxv_spin_db_result Struct Reference

Database result structure.

`#include <rxv_spin.h>`

Data Fields

- apr_status_t status
- char * error
- rxv_spin_data_t * data

6.5.1 Detailed Description

Database result structure.

6.5.2 Field Documentation

6.5.2.1 apr_status_t rxv_spin_db_result::status

APR_SUCCESS when all is cool

6.5.2.2 char* rxv_spin_db_result::error

descriptive error or NULL

6.5.2.3 rxv_spin_data_t* rxv_spin_db_result::data

results returned or NULL

The documentation for this struct was generated from the following file:

- spin/rxv_spin.h

7 mod_spin File Documentation

7.1 spin/rxv_spin.h File Reference

mod_spin data structures and functions

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <stdarg.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <apr.h>
#include <apr_general.h>
#include <apr_strings.h>
#include <apr_lib.h>
#include <apr_pools.h>
#include <apr_buckets.h>
#include <apr_hash.h>
#include <apr_file_io.h>
#include <apr_portable.h>
#include <apr_time.h>
#include <apr_sdbm.h>
#include <apr_md5.h>
#include <apr_base64.h>
#include <apreq_param.h>
#include <apreq_cookie.h>
#include <apreq_module_apache2.h>
#include <http_request.h>
#include <libxml/parser.h>
#include <libxml/tree.h>
#include <libpq-fe.h>
#include <mysql.h>
```

Data Structures

- struct rxv_spin_data

  Data structure.

- struct rxv_spin_context

  Context structure.

- struct rxv_spin_conn

  Connection structure.

- struct rxv_spin_cpool

  Connection pool structure.

- struct rxv_spin_db_result

  Database result structure.

Defines

- #define RXV_SPIN_DATA_SGL 0x01
- #define RXV_SPIN_DATA_RWS 0x02
- #define RXV_SPIN_DATA_MTA 0xFF
- #define RXV_SPIN_TRIM_LEFT 0x01
- #define RXV_SPIN_TRIM_RIGHT 0x02
- #define rxv_spin_single_trimboth(s) rxv_spin_single_trim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_single_trimleft(s) rxv_spin_single_trim((s),(RXV_SPIN_TRIM_LEFT))
- #define rxv_spin_single_trimright(s) rxv_spin_single_trim((s),(RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_column_del(rows, key) rxv_spin_column_set((rows),(key),NULL)
- #define rxv_spin_str_trimboth(s) rxv_spin_str_trim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_str_trimleft(s) rxv_spin_str_trim((s),(RXV_SPIN_TRIM_LEFT))
- #define rxv_spin_str_trimright(s) rxv_spin_str_trim((s),(RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_ctx_strget(ctx, key) rxv_spin_single_get(rxv_spin_ctx_get((ctx),(key)))
- #define rxv_spin_ctx_strset(ctx, key, val) rxv_spin_ctx_set((ctx),(key),rxv_spin_single((ctx) → pool,(val)))
- #define rxv_spin_ctx_del(ctx, key) rxv_spin_ctx_set((ctx),(key),NULL)
- #define rxv_spin_app_strget(ctx, key) rxv_spin_single_get(rxv_spin_app_get((ctx),(key)))
- #define rxv_spin_app_strset(ctx, key, val) rxv_spin_app_set((ctx),(key),rxv_spin_single((ctx) → pool,(val)))
- #define rxv_spin_ses_strget(ctx, key) rxv_spin_single_get(rxv_spin_ses_get((ctx),(key)))
- #define rxv_spin_ses_strset(ctx, key, val) rxv_spin_ses_set((ctx),(key),rxv_spin_single((ctx) → pool,(val)))
- #define RXV_SPIN_CONN_PGSQL 0x01
- #define RXV_SPIN_CONN_MYSQL 0x02
- #define RXV_SPIN_CONN_MINID 0x01
- #define RXV_SPIN_CONN_MAXID 0x3F
- #define RXV_SPIN_CONN_FOREIGN 0x40
- #define RXV_SPIN_CONN_POOLED 0x80
- #define RXV_SPIN_CONN_IS_PGSQL(c) ((((c) → type)&RXV_SPIN_CONN_MAXID)==RXV_SPIN_CONN_PGSQL)
- #define RXV_SPIN_CONN_IS_MYSQL(c) ((((c) → type)&RXV_SPIN_CONN_MAXID)==RXV_SPIN_CONN_MYSQL)
- #define RXV_SPIN_CONN_IS_FOREIGN(c) (((c) → type)&RXV_SPIN_CONN_FOREIGN)
- #define RXV_SPIN_CONN_IS_POOLED(c) (((c) → type)&RXV_SPIN_CONN_POOLED)
- #define RXV_SPIN_DB_INFO_DB 0x01
- #define RXV_SPIN_DB_INFO_USER 0x02
- #define RXV_SPIN_DB_INFO_PASS 0x03
- #define RXV_SPIN_DB_INFO_HOST 0x04
- #define RXV_SPIN_DB_INFO_PORT 0x05

- #define RXV_SPIN_DB_INFO_TTY 0x06
- #define RXV_SPIN_DB_INFO_OPTIONS 0x07
- #define RXV_SPIN_DB_INFO_ERROR 0x08
- #define rxv_spin_db_db(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_DB)
- #define rxv_spin_db_user(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_USER)
- #define rxv_spin_db_pass(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_PASS)
- #define rxv_spin_db_host(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_HOST)
- #define rxv_spin_db_port(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_PORT)
- #define rxv_spin_db_tty(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_TTY)
- #define rxv_spin_db_options(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_OPTIONS)
- #define rxv_spin_db_error(conn) rxv_spin_db_info((conn),RXV_SPIN_DB_INFO_ERROR)

Typedefs

- typedef rxv_spin_data rxv_spin_data_t
- typedef rxv_spin_context rxv_spin_context_t
- typedef rxv_spin_context_t rxv_spin_ctx_t
- typedef rxv_spin_conn rxv_spin_conn_t
- typedef rxv_spin_cpool rxv_spin_cpool_t
- typedef rxv_spin_db_result rxv_spin_db_result_t
- typedef int(∗) rxv_spin_service_t (rxv_spin_context_t ∗context)

Functions

- rxv_spin_data_t ∗ rxv_spin_single (apr_pool_t ∗pool, const char ∗str)
- char ∗ rxv_spin_single_get (rxv_spin_data_t ∗single)
- rxv_spin_data_t ∗ rxv_spin_single_set (rxv_spin_data_t ∗single, const char ∗str)
- rxv_spin_data_t ∗ rxv_spin_single_mem (apr_pool_t ∗pool, const char ∗str, size_t size)
- rxv_spin_data_t ∗ rxv_spin_single_memset (rxv_spin_data_t ∗single, const char ∗str, size_t size)
- rxv_spin_data_t ∗ rxv_spin_single_tolower (rxv_spin_data_t ∗single)
- rxv_spin_data_t ∗ rxv_spin_single_toupper (rxv_spin_data_t ∗single)
- rxv_spin_data_t ∗ rxv_spin_single_trim (rxv_spin_data_t ∗single, unsigned char what)
- rxv_spin_data_t ∗ rxv_spin_meta (apr_pool_t ∗pool,...)
- rxv_spin_data_t ∗ rxv_spin_meta_vstr (apr_pool_t ∗pool,...)
- rxv_spin_data_t ∗ rxv_spin_meta_parse (apr_pool_t ∗pool, char ∗str, const char ∗sep)
- rxv_spin_data_t ∗ rxv_spin_meta_empty (apr_pool_t ∗pool, size_t size)
- apr_hash_t ∗ rxv_spin_meta_hash (apr_pool_t ∗pool, rxv_spin_data_t ∗data)
- rxv_spin_data_t ∗ rxv_spin_meta_mark (rxv_spin_data_t ∗data, size_t element)
- rxv_spin_data_t ∗ rxv_spin_meta_markeach (rxv_spin_data_t ∗data, size_t off, size_t step)
- rxv_spin_data_t ∗ rxv_spin_meta_select (rxv_spin_data_t ∗data, rxv_spin_data_t ∗select, apr_hash_t ∗hash)
- rxv_spin_data_t ∗ rxv_spin_rows (apr_pool_t ∗pool,...)
- apr_hash_t ∗ rxv_spin_rows_hash (apr_pool_t ∗pool, rxv_spin_data_t ∗rows, const char ∗column)
- rxv_spin_data_t ∗ rxv_spin_rows_mark (rxv_spin_data_t ∗rows, const char ∗column, size_t element)
- rxv_spin_data_t ∗ rxv_spin_rows_markeach (rxv_spin_data_t ∗rows, const char ∗column, size_t off, size_t step)
- rxv_spin_data_t ∗ rxv_spin_rows_select (rxv_spin_data_t ∗rows, rxv_spin_data_t ∗select, const char ∗column, const char ∗marker, apr_hash_t ∗hash)
- rxv_spin_data_t ∗ rxv_spin_column_get (apr_pool_t ∗pool, rxv_spin_data_t ∗rows, const char ∗key)

- rxv_spin_data_t * rxv_spin_column_set (rxv_spin_data_t *rows, const char *key, rxv_spin_data_t *column)
- rxv_spin_data_t * rxv_spin_resize (apr_pool_t *pool, rxv_spin_data_t *data, size_t size)
- rxv_spin_data_t * rxv_spin_copy (apr_pool_t *pool, rxv_spin_data_t *data)
- char * rxv_spin_str_tolower (const char *str)
- char * rxv_spin_str_toupper (const char *str)
- char * rxv_spin_str_trim (char *str, unsigned char what)
- rxv_spin_data_t * rxv_spin_ctx_get (rxv_spin_context_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ctx_set (rxv_spin_context_t *ctx, const char *key, rxv_spin_data_t *value)
- rxv_spin_data_t * rxv_spin_app_get (rxv_spin_context_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_app_set (rxv_spin_context_t *ctx, const char *key, rxv_spin_data_t *val)
- apr_status_t rxv_spin_app_del (rxv_spin_context_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ses_get (rxv_spin_context_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ses_set (rxv_spin_context_t *ctx, const char *key, rxv_spin_data_t *val)
- apr_status_t rxv_spin_ses_del (rxv_spin_context_t *ctx, const char *key)
- char * rxv_spin_ses_idget (rxv_spin_context_t *ctx)
- int rxv_spin_ses_valid (rxv_spin_context_t *ctx)
- rxv_spin_conn_t * rxv_spin_db_connect (apr_pool_t *pool, rxv_spin_cpool_t *cpool, const char *conninfo, unsigned char type)
- rxv_spin_db_result_t * rxv_spin_db_exec (apr_pool_t *pool, rxv_spin_conn_t *conn, const char *query)
- char * rxv_spin_db_info (rxv_spin_conn_t *conn, unsigned char what)
- apr_status_t rxv_spin_db_status (rxv_spin_conn_t *conn)
- char * rxv_spin_db_escape (apr_pool_t *pool, rxv_spin_conn_t *conn, const char *str)
- apr_status_t rxv_spin_db_reset (rxv_spin_conn_t *conn)
- rxv_spin_cpool_t * rxv_spin_cpool_create (apr_pool_t *pool)
- rxv_spin_conn_t * rxv_spin_cpool_get (apr_pool_t *pool, rxv_spin_cpool_t *cpool, const char *conninfo)
- rxv_spin_conn_t * rxv_spin_cpool_set (rxv_spin_cpool_t *cpool, const char *conninfo, void *conn, apr_status_t(*cleanup)(void *data))
- apr_status_t rxv_spin_conn_close (rxv_spin_conn_t *conn)

7.1.1 Detailed Description

mod_spin data structures and functions

Index