mod_spin

1.2.0

Generated by Doxygen 1.6.1

Tue Dec 29 12:46:33 2009

Contents

1 Main Page

Introduction

See also:

Installation News Licence

```
Welcome to mod_spin for Apache 2.2+ from Rexursive(R)
=====================================================

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 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 via APR DBD

mod_spin is written in C, just like Apache 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. It should be particularly easy to do
that on systems that run RPM packaging system, such as Fedora, 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 a content handler and/or a filter, meaning, for a specified file
extension, mod_spin will read the file (or other input), 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.

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 functions are
```

called, one when the library is dynamically linked into Apache, one during the fixup phase of the request and one during the handler phase. These functions take one argument, which is a structure containing the context. The context holds the data to be replaced, 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 functions are 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 an SQL database or XML files). Apache Portable Runtime DBD layer is used to provide access to various SQL backends, like PostgreSQL, MySQL, SQLite2/3, Oracle etc.

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 sections "Service function", "Prepare function" and "Init function" for all the details related to the entry points 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. Also, technologies like GCC's stack protector, kernel's exec-shield and NX bits may prove useful.

Here are some very important security implications that may arise from the use of mod_spin. This is in relation to connection pools. To understand the issues involved, one needs to understand how Apache deals with multiple client connections.

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!

General recommendation is: if you don't have full control over all applications and you're using session/application, SQL/XML based, 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 that 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 management
==========

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. 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, template cache and connection pools will be associated with the process pool (i.e. objects from the connection pool may be pointing to objects from the process pool for longer than one request). This can lead, over time and given huge number of requests, to small memory leaks.

To avoid this, Apache comes with a configuration directive that helps in reduction of such problems. This directive is MaxRequestsPerChild, and it is set to 10,000 by default. You may also want to consider using MaxMemFree directive, which forces Apache's pool machinery to release free memory more agressively. Making changes to both of these directives may have performance implications, so test in your scenario before applying.

Language construct overview
==========

The template language of mod_spin is simple, it has a loop and a few forms of conditionals. You can see some examples below.

Loop:

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

Conditionals:

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

#if(${reference} == "literal text")
  ${reference} equals text
#else
  ${reference} doesn't equal text
```

```
#end

#if(${reference} == ${anotherreference})
  ${reference} equals ${anotherreference}
#else
  ${reference} doesn't equal ${anotherreference}
#end

#if($#{reference} == 3)
  the size of ${reference} equals 3
#else
  the size ${reference} doesn't equal 3
#end

#if($@{reference.column} % 2 == 0)
  current index of ${reference.column} is divisible by 2
#else
  current index of ${reference.column} is not divisible by 2
#end
```

Conditionals can also be reversed in meaning, by using #unless instead of #if. All valid #if constructs are possible with #unless as well. Here is an example:

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

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.

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.

References
==========

They come in three flavours:

```
${reference}  - the value (text, rows) of the reference (regular reference)
$#{reference} - the size of the reference (size reference)
$@{reference} - current index of the reference within the loop (index ref.)
```

```
The first form is straightforward as it simply takes the value of the
reference and it uses that. Used within text, only singles get substituted.
Used in a #for loop, singles get looped around once, rows get the number of
loops equivalent to the number of rows.

The second form takes the size of the reference, which for singles means the
length of the text and for rows the number of rows in each of the columns.
This form doesn't make sense in a #for loop and if placed there it will be
treated as a regular reference.

The third form takes the current index of the reference within a loop, if
applicable. Indexes start at 1. If the index wouldn't make sense (e.g. the
reference is a single, there is no loop etc.), it is treated as zero or as
NULL in conditionals and replaced with "0" if placed within text. This form
doesn't make sense in a #for loop and if placed there it will be treated as a
regular reference.

Text
====

Any text that isn't part of the #for loop or #if/#unless, will be literally
copied into the output. The space occupied by #for, #if and #unless will not
be space filled in the output, but removed as if it never existed.

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.

Loops
=====

These are quite simple:

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

You can only use regular references to loop around and other types of
references placed in #for will be treated as regular. For instance:

#for($#{reference})
  some text here
#end

is the same as:

#for(${reference})
  some text here
#end

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.

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

The only other command in mod_spin apart from #for loop is the conditional, as
shown above in the overview. Again, it looks like this, for the simplest of
expressions (i.e. a reference):

#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

Expressions valid in conditionals
=================================

An expression placed inside #if or #unless always starts with a reference and
placing anything else on the left is an error (or more explicitly, a parsing
error). Here are all the forms of expressions allowed in conditionals and when
they yield truth:

#if(${ref})  - ref exists and is not NULL
#if($#{ref}) - the size of ref is greater than zero
#if($@{ref}) - the current index of ref is greater than zero

#if(${ref}  =~ /regex/) - ref matches Perl compatible regular expression regex
#if($#{ref} =~ /regex/) - size of ref, as string, matches regex
#if($@{ref} =~ /regex/) - index of ref, as string, matches regex

#if(${ref}  == "str") - ref is the same as literal string str
#if($#{ref} == "str") - size of ref, as string, is the same as string str
#if($@{ref} == "str") - index of ref, as string, is the same as string str

#if(${ref}  == num) - ref, as number, equals num (integer >= 0)
#if($#{ref} == num) - size of ref is num
#if($@{ref} == num) - index of ref is num

#if(${ref1} == ${ref2})  - ref1 is equal to ref2, as strings
#if(${ref1} == $#{ref2}) - ref1 is equal to size of ref2, as strings
#if(${ref1} == $@{ref2}) - ref1 is equal to index of ref2, as strings

#if($#{ref1} == ${ref2})  - size ref1 is equal to ref2, as numbers
#if($#{ref1} == $#{ref2}) - size ref1 is equal to size of ref2
#if($#{ref1} == $@{ref2}) - size ref1 is equal to index of ref2

#if($@{ref1} == ${ref2})  - index of ref1 is equal to ref2, as numbers
#if($@{ref1} == $#{ref2}) - index of ref1 is equal to size of ref2
#if($@{ref1} == $@{ref2}) - index of ref1 is equal to index of ref2

#if(${ref}  % mod == num) - ref, as number, modulo mod (integer > 0) is num
#if($#{ref} % mod == num) - size of ref modulo mod is num
#if($@{ref} % mod == num) - index of ref modulo mod is num

Regular expression matches, integer comparisons, literal string, reference to
reference and modulo expression comparisons don't work for regular references
that are not singles and they will always yield false. No pointer comparisons
are ever done, so attemting to compare rows type data will always fail. Of
```

course, all this cannot be determined at parse time, but at runtime. As you
can see, in expressions with a reference on the right, the reference on the
left is the "master" and it determines the type of comparison done in the
expression. For regular references, this is a string comparison (i.e. text is
compared, not pointers), for sizes and indexes, it is a numerical comparison.

Numbers, except mod in the modulo expression, are all integers greater or
equal zero. Literal strings are double quoted (e.g. "a string"). To escape the
double quote itself, use "a string with a \" in it". Regular expressions are
specified withing slashes (e.g. /^begin.*$/). To escape the slash, use
a backslash before it: /^begin\/.*$/. Note that mod_spin isn't aware of any
character encodings and from its perspective bytes are characters. If you need
to make comparisons that take into account character encodings, you will have
to do that inside your applications (for now).

Indexes for singles and rows outside a relevant loop are assumed to be zero.

If reference on the left doesn't exist or is NULL, the whole expression will
always evaluate to false and the bit after #else (if any) will end up being
processed.

Regular references are converted to numbers using the atol() function, so
strings that don't start with numbers turn out as zero.

Loops, conditionals 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.

Similarly, other conditional forms behave as expected, because they always
have a reference on the left hand side. For instance:

#if($#{impossible.reference} == 3)
  this will never be visible
#else
  this will always be visible
#end

And here is 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.

References with explicit indexes
================================

It is possible to use references that have an explicit index within a #for
loop. It is done like this:

#for{${abc})
  Reference with implicit index: ${abc.xyz}
  Reference with explicit index: ${abc.xyz[3]})
#end

The second line within the loop above would always display the third row
(provided the data there is single) of the "xyz" column in "abc" rows.

Such references can be use anywhere where they can be successfully resolved.
For instance, this should also work (i.e. produce output):

Outside any loop: ${abc.xyz[3].x[2]}

Provided, of course, that abc is rows containing column xyz that has at least
three rows, containing column x, which has at least two rows and the second
element in that column is a single.

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

Service function is the main entry function into your application. It is
called in the handler phase of request processing and BEFORE template
processing, so it has the potential to change which template is going to be
processed (only when used as a handler) as well as to decline or do the
processing completely.

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.

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. Please note that when using mod_spin as a filter, the only valid return code is OK.

The meaning of return codes is as follows:

OK: Everything was OK, commit application/session store and 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.

HTTP_ERROR (e.g. HTTP_INTERNAL_SERVER_ERROR): Stop all processing and give control back to Apache request processing. Don't commit application/session store, since there was an error in processing.

ANYTHING ELSE: Commit application/session store and give control back to Apache request processing without processing the template.

Some examples of ANYTHING ELSE would be:

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.

Prepare function
================

There is an extra hook that is called before the request is handled, in the fixup phase of Apache request processing. By default, this function is called rxv_spin_prepare() and it takes one argument - the context. It is called if it exists in the shared library.

The main purpose of the hook is to allow application writers to insert their code before the actual request processing. This comes in handy, especially if you want some code executed for URIs that aren't handled by the mod_spin handler, but fall under the application umbrella. For instance, you can have authentication code in this function, thus using a mod_spin application to regulate access to all URIs of a configured application (see spin_app for an example).

This function is not called for sub-requests. It is also not called unless an application is configured for that particular request (i.e. per virtual host, directory, location etc.).

Please note that at the point of call of this function, request parameters have not been parsed yet by libapreq2. Meaning, although you do get the context, you are getting only some of the information that is normally available to the rxv_spin_service() function. This was designed on purpose, since some of the requests that pass through here may not be handled by mod_spin handler at all. In other words, we are not consuming the body of the requests here.

The function returns an integer, which can be OK, DECLINED, DONE or any other HTTP_code. It will be returned back to the caller (i.e. Apache hook machinery) directly from the fixup hook.

Init function
=============

This function, if exists, will be called when the application is dynamically linked into Apache. It is used for process specific initialisation. By default, this function is called rxv_spin_init(). It takes context as an argument, but it doesn't return any values.

The code of mod_spin will allow only a single thread to execute this function at a time. However, if the function is such that it should be called only once per process, the function itself will have to make sure code isn't executed more than once.

You can attach pool cleanup function(s) to process specific pool if there is a need to clean up after the init function. This should be done in the init function itself.

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

All loading of shared libraries is done using apr_dso_load() call from APR, so mod_spin makes sure that each process keeps cache of loaded DSOs and that it opens a particular library only once. This is done to avoid registration of a pool cleanup for each call to apr_dso_load(), which would quickly grow process pool. Once the new applications are deployed, in order to reliably reload them, the main Apache process has to be gracefully restarted, so that all child processes die and new ones replace them. This will ensure new applications are loaded across the board.

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.

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.

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. It is worth an effort to reduce what's dynamic to a minimum.

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

mod_spin relies on mod_unique_id to provide a unique session identifier by producing an MD5 hash of it. It then produces an HMAC MD5 of the hash, using the crypto salt (key). Both of these (unique id hash and the HMAC) 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.

You must define SpinCookie configuration parameter, or the sessions won't be supported for that application at all.

Each session will have corresponding record in the table specified by SpinStoreTable configuration directive, of the SQL database you point to using SpinStore configuration directive. The application will have a record identified with "__application" in the same table as well. 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 session data). If you configure file based store backend, application will store its data in the "__application" file located in the directory specified in SpinStore directive. Each session will have its own data stored in the same directory, in the file named after the session ID. Format of these files and records in SQL database is XML.

The maintenance of stale sessions is easy. Simply define a cron job that goes around and deletes from the table whatever is older than you like. SpinTimeout configuation directive (if defined above zero) won't actually delete any records from the table - the parameter is used to determine when the data of the session is too old and should therefore be ignored. You need to have an outside job for cleaning records that are very old. Ditto for file based backend store, except that session files (instead of SQL records) that haven't been accessed for a certain amount of time should be removed.

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 SpinStore 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 SpinStore.

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 s (p*)>
  <!ELEMENT p (#PCDATA)>
  <!ATTLIST p n CDATA #REQUIRED>
]>
<s>
```

```
  <p n="spinparameter1">The value associated with spinparameter1</p>
  <p n="spinparameter2">The value associated with spinparameter2</p>
</s>
```

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 <p/> tags are placed into the application's store. On each new request and if the last parsing was more than ten seconds ago, the configuration file is checked for modification. If the configuration file is newer, it is parsed again and the keys and values are placed into the application store. New values will overwrite existing values associated with same keys, but other key/value pairs in the application store will not be changed.

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 connection string, which is specified when the database is opened. mod_spin creates a hash of all those identifiers and stores connection pools, 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 an existing connection from the pool. This can dramatically improve performance of applications that frequently use (database) connections.

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

With every process having its own private connections to the back-end server, the total number of connections can be rather big. 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 may be 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.

SpinConnPool configuration directive enables system administrators to control pooling of connections. This can be useful if you find that a particular application is causing a large number of connections to be kept open.

SpinConnCount configuration directive enables system administrators to control the number of connections in the pool. By default, up to 5 connections will be kept in the per-process connection pool, for each connection identifier.

You can register any type of connection with the connection pool. 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 come to mind first) that
solved all of these problems (in a slightly different manner) 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.

Thread safety
=============

Some Apache MPMs (Multi-Processing Modules), e.g. worker, spin off multiple
threads of execution, under which mod_spin should be OK. However, if you spin
off your own threads and want to use mod_spin structures across threads, you
MUST SYNCHRONISE access, or you will experience problems.

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
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 2.2 or better: http://httpd.apache.org/.

2. Apache Portable Runtime and Apache Portable Runtime Utilities 1.2.8
   or better. Version 1.3.x is highly recommended, as it contains code for
   correct handling of database column names. The home page of APR project is
   here: http://apr.apache.org/

   IMPORTANT NOTE: You should always use the version of APR/APU that you used
   to link Apache with. 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 2.2 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 is known to work with libapreq2-2.07 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 2.2, 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 2.5.33 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.

7. The build and install are tested with GNU tools on Linux (Fedora). Other
   tools may work, but you will have to find out for yourself.

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

There are three options to install mod_spin into Apache:

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

IMPORTANT: You have to have Apache 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-1-config, apu-1-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, 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 something like:

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.
For instance, this library installs its header files in /usr/include/libxml2
on Fedora. 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 (or /usr/local/lib64 on some 64-bit systems). If you want
rxv_spin library in the system location (i.e. /usr/lib or /usr/lib64), 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 or /usr/lib64. When building
from source, mod_spin will link against a specific library, possibly located
in /usr/local/lib or /usr/local/lib64, 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
==========

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
==========

Session identifiers are taken from mod_unique_id and hashed using MD5. An HMAC
MD5 of the hash is appended to the hash. This then creates a method of
protection against session spoofing (not unbreakable, but enough to prevent

```
massive amount of sessions from being introduced). 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 an RPM based distro to do this (the packages are normally
built for Fedora on x86). If you are, then it's really simple:

rpm -Uvh mod_spin-X.Y.Z-R.ARCH.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 (or deploy your own
files via RPM of you application) 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:

rpm -Uvh mod_spin-devel-X.Y.Z-R.ARCH.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-flex-reentrant=/path/to/flex/installation/directory]
make
make install

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

IMPORTANT: THIS OPTION HAS NOT BEEN TESTED IN A WHILE AND IT MAY NOT WORK AT
ALL. IT SURE ISN'T RECOMMENDED.

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-flex-reentrant=/path/to/flex/installation/directory]
make
make install

From the top directory of Apache source do:

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

./buildconf

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

make distclean

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

```
Configure Apache, 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 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 source for static linking, you should
run:

./buildconf

from the top directory of the Apache 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 2.5.33 and above), but unless you've changed the scanner rules,
rebuilding the RPMs shouldn't fail (i.e. generated C files are shipped with
the distribution). If scanner rules were changed, however, you will need
reentrant flex. Replacing flex with a reentrant package may have system wide
implications, especially if you compile software that produces scanners.
Therefore, mod_spin was designed to use flex-reentrant package, which can
coexist peacefully alongside flex on your RPM based system.

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 build 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 or rebuild the binary RPM and install it 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!

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

If you are running Security Enhanced Linux (for instance, Fedora), you may need to do a bit of extra configuration for mod_spin to work correctly (depending on the application/session store type you choose). For instance, with SQLite2/3, if you place application/session database in /var/tmp/myapp.db file, you'll have to make sure that this file belongs to the correct security context, which is normally root:object_r:httpd_cache_t. To do that, you'd run:

```
chcon -u root -r object_r -t httpd_cache_t /var/tmp/myapp.db
```

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.db file, although file permissions allow it, file access would be denied by policy and the client would get an internal server error response.

To make labeling of files correct, use semanage command, to set you local file contexts to the correct value. For instance:

```
semanage fcontext -a -t http_cache_t '/var/tmp/modspin(/.*)?'
```

Obviously, if you're running other databases where connections are made via a TCP or Unix domain socket, you will have to allow Apache to connect to those sockets. On Fedora, this is done using SELinux booleans. You can obtain the list of those by running sestatus -b. You can change them by running setsebool. Check relevant manual pages for all details.

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
#    AddOutputFilter SPIN-TEMPLATE .sm
    AddType text/html .sm
    SpinApplication /usr/local/libexec/spinapps/spinapp.so
    SpinAppInit rxv_spin_init                          # Default
    SpinAppPrepare rxv_spin_prepare                    # Default
    SpinAppService rxv_spin_service                    # Default
    SpinAppConfig /usr/local/etc/spinapps/spinapp.xml
    SpinStore file:/var/tmp/mod_spin
#    SpinStore sqlite3:/var/tmp/mod_spin/store.db
#    SpinStore mysql:dbname=spintest,flags=CLIENT_FOUND_ROWS
#    SpinStore pgsql:dbname=spintest
    SpinStoreTable spinstore                           # Default
    SpinCookie SpinSession
    SpinCookiePath /                                   # Default
    SpinCookieDomain .example.com
    SpinTimeout 0                                      # Default
    SpinCache on                                       # Default
    SpinConnPool on                                    # Default
    SpinConnCount 5                                    # Default
    SpinSalt                                           # No default
</IfModule>
```

SpinApplication, SpinAppInit, SpinAppPrepare, SpinAppService, SpinAppConfig, SpinStore, SpinStoreTable, SpinCookie, SpinCookiePath, SpinCookieDomain, SpinTimeout, SpineCache, SpinConnPool and SpinSalt can be placed in the main server section, virtual host section, directory or location section. The nearest available will be used. SpinConnCount 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.

SpinAppInit is the name of the application init function in the shared library (rxv_spin_init() by default). This function, if exists, will be called when the library is dynamically linked into Apache.

SpinAppPrepare is the name of the application prepare function (called in the fixups phase of the request processing) in the shared library (rxv_spin_prepare() by default). Function named in SpinAppPrepare will be called after the library is dynamically linked by mod_spin.

SpinAppService is the name of the application service function (called from the handler) in the shared library (rxv_spin_service() by default). Function named in SpinAppService will be called after the library is dynamically linked by mod_spin.

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 s (p*)>
  <!ELEMENT p (#PCDATA)>
  <!ATTLIST p n CDATA #REQUIRED>
]>
<s>
  <p n="spinparameter1">The value associated with spinparameter1</p>
  <p n="spinparameter2">The value associated with spinparameter2</p>
</s>
```

The DTD of the document is local in the above example, but you don't need to specify it in your file. You can even omit the XML declaration.

SpinStore is a location (in APR Util DBD speak) of your session and appliction

store. The store is actually a table in an SQL database (e.g. PostgreSQL, SQLite3 etc.). This parameter obeys SpinConnPool parameter. With version 1.3.x of APR Util DBD, supported database prefixes are pgsql, mysql, sqlite2 (this won't work, as this driver doesn't support prepared statements), sqlite3 and oracle. This obviously depends on how APR Utils was compiled and linked and what databases really exist in your environment. If you're using MySQL, you have to specify flags=CLIENT_FOUND_ROWS in your connect string, or the store isn't going to work properly.

You can also specify a special driver, file:, followed by the directory where the store will be located. In this case, application and session data is kept in XML files. This store backend features best performance. See the above example snippet from the httpd.conf file for more details.

SpinStoreTable is the name of the SQL table used for the store. This parameter has no effect when file based store is in use.

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. Session cookie handling is entirely done in mod_spin. You need to define SpinSalt for sessions to work.

SpinCookiePath is the path of the cookie used for session management. It is set to "/" by default.

SpinCookieDomain is the domain of the cookie used for session management. It is not set by default.

SpinTimeout is session timeout in seconds. After this time, the session data from session store will be ignored (i.e. not retrieved) and overwritten by new data if the same session ID is used. By default, this value is set to zero, meaning no timeout. Specifying anything but an integer greater or equal zero will set the value to zero.

An external program, possibly started from cron, can reap session data that timed out, whether in a database or file based store.

SpinCache enables template caching and it is turned on by default. Be careful with this option - large number of cached templates and/or big templates can quickly cause Apache to consume large amounts of memory and create a DOS. NEVER use big templates with the handler when the caching is turned on. Either use the filter, to which this option does not apply or disable caching for the big templates. Any template that has not been used for more than 10 seconds will be removed from cache.

SpinConnPool enables pooling of connections and it is turned on by default.

SpinConnCount specifies the number of connections to keep in the connection pool for every connection identifier, per process. Specifying anything but an integer greater than zero will set the value to the default, which is 5. Note that any connection pool that has not been used for more than a minute will be removed.

SpinSalt is the cryptographic salt (key) used to produce HMACs for 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 spoof 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, you can set SpinSalt parameter more than once and that is not an error. This functionality exists to enable security savvy administrators to rotate cryptographic salt, in order to make session spoofing even more difficult.

```
An external program, possibly started from cron, can manipulate SpinSalt
parameters in the configuration file (or, file included from 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 1.2.0 released 2009-12-29

** stable

* version 1.1.9 released 2009-02-18

Here we introduce caching of templates again. It should give decent
performance improvements. Note that caching applies only to templates served
by the handler and not the filter. Read the INSTALL file to find out which
templates are suitable for caching.

** move xml2config variable where it belongs
** fix data creation from brigades
** add rxv_spin_array() function
** make SpinSalt non-global
** add SPINLIBS to librxv_spin.la
** introduce template caching: SpinCache directive
** let parser do the parsing
** tighten up scanner rules to match more text
** save some memory by redoing AST structures

* version 1.1.8 released 2008-08-15

Please note that this version marks major changes in the data API. Data is now
an opaque type and all access to it is through functions. Many functions
changed their names, some were deleted, some added etc. Check the
documentation for all the details, as number of changes is too big to mention
here.

** add path and domain cookie options
** add rxv_spin_ses_kill() for destroying sessions
** use proper row numbers for apr_dbd_get_row()
** use apr_version.h and apu_version.h
** shorten XML tags to <s/> and <p/> and attribute n
** make data types opaque and adjust API (BIG: check docs!)
** introduce ten second delay for configuration parsing
** use APU resource lists to manage connection pools
** introduce reference with fixed indexes

* version 1.1.7 released 2007-11-20

** use shared buckets to reduce copying
** improve scanning speed by matching multiple lines of text

* version 1.1.6 released 2007-08-23

** relicense under LGPL 2.1
** add rxv_spin_meta_brigade()
** switch to xmlreader/xmlwriter for XML processing
** add filter functionality
```

```
** use mmap()-ed files
** kill SpinSendfile and SpinCacheAll configuration directives
** kill SpinCache and SpinCacheCount configuration directives
** template processing with finite memory even for big files
** rework scanner to avoid rescanning tokens
** rework scanner to scan bucket brigades
** avoid namespace pollution in scanner/parser
** implement one minute connection timeout

* version 1.1.5 released 2007-08-14

** application/session data saved in XML
** file support for application/session store
** make sure we clean up in child in case of fork
** split up -libs RPM subpackage
** add pkgconfig support
** fix docs URLs in RPM spec file
** detect Doxygen tag files better
** use mtime instead of atime for session expiry
** extend configuration script with more build time variables
** move more M4 macros to private file

* version 1.1.4 released 2007-05-31

** improvements to RPM spec file

* version 1.1.3 released 2007-02-15

** fix non-portable shell construct in mod_spin.m4
** introduce SpinAppInit configuration directive
** use process pool under a thread lock
** only use APR thread calls if threads are compiled in
** add rxv_spin_ctx_tpool() function
** add MySQL store creation script
** implement RFC 2104 HMAC MD5 for session cookies
** add rxv_spin_hash() and rxv_spin_hmac() functions

* version 1.1.2 released 2006-10-11

** documentation fixes
** introduce SELECT pool to reduce memory pressure
** update RPM dependencies

* version 1.1.1 released 2006-06-27

Please note that this development version features MAJOR CHANGES TO THE API
AND APPLICATION/SESSION STORE implementation. There is almost no chance that
your existing mod_spin application will work with this one unchanged. Check
the API documentation for all details. Some examples can be seen in spin_app,
spin_feedback and spin_auth applications.

** improve RPM spec file
** ship test template and database creation script in RPM
** set SHOW_DIRECTORIES to NO in doxygen.conf
** improve APR/APU config script detection
** add rxv_spin_dso_load() function
** add rxv_spin_db_data() function
** remove rxv_spin_db_exec() function
** add rxv_spin_db_select()/rxv_spin_db_query() functions
** add rxv_spin_db_pselect()/rxv_spin_db_pquery() functions
** switch to apr_dbd for database access
** remove rxv_spin_db_info() and rxv_spin_db_reset()
** remove legacy macros
** remove rxv_spin_db_result_t
** remove SpinWorkspace configuration directive
** add SpinStore and SpinStoreTable configuration directives
** switch to opaque pointers
```

```
** drop util.c and import functions into mod_spin.c
** replace rxv_spin_context_t with rxv_spin_ctx_t
** make rxv_spin_conn_t rxv_spin_db_t
** completely rework the database API
** rename cpool.c to conn.c
** don't explicity check libraries APR/APU drags in
** use enum instead of #define for various constants
** use SQL instead of SDBM for application/session store backend
** provide create-store.sql for store creation
** segfault on out of memory conditions
** remove rxv_spin_db_escape() function
** implement connection pool LRU
** implement template cache LRU
** rename SpinTemplateCache configuration directive to SpinCache
** introduce SpinCacheCount and SpinConnCount configuration directives

* version 1.1.0 released 2005-11-22

This is a new, development branch of mod_spin and is therefore experimental
code. Code _may_ change significantly as we go toward 1.2.0 release, so don't
count on things being there in the next development release.

** new development branch
** use multi-env libapreq2 (2.06 and above)
** replace req and jar with apreq handle in the context
** introduce rxv_spin_prepare() callback
** rename SpinAppEntry configuration parameter to SpinAppService
** add SpinAppPrepare configuration parameter
** do away with pre-connection hook
** rename spin directory to src
** don't fail install if mod_spin.m4 cannot be installed
** use flex-reentrant if available
** check for ap_regex headers
** use --include instead of --cflags for newer mysql_config
** use Bitstream fonts in doxygen.css
** make GCC 4.0 complain less
** fix incorrect AC_PATH_PROG calls
** introduce size and index references
** introduce regex, literal, numeric and modulo conditionals
** remove SpinClearCount configuration directive
** add SpinTemplateCache configuration directive
** link apr, apr-util, apreq2 and rxv_spin libs explicitly
** add SpinConnPool configuration directive
** add RXV_SPIN_CONFIG_NICE for making config.nice
** check for inode change on the template
** conditional use of flex and bison

* 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 LESSER GENERAL PUBLIC LICENSE
       Version 2.1, February 1999

 Copyright (C) 1991, 1999 Free Software Foundation, Inc.
 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

[This is the first released version of the Lesser GPL.  It also counts
 as the successor of the GNU Library Public License, version 2, hence
```

```
 the version number 2.1.]

    Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
Licenses are intended to guarantee your freedom to share and change
free software--to make sure the software is free for all its users.

  This license, the Lesser General Public License, applies to some
specially designated software packages--typically libraries--of the
Free Software Foundation and other authors who decide to use it.  You
can use it too, but we suggest you first think carefully about whether
this license or the ordinary General Public License is the better
strategy to use in any particular case, based on the explanations below.

  When we speak of free software, we are referring to freedom of use,
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 and use pieces of
it in new free programs; and that you are informed that you can do
these things.

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

  For example, if you distribute copies of the library, whether gratis
or for a fee, you must give the recipients all the rights that we gave
you.  You must make sure that they, too, receive or can get the source
code.  If you link other code with the library, you must provide
complete object files to the recipients, so that they can relink them
with the library after making changes to the library and recompiling
it.  And you must show them these terms so they know their rights.

  We protect your rights with a two-step method: (1) we copyright the
library, and (2) we offer you this license, which gives you legal
permission to copy, distribute and/or modify the library.

  To protect each distributor, we want to make it very clear that
there is no warranty for the free library.  Also, if the library is
modified by someone else and passed on, the recipients should know
that what they have is not the original version, so that the original
author's reputation will not be affected by problems that might be
introduced by others.

  Finally, software patents pose a constant threat to the existence of
any free program.  We wish to make sure that a company cannot
effectively restrict the users of a free program by obtaining a
restrictive license from a patent holder.  Therefore, we insist that
any patent license obtained for a version of the library must be
consistent with the full freedom of use specified in this license.

  Most GNU software, including some libraries, is covered by the
ordinary GNU General Public License.  This license, the GNU Lesser
General Public License, applies to certain designated libraries, and
is quite different from the ordinary General Public License.  We use
this license for certain libraries in order to permit linking those
libraries into non-free programs.

  When a program is linked with a library, whether statically or using
a shared library, the combination of the two is legally speaking a
combined work, a derivative of the original library.  The ordinary
General Public License therefore permits such linking only if the
entire combination fits its criteria of freedom.  The Lesser General
```

Public License permits more lax criteria for linking other code with
the library.

We call this license the "Lesser" General Public License because it
does Less to protect the user's freedom than the ordinary General
Public License. It also provides other free software developers Less
of an advantage over competing non-free programs. These disadvantages
are the reason we use the ordinary General Public License for many
libraries. However, the Lesser license provides advantages in certain
special circumstances.

For example, on rare occasions, there may be a special need to
encourage the widest possible use of a certain library, so that it becomes
a de-facto standard. To achieve this, non-free programs must be
allowed to use the library. A more frequent case is that a free
library does the same job as widely used non-free libraries. In this
case, there is little to gain by limiting the free library to free
software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free
programs enables a greater number of people to use a large body of
free software. For example, permission to use the GNU C Library in
non-free programs enables many more people to use the whole GNU
operating system, as well as its variant, the GNU/Linux operating
system.

Although the Lesser General Public License is Less protective of the
users' freedom, it does ensure that the user of a program that is
linked with the Library has the freedom and the wherewithal to run
that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and
modification follow. Pay close attention to the difference between a
"work based on the library" and a "work that uses the library". The
former contains code derived from the library, whereas the latter must
be combined with the library in order to run.

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

0. This License Agreement applies to any software library or other
program which contains a notice placed by the copyright holder or
other authorized party saying it may be distributed under the terms of
this Lesser General Public License (also called "this License").
Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data
prepared so as to be conveniently linked with application programs
(which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work
which has been distributed under these terms. A "work based on the
Library" means either the Library or any derivative work under
copyright law: that is to say, a work containing the Library or a
portion of it, either verbatim or with modifications and/or translated
straightforwardly into another language. (Hereinafter, translation is
included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for
making modifications to it. For a library, 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 library.

Activities other than copying, distribution and modification are not
covered by this License; they are outside its scope. The act of
running a program using the Library is not restricted, and output from

```
such a program is covered only if its contents constitute a work based
on the Library (independent of the use of the Library in a tool for
writing it).  Whether that is true depends on what the Library does
and what the program that uses the Library does.

  1. You may copy and distribute verbatim copies of the Library's
complete 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 distribute a copy of this License along with the
Library.

  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 Library or any portion
of it, thus forming a work based on the Library, 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) The modified work must itself be a software library.

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

    c) You must cause the whole of the work to be licensed at no
    charge to all third parties under the terms of this License.

    d) If a facility in the modified Library refers to a function or a
    table of data to be supplied by an application program that uses
    the facility, other than as an argument passed when the facility
    is invoked, then you must make a good faith effort to ensure that,
    in the event an application does not supply such function or
    table, the facility still operates, and performs whatever part of
    its purpose remains meaningful.

    (For example, a function in a library to compute square roots has
    a purpose that is entirely well-defined independent of the
    application.  Therefore, Subsection 2d requires that any
    application-supplied function or table used by this function must
    be optional: if the application does not supply it, the square
    root function must still compute square roots.)

These requirements apply to the modified work as a whole.  If
identifiable sections of that work are not derived from the Library,
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 Library, 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 Library.

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

  3. You may opt to apply the terms of the ordinary GNU General Public
```

License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you 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.

If distribution of 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 satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer’s own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the

Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be 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.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact

13. The Free Software Foundation may publish revised and/or new versions of the Lesser 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 Library 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 Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, 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

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "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 LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. 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 LIBRARY 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 LIBRARY (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 LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), 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 Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. 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 library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

```
    This library 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
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA

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

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

  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice

That's all there is to it!
```

2 Module Index

2.1 Modules

Here is a list of all modules:

3 File Index

3.1 File List

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

4 Module Documentation

4.1 Data functions

Defines

- #define rxv_spin_multi(d) (!rxv_spin_string(d) && rxv_spin_size(d)>0)
- #define rxv_spin_single(d) (!rxv_spin_multi(d))
- #define rxv_spin_strim2(s) rxv_spin_strim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_-RIGHT))
- #define rxv_spin_striml(s) rxv_spin_strim((s),(RXV_SPIN_TRIM_LEFT))
- #define rxv_spin_strimr(s) rxv_spin_strim((s),(RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_lower(s)
- #define rxv_spin_upper(s)
- #define rxv_spin_trim(s, w)
- #define rxv_spin_trim2(s) rxv_spin_trim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_-RIGHT))
- #define rxv_spin_triml(s) rxv_spin_trim((s),(RXV_SPIN_TRIM_LEFT))
- #define rxv_spin_trimr(s) rxv_spin_trim((s),(RXV_SPIN_TRIM_RIGHT))

Typedefs

- typedef struct rxv_spin_data rxv_spin_data_t
- typedef struct rxv_spin_curs rxv_spin_curs_t

Enumerations

- enum rxv_spin_trim_e { RXV_SPIN_TRIM_LEFT = 1, RXV_SPIN_TRIM_RIGHT }

Functions

- rxv_spin_data_t * rxv_spin_datum (apr_pool_t *pool, const char *str, rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_mdatum (apr_pool_t *pool, const char *str, apr_size_t size, rxv_spin_-data_t *data)
- char * rxv_spin_string (rxv_spin_data_t *single)
- apr_hash_t * rxv_spin_guts (rxv_spin_data_t *rows)
- apr_size_t rxv_spin_size (rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_column (apr_pool_t *pool, const char *name, rxv_spin_data_t *data,...)
- rxv_spin_data_t * rxv_spin_parse (apr_pool_t *pool, const char *name, char *str, const char *sep, rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_array (apr_pool_t *pool, const char *name, apr_array_header_t *arr, rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_brigade (apr_pool_t *pool, const char *name, apr_bucket_brigade *bb, rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_null (apr_pool_t *pool, const char *name, apr_size_t size, rxv_spin_-data_t *data)
- rxv_spin_data_t * rxv_spin_rows (apr_pool_t *pool, rxv_spin_data_t *data,...)
- apr_size_t rxv_spin_first (apr_pool_t *pool, rxv_spin_data_t *rows,...)
- apr_size_t rxv_spin_next (rxv_spin_curs_t *curs)
- rxv_spin_data_t * rxv_spin_this (rxv_spin_curs_t *curs)

- rxv_spin_data_t * rxv_spin_entry (rxv_spin_data_t *rows, const char *name, apr_size_t index)
- rxv_spin_data_t * rxv_spin_resize (rxv_spin_data_t *data, apr_size_t size)
- rxv_spin_data_t * rxv_spin_copy (apr_pool_t *pool, rxv_spin_data_t *src, rxv_spin_data_t *data)
- char * rxv_spin_slower (const char *str)
- char * rxv_spin_supper (const char *str)
- char * rxv_spin_strim (const char *str, rxv_spin_trim_e what)

4.1.1 Detailed Description

Data functions (mod_spin API)

4.1.2 Define Documentation

4.1.2.1 #define rxv_spin_multi(d) (!rxv_spin_string(d) && rxv_spin_size(d)>0)

check if data is of rows type

4.1.2.2 #define rxv_spin_single(d) (!rxv_spin_multi(d))

check if data is of single type

4.1.2.3 #define rxv_spin_strim2(s) rxv_spin_strim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_-TRIM_RIGHT))

trim string on both sides

4.1.2.4 #define rxv_spin_striml(s) rxv_spin_strim((s),(RXV_SPIN_TRIM_LEFT))

trim string on the left side

4.1.2.5 #define rxv_spin_strimr(s) rxv_spin_strim((s),(RXV_SPIN_TRIM_RIGHT))

trim string on the right side

4.1.2.6 #define rxv_spin_lower(s)

Value:

```
((s)?(rxv_spin_single(s)                                                \
              ?rxv_spin_datum(NULL,rxv_spin_slower(rxv_spin_string(s)),(s)) \
              :NULL)                                                    \
             :NULL)
```

convert single to lowercase

4.1.2.7 #define rxv_spin_upper(s)

Value:

```
((s)?(rxv_spin_single(s)                                                \
              ?rxv_spin_datum(NULL,rxv_spin_supper(rxv_spin_string(s)),(s)) \
              :NULL)                                                    \
            :NULL)
```

convert single to uppercase

4.1.2.8 #define rxv_spin_trim(s, w)

Value:

```
((s)?(rxv_spin_single(s)                                                  \
              ?rxv_spin_datum(NULL,rxv_spin_strim(rxv_spin_string(s),(w)),(s)) \
              :NULL)                                                      \
            :NULL)
```

trim single

4.1.2.9 #define rxv_spin_trim2(s) rxv_spin_trim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_-TRIM_RIGHT))

trim single on both sides

4.1.2.10 #define rxv_spin_triml(s) rxv_spin_trim((s),(RXV_SPIN_TRIM_LEFT))

trim single on the left side

4.1.2.11 #define rxv_spin_trimr(s) rxv_spin_trim((s),(RXV_SPIN_TRIM_RIGHT))

trim single on the right side

4.1.3 Typedef Documentation

4.1.3.1 typedef struct rxv_spin_data rxv_spin_data_t

data type

4.1.3.2 typedef struct rxv_spin_curs rxv_spin_curs_t

cursor type

4.1.4 Enumeration Type Documentation

4.1.4.1 enum rxv_spin_trim_e

Enumerator:

RXV_SPIN_TRIM_LEFT trim whitespace on the left

RXV_SPIN_TRIM_RIGHT trim whitespace on the right

4.1.5 Function Documentation

4.1.5.1 rxv_spin_data_t* rxv_spin_datum (apr_pool_t * *pool*, const char * *str*, rxv_spin_data_t * *data*)

Create single data from from a nul terminated string. String is not copied.

Parameters:

pool Pool for allocation of data structure

str String to create data from

data Pointer to existing data or NULL to allocate anew

Returns:

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

Example:

```
rxv_spin_datum(pool,"some string",NULL);
```

Remarks:

Pool can be NULL is data is not NULL.

4.1.5.2 rxv_spin_data_t* rxv_spin_mdatum (apr_pool_t * *pool*, const char * *str*, apr_size_t *size*, rxv_spin_data_t * *data*)

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.

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

data Pointer to existing data or NULL to allocate anew

Returns:

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

Example:

```
rxv_spin_mdatum(pool,"1234567890",10,NULL);
```

Remarks:

Pool can be NULL is data is not NULL.

4.1.5.3 char* rxv_spin_string (rxv_spin_data_t * *single*)

Get a nul terminated string from single data. String is not copied.

Parameters:

single Single data

Returns:

nul terminated string, NULL on error

Example:

```
rxv_spin_string(data);
```

4.1.5.4 apr_hash_t* rxv_spin_guts (rxv_spin_data_t * *rows*)

Get a hash table from rows data. Hash table is not copied.

Parameters:

rows Rows data

Returns:

pointer to hash table NULL on error

Example:

```
rxv_spin_guts(data);
```

Remarks:

Values in the hash table are apr_array_header_t pointers. The arrays contain rxv_spin_data_t (meaning: array->elts is rxv_spin_data_t pointer), so it is possible to use this mechanism to iterate through individual colums as well. If you do modify data this way, make sure all arrays contained in the hash have the same size, or you may get segfaults when you try to access the data. Keep in mind that although you can find out the size of the rxv_spin_data_t this way, it is still an opaque type that should be accessed through mod_spin API.

4.1.5.5 apr_size_t rxv_spin_size (rxv_spin_data_t * *data*)

Get the size of data. This does not include terminating nul byte for single. Number of rows is returned for rows.

Parameters:

data Data

Returns:

size of single or rows or 0 on error

Example:

```
rxv_spin_size(data);
```

4.1.5.6 rxv_spin_data_t* rxv_spin_column (apr_pool_t * *pool*, const char * *name*, rxv_spin_data_t * *data*, ...)

Create a column of data from other data. The data returned is of type rows. Data is not copied.

Parameters:

pool Pool for allocation of data structure

name Name of the column

data Pointer to existing data or NULL to allocate anew

... List of rxv_spin_data_t pointers, ending with NULL

Returns:

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

Example:

```
rxv_spin_column(pool,"column1",NULL,
                rxv_spin_rows(pool,NULL,x,y,z,NULL),
                rxv_spin_rows(pool,NULL,u,v,w,NULL),
                rxv_spin_single(pool,"some important data",NULL),
                NULL);
```

4.1.5.7 rxv_spin_data_t* rxv_spin_parse (apr_pool_t * *pool*, const char * *name*, char * *str*, const char * *sep*, rxv_spin_data_t * *data*)

Create a column of data by parsing a string. The data returned is of type rows. String to parse is NOT copied into pool storage before parsing, therefore, if you want to parse constant strings, you MUST copy them. 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

name Name of the column

str String to parse

sep Separators to use when parsing

data Pointer to existing data or NULL to allocate anew

Returns:

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

Example:

```
rxv_spin_parse(pool,"column1",
               apr_pstrdup(pool,"first str,second str/third str"),
               ",/",NULL);
```

4.1.5.8 rxv_spin_data_t* rxv_spin_array (apr_pool_t * *pool*, const char * *name*, apr_array_header_t * *arr*, rxv_spin_data_t * *data*)

Create a column of data from an array of nul terminated strings. The data returned is of type rows. Strings are not copied.

Parameters:

pool Pool for allocation of data structure

name Name of the column

arr Array of pointers to nul terminated strings

data Pointer to existing data or NULL to allocate anew

Returns:

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

Example:

```
rxv_spin_array(pool,"column1",arr,NULL);
```

4.1.5.9 rxv_spin_data_t* rxv_spin_brigade (apr_pool_t * *pool*, const char * *name*, apr_bucket_brigade * *bb*, rxv_spin_data_t * *data*)

Create a column of data by reading a bucket brigade. The data returned is of type rows. All read data is copied into the storage allocated from the pool.

Parameters:

pool Pool for allocation of data structure

name Name of the column

bb Bucket brigade to read

data Pointer to existing data or NULL to allocate anew

Returns:

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

Example:

```
rxv_spin_brigade(pool,"column1",bb,NULL);
```

4.1.5.10 rxv_spin_data_t* rxv_spin_null (apr_pool_t * *pool*, const char * *name*, apr_size_t *size*, rxv_spin_data_t * *data*)

Create an empty column of data.

Parameters:

pool Pool for allocation of data structures

name Name of the column

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

data Pointer to existing data or NULL to allocate anew

Returns:

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

Example:

```
rxv_spin_null(pool,"column1",rxv_spin_rws_len(r),NULL);
```

Remarks:

This function is useful for creating boilerplate columns.

4.1.5.11 rxv_spin_data_t* rxv_spin_rows (apr_pool_t * *pool*, rxv_spin_data_t * *data*, ...)

Merge rows of data. If rows contain different array lengths, the biggest size will be used. Data is not copied.

Parameters:

pool Pool for allocation of data structure

data Pointer to existing data or NULL to allocate anew

... List of rxv_spin_data_t pointers, ending with NULL

Returns:

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

Example:

```
rxv_spin_rows(pool,NULL,def,ghi,xyz,NULL);
```

Remarks:

Rows that appear later in the argument list override columns of the same name from earlier arguments. If columns need to have their size adjusted, empty elements will be pushed to the end of the arrays that columns point to. Note that give that data isn't copied, this resizing affects original columns too.

4.1.5.12 apr_size_t rxv_spin_first (apr_pool_t * *pool*, rxv_spin_data_t * *rows*, ...)

Get first row of data.

Parameters:

pool Pool for allocation of data structure

rows Data to get the first row from

... List of column names (constant pointer to nul terminated string) and cursors (pointer to pointer of rxv_spin_curs_t), ending with NULL

Returns:

1 if successful, or 0 if no rows available

Example:

```
rxv_spin_curs_t *c1=NULL,*c2=NULL;

rxv_spin_first(pool,rows,"column1",&c1,"column2",&c2,NULL);
```

4.1.5.13 apr_size_t rxv_spin_next (rxv_spin_curs_t * *curs*)

Get next row of data.

Parameters:

curs Pointer to the cursor

Returns:

current row number, counted from 1, or 0 if no more available

Example:

```
for(r=rxv_spin_first(p,rws,"c1",&c1,NULL);r;r=rxv_spin_next(c1)){
  ...
}
```

Remarks:

Note that it is sufficient to call rxv_spin_next() just once on any of the cursors that have been initialised through rxv_spin_first() to get the next row on all of them.

4.1.5.14 rxv_spin_data_t* rxv_spin_this (rxv_spin_curs_t * *curs*)

Get data from the cursor.

Parameters:

curs Pointer to to the cursor

Returns:

rxv_spin_data_t pointer, NULL on error

Example:

```
data=rxv_spin_this(curs);
```

4.1.5.15 rxv_spin_data_t* rxv_spin_entry (rxv_spin_data_t * *rows*, const char * *name*, apr_size_t *index*)

Get an entry from rows of data.

Parameters:

rows Rows data

name Name of the column

index Row number, starting at 1

Returns:

rxv_spin_data_t pointer, NULL on error

Example:

```
data=rxv_spin_entry(rows,"thiscolumn",3);
```

4.1.5.16 rxv_spin_data_t* rxv_spin_resize (rxv_spin_data_t * *data*, apr_size_t *size*)

Resize rows.

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(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.

4.1.5.17 rxv_spin_data_t* rxv_spin_copy (apr_pool_t * *pool*, rxv_spin_data_t * *src*, 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

src Data to copy, any type

data Pointer to existing data or NULL to allocate anew

Returns:

copy of the data, NULL on error

Example:

```
rxv_spin_copy(pool,data,NULL);
```

Remarks:

If you're using this function on rows, 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.

4.1.5.18 char* rxv_spin_slower (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_slower(single);
```

4.1.5.19 char* rxv_spin_supper (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_supper(single);
```

4.1.5.20 char* rxv_spin_strim (const char * *str*, rxv_spin_trim_e *what*)

Trim whitespace from a string. No copy is made. You MUST copy the string 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_strim(apr_strdup(pool,"  string with whitespace   ",
               RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_RIGHT);
```

4.2 Context functions

Defines

- #define rxv_spin_sget(ctx, key) rxv_spin_string(rxv_spin_get((ctx),(key)))
- #define rxv_spin_sset(ctx, key, val) rxv_spin_set((ctx),(key),rxv_spin_datum(rxv_spin_-pool(ctx),(val),NULL))
- #define rxv_spin_del(ctx, key) rxv_spin_set((ctx),(key),NULL)

Typedefs

- typedef struct rxv_spin_ctx rxv_spin_ctx_t

Functions

- apr_pool_t * rxv_spin_pool (rxv_spin_ctx_t *ctx)
- apr_pool_t * rxv_spin_ppool (rxv_spin_ctx_t *ctx)
- rxv_spin_data_t * rxv_spin_data (rxv_spin_ctx_t *ctx)
- request_rec * rxv_spin_r (rxv_spin_ctx_t *ctx)
- apreq_handle_t * rxv_spin_req (rxv_spin_ctx_t *ctx)
- void * rxv_spin_xget (rxv_spin_ctx_t *ctx)
- void * rxv_spin_xset (rxv_spin_ctx_t *ctx, void *extra)
- rxv_spin_data_t * rxv_spin_get (rxv_spin_ctx_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_set (rxv_spin_ctx_t *ctx, const char *key, rxv_spin_data_t *value)

4.2.1 Detailed Description

Context functions (mod_spin API)

4.2.2 Define Documentation

4.2.2.1 #define rxv_spin_sget(ctx, key) rxv_spin_string(rxv_spin_get((ctx),(key)))

get a string from the context by converting from single

4.2.2.2 #define rxv_spin_sset(ctx, key, val) rxv_spin_set((ctx),(key),rxv_spin_datum(rxv_spin_-pool(ctx),(val),NULL))

set a string into context by converting to single

4.2.2.3 #define rxv_spin_del(ctx, key) rxv_spin_set((ctx),(key),NULL)

delete a value from context

4.2.3 Typedef Documentation

4.2.3.1 typedef struct rxv_spin_ctx rxv_spin_ctx_t

context type

4.2.4 Function Documentation

4.2.4.1 apr_pool_t* rxv_spin_pool (rxv_spin_ctx_t * *ctx*)

Retrieve context specific pool.

Parameters:

ctx Context

Returns:

pointer to context specific pool, NULL on error

Example:

```
rxv_spin_pool(ctx);
```

4.2.4.2 apr_pool_t* rxv_spin_ppool (rxv_spin_ctx_t * *ctx*)

Retrieve process specific pool.

Parameters:

ctx Context

Returns:

pointer to process specific pool, NULL on error

Example:

```
rxv_spin_ppool(ctx);
```

4.2.4.3 rxv_spin_data_t* rxv_spin_data (rxv_spin_ctx_t * *ctx*)

Retrieve data from context.

Parameters:

ctx Context

Returns:

pointer to data, NULL on error

Example:

```
rxv_spin_data(ctx);
```

4.2.4.4 request_rec* rxv_spin_r (rxv_spin_ctx_t * *ctx*)

Retrieve Apache request from context.

Parameters:

ctx Context

Returns:

pointer to Apache request, NULL on error

Example:

```
rxv_spin_r(ctx);
```

4.2.4.5 apreq_handle_t* rxv_spin_req (rxv_spin_ctx_t * *ctx*)

Retrieve parsed request from context.

Parameters:

ctx Context

Returns:

pointer to parsed request, NULL on error

Example:

```
rxv_spin_req(ctx);
```

4.2.4.6 void* rxv_spin_xget (rxv_spin_ctx_t * *ctx*)

Retrieve extra data from the context.

Parameters:

ctx Context

Returns:

pointer to extra data, NULL on error

Example:

```
rxv_spin_xget(ctx);
```

4.2.4.7 void* rxv_spin_xset (rxv_spin_ctx_t * *ctx*, void * *extra*)

Set extra data into the context.

Parameters:

ctx Context

extra Pointer to extra data to set into the context

Returns:

pointer to extra data, NULL on error

Example:

```
rxv_spin_xset(ctx,somedata);
```

4.2.4.8 rxv_spin_data_t* rxv_spin_get (rxv_spin_ctx_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, NULL if not found

Example:

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

4.2.4.9 rxv_spin_data_t* rxv_spin_set (rxv_spin_ctx_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, NULL on error

Example:

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

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.

Warning:

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.
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.

4.3 Application and session functions

Defines

- #define rxv_spin_app_sget(ctx, key) rxv_spin_string(rxv_spin_app_get((ctx),(key)))
- #define rxv_spin_app_sset(ctx, key, val) rxv_spin_app_set((ctx),(key),rxv_spin_datum(rxv_spin_pool(ctx),(val),NULL))
- #define rxv_spin_ses_sget(ctx, key) rxv_spin_string(rxv_spin_ses_get((ctx),(key)))
- #define rxv_spin_ses_sset(ctx, key, val) rxv_spin_ses_set((ctx),(key),rxv_spin_datum(rxv_spin_pool(ctx),(val),NULL))

Functions

- rxv_spin_data_t * rxv_spin_app_get (rxv_spin_ctx_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_app_set (rxv_spin_ctx_t *ctx, const char *key, rxv_spin_data_t *val)
- apr_status_t rxv_spin_app_del (rxv_spin_ctx_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ses_get (rxv_spin_ctx_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ses_set (rxv_spin_ctx_t *ctx, const char *key, rxv_spin_data_t *val)
- apr_status_t rxv_spin_ses_del (rxv_spin_ctx_t *ctx, const char *key)
- void rxv_spin_ses_kill (rxv_spin_ctx_t *ctx)
- char * rxv_spin_ses_id (rxv_spin_ctx_t *ctx)
- int rxv_spin_ses_valid (rxv_spin_ctx_t *ctx)
- apr_time_t rxv_spin_ses_atime (rxv_spin_ctx_t *ctx)

4.3.1 Detailed Description

Application and session functions (mod_spin API)

4.3.2 Define Documentation

4.3.2.1 #define rxv_spin_app_sget(ctx, key) rxv_spin_string(rxv_spin_app_get((ctx),(key)))

get a string from application instead of single

4.3.2.2 #define rxv_spin_app_sset(ctx, key, val) rxv_spin_app_set((ctx),(key),rxv_spin_datum(rxv_spin_pool(ctx),(val),NULL))

set string into application instead of single

4.3.2.3 #define rxv_spin_ses_sget(ctx, key) rxv_spin_string(rxv_spin_ses_get((ctx),(key)))

get a string from session instead of single

4.3.2.4 #define rxv_spin_ses_sset(ctx, key, val) rxv_spin_ses_set((ctx),(key),rxv_spin_datum(rxv_spin_pool(ctx),(val),NULL))

set string into application instead of single

4.3.3 Function Documentation

4.3.3.1 rxv_spin_data_t* rxv_spin_app_get (rxv_spin_ctx_t * *ctx*, const char * *key*)

Retrieve a value from the application store.

Parameters:

ctx Context

key Unique key by which this value is identified

Returns:

pointer to the value, NULL on error

Example:

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

4.3.3.2 rxv_spin_data_t* rxv_spin_app_set (rxv_spin_ctx_t * *ctx*, const char * *key*, rxv_spin_data_t * *val*)

Put value in the application store.

Parameters:

ctx Context

key Unique key by which this value is identified

val Value to be placed in the store (single)

Returns:

pointer to the value, NULL on error

Example:

```
rxv_spin_app_set(ctx,"some application key",
                 rxv_spin_datum(pool,"some application value",NULL));
```

Warning:

The data set into the application store 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.

4.3.3.3 apr_status_t rxv_spin_app_del (rxv_spin_ctx_t * *ctx*, const char * *key*)

Delete a record in the application store.

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.

4.3.3.4 rxv_spin_data_t* rxv_spin_ses_get (rxv_spin_ctx_t * *ctx*, const char * *key*)

Retrieve a value from the session store.

Parameters:

ctx Context

key Unique key by which this value is identified

Returns:

pointer to the value, NULL on error

Example:

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

4.3.3.5 rxv_spin_data_t* rxv_spin_ses_set (rxv_spin_ctx_t * *ctx*, const char * *key*, rxv_spin_data_t * *val*)

Put value in the session store.

Parameters:

ctx Context

key Unique key by which this value is identified

val Value to be placed in the store (single)

Returns:

pointer to the value, NULL on error

Example:

```
rxv_spin_ses_set(ctx,"some session key",
                 rxv_spin_datum(pool,"some session value",NULL));
```

Warning:

The data set into the session store 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.

4.3.3.6 apr_status_t rxv_spin_ses_del (rxv_spin_ctx_t * *ctx*, const char * *key*)

Delete a record in the session store.

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.

4.3.3.7 void rxv_spin_ses_kill (rxv_spin_ctx_t * *ctx*)

Kill the session

Parameters:

ctx Context

Example:

```
rxv_spin_ses_kill(ctx);
```

Remarks:

Destroys the session by marking session data for removal from the store. Note that session cookie will still get served. This function would usually be used to log users out of the system, as any session data would have to be reestablished once user logs back in.

4.3.3.8 char* rxv_spin_ses_id (rxv_spin_ctx_t * *ctx*)

Get session id

Parameters:

ctx Context

Returns:

Session id or NULL if no session support exists

Example:

```
rxv_spin_ses_id(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.

4.3.3.9 int rxv_spin_ses_valid (rxv_spin_ctx_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).

4.3.3.10 apr_time_t rxv_spin_ses_atime (rxv_spin_ctx_t * *ctx*)

Last access time for the session.

Parameters:

ctx Context

Returns:

Access time, 0 if session has been killed or on error

Example:

```
rxv_spin_ses_atime(ctx);
```

4.4 Database functions

Functions

- apr_pool_t * rxv_spin_db_pool (rxv_spin_db_t *db)
- char * rxv_spin_db_cinfo (rxv_spin_db_t *db)
- const apr_dbd_driver_t * rxv_spin_db_driver (rxv_spin_db_t *db)
- apr_dbd_t * rxv_spin_db_handle (rxv_spin_db_t *db)
- apr_dbd_transaction_t * rxv_spin_db_txn (rxv_spin_db_txn_t *txn)
- rxv_spin_db_t * rxv_spin_db_open (rxv_spin_ctx_t *ctx, const char *conninfo)
- apr_status_t rxv_spin_db_close (rxv_spin_ctx_t *ctx, rxv_spin_db_t *db)
- apr_status_t rxv_spin_db_status (rxv_spin_ctx_t *ctx, rxv_spin_db_t *db)
- rxv_spin_data_t * rxv_spin_db_data (apr_pool_t *pool, rxv_spin_db_t *db, apr_dbd_results_t *dbdres)
- rxv_spin_data_t * rxv_spin_db_select (apr_pool_t *pool, rxv_spin_db_t *db, const char *query)
- int rxv_spin_db_query (apr_pool_t *pool, rxv_spin_db_t *db, const char *query)
- rxv_spin_data_t * rxv_spin_db_pselect (apr_pool_t *pool, rxv_spin_db_t *db, const char *query,...)
- int rxv_spin_db_pquery (apr_pool_t *pool, rxv_spin_db_t *db, const char *query,...)
- rxv_spin_db_txn_t * rxv_spin_db_start (apr_pool_t *pool, rxv_spin_db_t *db)
- apr_status_t rxv_spin_db_end (rxv_spin_db_txn_t *txn)

4.4.1 Detailed Description

Database functions (mod_spin API)

4.4.2 Function Documentation

4.4.2.1 apr_pool_t* rxv_spin_db_pool (rxv_spin_db_t * *db*)

Retrieve database specific pool.

Parameters:

db Database connection

Returns:

pointer to database specific pool, NULL on error

Example:

```
rxv_spin_db_pool(db);
```

4.4.2.2 char* rxv_spin_db_cinfo (rxv_spin_db_t * *db*)

Retrieve database connection information.

Parameters:

db Database connection

Returns:

pointer to connection information, NULL on error

Example:

```
rxv_spin_db_cinfo(db);
```

4.4.2.3 const apr_dbd_driver_t* rxv_spin_db_driver (rxv_spin_db_t * *db*)

Retrieve database driver.

Parameters:

db Database connection

Returns:

pointer to database driver, NULL on error

Example:

```
rxv_spin_db_driver(db);
```

4.4.2.4 apr_dbd_t* rxv_spin_db_handle (rxv_spin_db_t * *db*)

Retrieve database handle.

Parameters:

db Database connection

Returns:

pointer to database handle, NULL on error

Example:

```
rxv_spin_db_handle(db);
```

4.4.2.5 apr_dbd_transaction_t* rxv_spin_db_txn (rxv_spin_db_txn_t * *txn*)

Retrieve database transaction.

Parameters:

txn Database transaction

Returns:

pointer to database transaction, NULL on error

Example:

```
rxv_spin_db_txn(txn);
```

Remarks:

This function returns underlying Apache Portable Runtime DBD transaction.

4.4.2.6 rxv_spin_db_t* rxv_spin_db_open (rxv_spin_ctx_t * *ctx*, const char * *conninfo*)

Connect to a database and optionally pool the connection.

Parameters:

ctx Context

conninfo Connection string

Returns:

pointer to a database connection, NULL on error

Example:

```
rxv_spin_db_open(ctx,"pgsql:dbname=spintest");
```

Remarks:

Whether or not the connection will be pooled, depends on what has been set as connection pool value in the context. And this depends on the SpinConnPool configuration parameter (default is on). Valid database prefixes are pgsql, mysql, sqlite2, sqlite3, oracle, freetds etc., depending on the drivers that have been compiled and linked with Apache Portable Runtime Utilities Library (APU).
Connections will be pooled by using connection string as a key into the hash. So, if a connection string differs in the amount of white space or case, this will open a new connection.
Cleanup will be registered with the context pool.

4.4.2.7 apr_status_t rxv_spin_db_close (rxv_spin_ctx_t * *ctx*, rxv_spin_db_t * *db*)

Close a database connection.

Parameters:

ctx Context

db Pointer to a database connection

Returns:

APR_SUCCESS on success, otherwise an error

Example:

```
rxv_spin_db_close(ctx,db);
```

4.4.2.8 apr_status_t rxv_spin_db_status (rxv_spin_ctx_t * *ctx*, rxv_spin_db_t * *db*)

Get the status of the connection.

Parameters:

ctx Context

db Database connection

Returns:

APR_SUCCESS if OK, otherwise an error

Example:

```
if(rxv_spin_db_status(ctx,db)!=APR_SUCCESS){
  ...
}
```

Remarks:

If this function returns any code other than success (for valid ctx and db), the database connection should not be used again.

4.4.2.9 rxv_spin_data_t* rxv_spin_db_data (apr_pool_t * *pool*, rxv_spin_db_t * *db*, apr_dbd_results_t * *dbdres*)

Convert APR DBD SQL result set to mod_spin database result.

Parameters:

pool Pool used for memory allocation

db Database connection

dbdres APR DBD results

Returns:

Valid rxv_spin_data_t pointer, NULL on error

Example:

```
rxv_spin_db_data(pool,conn,db,dbdres);
```

Remarks:

Make sure the pool passed into this function is the same one used for select or you will have memory allocation problems.

4.4.2.10 rxv_spin_data_t* rxv_spin_db_select (apr_pool_t * *pool*, rxv_spin_db_t * *db*, const char * *query*)

Execute a database query that returns a result set (i.e. SELECT).

Parameters:

pool Pool used for memory allocation

db Database connection

query SQL query to be performed

Returns:

Valid rxv_spin_data_t pointer, NULL on error

Example:

```
result=rxv_spin_db_select(pool,db,"select * from spintest");
```

Remarks:

If the query finished successfully, returned data will not be NULL, but a pointer to an empty data structure.

4.4.2.11 int rxv_spin_db_query (apr_pool_t * *pool*, rxv_spin_db_t * *db*, const char * *query*)

Execute a database query that doesn't return a result set.

Parameters:

pool Pool used for memory allocation

db Database connection

query SQL query to be performed

Returns:

Number of rows affected, -1 on error

Example:

```
nrows=rxv_spin_db_query(pool,db,"delete from spintest");
```

4.4.2.12 rxv_spin_data_t* rxv_spin_db_pselect (apr_pool_t * *pool*, rxv_spin_db_t * *db*, const char * *query*, ...)

Prepare and execute database that returns a result set (i.e. SELECT)

Parameters:

pool Pool used for memory allocation

db Database connection

query SQL query to be prepared and executed

... Parameters for prepared statement, (constant pointer to nul terminated string)

Returns:

Valid rxv_spin_data_t pointer, NULL on error

Example:

```
result=rxv_spin_db_pselect(pool,db,"select * from names where name = %s",
                           "Dude",NULL);
```

Remarks:

Prepared statements done using this function only accept string parameters. For anything more complicated, use native database API and then convert the data so it can be used within the context. Make sure the number of paramters passed is correct, or you may be in for a nasty surprise when your program segfaults.

Warning:

Make sure you use %s where the replaced parameter is supposed to go. This is the official Apache Portable Runtime DBD way :-).
The last parameter in the list should be NULL and it has to be specified even when there are no other parameters.

4.4.2.13 int rxv_spin_db_pquery (apr_pool_t * *pool*, rxv_spin_db_t * *db*, const char * *query*, ...)

Prepare and execute database that doesn't return a result set.

Parameters:

pool Pool used for memory allocation

db Database connection

query SQL query to be prepared and executed

... Parameters for prepared statement, (constant pointer to nul terminated string)

Returns:

Number of rows affected, -1 on error

Example:

```
nrows=rxv_spin_db_pquery(pool,db,"delete from names where name = %s",
                         "Dude",NULL);
```

Remarks:

Prepared statements done using this function only accept string parameters. For anything more complicated, use native database API and then convert the data so it can be used within the context. Make sure the number of paramters passed is correct, or you may be in for a nasty surprise when your program segfaults.

Warning:

Make sure you use %s where the replaced parameter is supposed to go. This is the official Apache Portable Runtime DBD way :-).
The last parameter in the list should be NULL and it has to be specified even when there are no other parameters.

4.4.2.14 rxv_spin_db_txn_t* rxv_spin_db_start (apr_pool_t * *pool*, rxv_spin_db_t * *db*)

Start a transaction.

Parameters:

pool Pool used for memory allocation

db Database connection

Returns:

Pointer to a valid transaction, NULL on error

Example:

```
txn=rxv_spin_db_start(pool,db);
```

Remarks:

The pool is also used to register a cleanup function for the transaction. It makes little sense to have transactions more permanent than one request, so this should always be the request pool, which can be obtained from the context.

4.4.2.15 apr_status_t rxv_spin_db_end (rxv_spin_db_txn_t * *txn*)

End a transaction.

Parameters:

txn Database transaction

Returns:

APR_SUCCESS on success, otherwise an error

Example:

```
rxv_spin_db_end(txn);
```

4.5 Connection functions

Functions

- void * rxv_spin_conn_get (rxv_spin_ctx_t *ctx, const char *conninfo)
- void * rxv_spin_conn_set (rxv_spin_ctx_t *ctx, const char *conninfo, void *conn, apr_status_t(*cleanup)(void *data))

4.5.1 Detailed Description

Connection functions (mod_spin API)

4.5.2 Function Documentation

4.5.2.1 void* rxv_spin_conn_get (rxv_spin_ctx_t * *ctx*, const char * *conninfo*)

Get a registered connection from the connection pool.

Parameters:

ctx Context

conninfo Connection string for this connection

Returns:

Pointer to the connection or NULL if not found

Example:

```
rxv_spin_conn_get(ctx,"openldap:ldap://ldap.example.com/dc=example,dc=com");
```

4.5.2.2 void* rxv_spin_conn_set (rxv_spin_ctx_t * *ctx*, const char * *conninfo*, void * *conn*, apr_status_t(*)(void *data) *cleanup*)

Register a connection with the connection pool.

Parameters:

ctx Context

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_conn_set(ctx,"openldap:ldap://host.domain/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_conn_get() function to verify if the connection was registered before.

Warning:

If you decide to close the connection by hand, you must remove the cleanup function from the thread specific pool (you can find out what that is by calling rxv_spin_tpool()).
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) or the connect string, 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.

4.6 Application entry functions

Typedefs

- typedef void(* rxv_spin_init_t)(rxv_spin_ctx_t *ctx)
- typedef int(* rxv_spin_prepare_t)(rxv_spin_ctx_t *ctx)
- typedef int(* rxv_spin_service_t)(rxv_spin_ctx_t *ctx)

4.6.1 Detailed Description

Application entry functions (mod_spin API)

4.6.2 Typedef Documentation

4.6.2.1 typedef void(∗ rxv_spin_init_t)(rxv_spin_ctx_t ∗ctx)

init function

4.6.2.2 typedef int(∗ rxv_spin_prepare_t)(rxv_spin_ctx_t ∗ctx)

prepare function

4.6.2.3 typedef int(∗ rxv_spin_service_t)(rxv_spin_ctx_t ∗ctx)

service function

4.7 DSO loading functions

Functions

- apr_status_t rxv_spin_dso_load (rxv_spin_ctx_t ∗ctx, const char ∗path, apr_dso_handle_t ∗∗handle)

4.7.1 Detailed Description

DSO loading functions (mod_spin API)

4.7.2 Function Documentation

4.7.2.1 apr_status_t rxv_spin_dso_load (rxv_spin_ctx_t ∗ *ctx*, const char ∗ *path*, apr_dso_handle_t ∗∗ *handle*)

Load a DSO into the cache.

Parameters:

ctx Context

path Full path of the DSO to load

handle DSO handle

Returns:

APR_SUCCESS on successful load, otherwise an error

Example:

```
rxv_spin_dso_load(ctx,"/path/to/dso/object.so",&handle);
```

Remarks:

To preserve pool sanity, this function copies path into the memory allocated from the private pool.

4.8 Crypto functions

Functions

- char * rxv_spin_hash (apr_pool_t *pool, const char *uniq)
- char * rxv_spin_hmac (apr_pool_t *pool, const char *uniq, const char *salt)

4.8.1 Detailed Description

Crypto functions (mod_spin API)

4.8.2 Function Documentation

4.8.2.1 char* rxv_spin_hash (apr_pool_t * *pool*, const char * *uniq*)

Create MD5 hash, using ASCII letters only

Parameters:

pool Pool for allocation the hash

uniq String to hash

Returns:

MD5 hash of the string, 32 bytes long, encoded [a-p]

Example:

```
rxv_spin_hash(pool,"some string");
```

4.8.2.2 char* rxv_spin_hmac (apr_pool_t * *pool*, const char * *uniq*, const char * *salt*)

Create MD5 HMAC, using ASCII letters only

Parameters:

pool Pool for allocation the hash

uniq String to hash

salt Key used for HMAC, must be 64 bytes long

Returns:

MD5 HMAC of the string, 32 bytes long, encoded [a-p]

Example:

```
rxv_spin_hmac(pool,"some string",verysecretkey);
```

5 File Documentation

5.1 src/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_strings.h>
#include <apr_pools.h>
#include <apr_hash.h>
#include <apr_time.h>
#include <apr_dso.h>
#include <apr_buckets.h>
#include <apr_dbd.h>
#include <apreq_param.h>
#include <apreq_cookie.h>
#include <apreq_module.h>
#include <httpd.h>
#include <http_request.h>
#include <ap_regex.h>
```

Defines

- #define **RXV_SPIN_STDC_HEADERS** 1
- #define **RXV_SPIN_HAVE_UNISTD_H** 1
- #define **RXV_SPIN_HAVE_SYS_TYPES_H** 1
- #define **RXV_SPIN_HAVE_APR_STRINGS_H** 1
- #define **RXV_SPIN_HAVE_APR_POOLS_H** 1
- #define **RXV_SPIN_HAVE_APR_HASH_H** 1
- #define **RXV_SPIN_HAVE_APR_TIME_H** 1
- #define **RXV_SPIN_HAVE_APR_DSO_H** 1
- #define **RXV_SPIN_HAVE_APR_BUCKETS_H** 1
- #define **RXV_SPIN_HAVE_APR_DBD_H** 1
- #define **RXV_SPIN_HAVE_APREQ_PARAM_H** 1
- #define **RXV_SPIN_HAVE_APREQ_COOKIE_H** 1
- #define **RXV_SPIN_HAVE_APREQ_MODULE_H** 1
- #define **RXV_SPIN_HAVE_HTTPD_H** 1
- #define **RXV_SPIN_HAVE_HTTP_REQUEST_H** 1

- #define **RXV_SPIN_HAVE_AP_REGEX_H** 1
- #define rxv_spin_multi(d) (!rxv_spin_string(d) && rxv_spin_size(d)>0)
- #define rxv_spin_single(d) (!rxv_spin_multi(d))
- #define rxv_spin_strim2(s) rxv_spin_strim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_-RIGHT))
- #define rxv_spin_striml(s) rxv_spin_strim((s),(RXV_SPIN_TRIM_LEFT))
- #define rxv_spin_strimr(s) rxv_spin_strim((s),(RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_lower(s)
- #define rxv_spin_upper(s)
- #define rxv_spin_trim(s, w)
- #define rxv_spin_trim2(s) rxv_spin_trim((s),(RXV_SPIN_TRIM_LEFT|RXV_SPIN_TRIM_-RIGHT))
- #define rxv_spin_triml(s) rxv_spin_trim((s),(RXV_SPIN_TRIM_LEFT))
- #define rxv_spin_trimr(s) rxv_spin_trim((s),(RXV_SPIN_TRIM_RIGHT))
- #define rxv_spin_sget(ctx, key) rxv_spin_string(rxv_spin_get((ctx),(key)))
- #define rxv_spin_sset(ctx, key, val) rxv_spin_set((ctx),(key),rxv_spin_datum(rxv_spin_-pool(ctx),(val),NULL))
- #define rxv_spin_del(ctx, key) rxv_spin_set((ctx),(key),NULL)
- #define rxv_spin_app_sget(ctx, key) rxv_spin_string(rxv_spin_app_get((ctx),(key)))
- #define rxv_spin_app_sset(ctx, key, val) rxv_spin_app_set((ctx),(key),rxv_spin_datum(rxv_spin_-pool(ctx),(val),NULL))
- #define rxv_spin_ses_sget(ctx, key) rxv_spin_string(rxv_spin_ses_get((ctx),(key)))
- #define rxv_spin_ses_sset(ctx, key, val) rxv_spin_ses_set((ctx),(key),rxv_spin_datum(rxv_spin_-pool(ctx),(val),NULL))

Typedefs

- typedef struct rxv_spin_data rxv_spin_data_t
- typedef struct rxv_spin_curs rxv_spin_curs_t
- typedef struct rxv_spin_ctx rxv_spin_ctx_t
- typedef struct rxv_spin_db rxv_spin_db_t
- typedef struct rxv_spin_db_txn rxv_spin_db_txn_t
- typedef void(∗ rxv_spin_init_t)(rxv_spin_ctx_t ∗ctx)
- typedef int(∗ rxv_spin_prepare_t)(rxv_spin_ctx_t ∗ctx)
- typedef int(∗ rxv_spin_service_t)(rxv_spin_ctx_t ∗ctx)

Enumerations

- enum rxv_spin_trim_e { RXV_SPIN_TRIM_LEFT = 1, RXV_SPIN_TRIM_RIGHT }

Functions

- rxv_spin_data_t ∗ rxv_spin_datum (apr_pool_t ∗pool, const char ∗str, rxv_spin_data_t ∗data)
- rxv_spin_data_t ∗ rxv_spin_mdatum (apr_pool_t ∗pool, const char ∗str, apr_size_t size, rxv_spin_-data_t ∗data)
- char ∗ rxv_spin_string (rxv_spin_data_t ∗single)
- apr_hash_t ∗ rxv_spin_guts (rxv_spin_data_t ∗rows)
- apr_size_t rxv_spin_size (rxv_spin_data_t ∗data)
- rxv_spin_data_t ∗ rxv_spin_column (apr_pool_t ∗pool, const char ∗name, rxv_spin_data_t ∗data,...)

- rxv_spin_data_t * rxv_spin_parse (apr_pool_t *pool, const char *name, char *str, const char *sep, rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_array (apr_pool_t *pool, const char *name, apr_array_header_t *arr, rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_brigade (apr_pool_t *pool, const char *name, apr_bucket_brigade *bb, rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_null (apr_pool_t *pool, const char *name, apr_size_t size, rxv_spin_data_t *data)
- rxv_spin_data_t * rxv_spin_rows (apr_pool_t *pool, rxv_spin_data_t *data,...)
- apr_size_t rxv_spin_first (apr_pool_t *pool, rxv_spin_data_t *rows,...)
- apr_size_t rxv_spin_next (rxv_spin_curs_t *curs)
- rxv_spin_data_t * rxv_spin_this (rxv_spin_curs_t *curs)
- rxv_spin_data_t * rxv_spin_entry (rxv_spin_data_t *rows, const char *name, apr_size_t index)
- rxv_spin_data_t * rxv_spin_resize (rxv_spin_data_t *data, apr_size_t size)
- rxv_spin_data_t * rxv_spin_copy (apr_pool_t *pool, rxv_spin_data_t *src, rxv_spin_data_t *data)
- char * rxv_spin_slower (const char *str)
- char * rxv_spin_supper (const char *str)
- char * rxv_spin_strim (const char *str, rxv_spin_trim_e what)
- apr_pool_t * rxv_spin_pool (rxv_spin_ctx_t *ctx)
- apr_pool_t * rxv_spin_ppool (rxv_spin_ctx_t *ctx)
- rxv_spin_data_t * rxv_spin_data (rxv_spin_ctx_t *ctx)
- request_rec * rxv_spin_r (rxv_spin_ctx_t *ctx)
- apreq_handle_t * rxv_spin_req (rxv_spin_ctx_t *ctx)
- void * rxv_spin_xget (rxv_spin_ctx_t *ctx)
- void * rxv_spin_xset (rxv_spin_ctx_t *ctx, void *extra)
- rxv_spin_data_t * rxv_spin_get (rxv_spin_ctx_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_set (rxv_spin_ctx_t *ctx, const char *key, rxv_spin_data_t *value)
- rxv_spin_data_t * rxv_spin_app_get (rxv_spin_ctx_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_app_set (rxv_spin_ctx_t *ctx, const char *key, rxv_spin_data_t *val)
- apr_status_t rxv_spin_app_del (rxv_spin_ctx_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ses_get (rxv_spin_ctx_t *ctx, const char *key)
- rxv_spin_data_t * rxv_spin_ses_set (rxv_spin_ctx_t *ctx, const char *key, rxv_spin_data_t *val)
- apr_status_t rxv_spin_ses_del (rxv_spin_ctx_t *ctx, const char *key)
- void rxv_spin_ses_kill (rxv_spin_ctx_t *ctx)
- char * rxv_spin_ses_id (rxv_spin_ctx_t *ctx)
- int rxv_spin_ses_valid (rxv_spin_ctx_t *ctx)
- apr_time_t rxv_spin_ses_atime (rxv_spin_ctx_t *ctx)
- apr_pool_t * rxv_spin_db_pool (rxv_spin_db_t *db)
- char * rxv_spin_db_cinfo (rxv_spin_db_t *db)
- const apr_dbd_driver_t * rxv_spin_db_driver (rxv_spin_db_t *db)
- apr_dbd_t * rxv_spin_db_handle (rxv_spin_db_t *db)
- apr_dbd_transaction_t * rxv_spin_db_txn (rxv_spin_db_txn_t *txn)
- rxv_spin_db_t * rxv_spin_db_open (rxv_spin_ctx_t *ctx, const char *conninfo)
- apr_status_t rxv_spin_db_close (rxv_spin_ctx_t *ctx, rxv_spin_db_t *db)
- apr_status_t rxv_spin_db_status (rxv_spin_ctx_t *ctx, rxv_spin_db_t *db)
- rxv_spin_data_t * rxv_spin_db_data (apr_pool_t *pool, rxv_spin_db_t *db, apr_dbd_results_t *dbdres)
- rxv_spin_data_t * rxv_spin_db_select (apr_pool_t *pool, rxv_spin_db_t *db, const char *query)
- int rxv_spin_db_query (apr_pool_t *pool, rxv_spin_db_t *db, const char *query)
- rxv_spin_data_t * rxv_spin_db_pselect (apr_pool_t *pool, rxv_spin_db_t *db, const char *query,...)

- int rxv_spin_db_pquery (apr_pool_t *pool, rxv_spin_db_t *db, const char *query,...)
- rxv_spin_db_txn_t * rxv_spin_db_start (apr_pool_t *pool, rxv_spin_db_t *db)
- apr_status_t rxv_spin_db_end (rxv_spin_db_txn_t *txn)
- void * rxv_spin_conn_get (rxv_spin_ctx_t *ctx, const char *conninfo)
- void * rxv_spin_conn_set (rxv_spin_ctx_t *ctx, const char *conninfo, void *conn, apr_status_t(*cleanup)(void *data))
- apr_status_t rxv_spin_dso_load (rxv_spin_ctx_t *ctx, const char *path, apr_dso_handle_t **handle)
- char * rxv_spin_hash (apr_pool_t *pool, const char *uniq)
- char * rxv_spin_hmac (apr_pool_t *pool, const char *uniq, const char *salt)

5.1.1 Detailed Description

mod_spin data structures and functions

5.1.2 Typedef Documentation

5.1.2.1 typedef struct rxv_spin_db rxv_spin_db_t

database connection type

5.1.2.2 typedef struct rxv_spin_db_txn rxv_spin_db_txn_t

database transaction type

Index