RIASSENCE FRAMEWORK

Published : 2011-03-11
License : None

1. INTRODUCTION_

The Riassence Core Framework (or casually, just "RSence") is a Rich Internet Application Framework with the defining characteristics:

- Thick Client
- Thin Server

THICK CLIENT_

The Riassence Core client framework is a relatively small, independent javascript package that communicates asynchronously with the server. The client is able to load resources on the fly and needs the server essentially just for bootstrapping, session management and data handling. The reasoning is that modern web browsers are fairly powerful universal layout engines with a safe, sand-boxed environment for running client applications. One major factor of a good user experience is responsiveness. To achieve that, the user interface needs to be controlled where it's displayed. A well written Riassence Core application provides realtime responsiveness and has the same user interface characteristics of a typical desktop application.

The desktop metaphor for user interfaces is the established way to develop client software. To do it on the web used to be a fairly involved process. One of the first approaches were Java Applets, but these had a tendency of long startup times and didn't integrate too well with the rest of the web. Flash applications are also separate sandboxes in the web page and as such doesn't integrate too well or leverage the capabilities of the browser itself. The Shockwave Flash -plugin is also proprietary and it's not available on all platforms.

Javascript is enabled on almost all web browsers, but used to be a difficult platform to build applications on. Writing a desktop-like Javascript GUI support framework is not an easy task either and then there are all those dreaded browser incompatibilities. The Riassence client framework addresses these issues and has been under active development since early 2006. It supports 99% of the web browsers out there, including Internet Explorer versions 6 through 8, Firefox, Chrome, Opera, Safari and the majority of mobile browsers in use.

One of the core building blocks is the HClass -class object that enables proper, inherited Object-Oriented programming in Javascript by building on the prototyped OO model Javascript has natively. It allows a high grade of code re-usability with minimal effort.

Another core feature is the ELEM interface. It provides a controlled, double-buffered, cross-browser engine for DOM manipulation.

Desktop-like events are provided by the EVENT interface. It abstracts event handling by binding the low-level event listeners and provides the higher level event interface used by the UI framework. Such high-level events include dragging, dropping, clicking, typing, key combinations etc.

The COMM package provides high-level data transfer capabilities and automatic client-server session handshaking and transparent, asynchronous data transfers between client and server.

HSystem is the "Kernel" for process and view control. It provides multitasking for processes with features like starting, stopping, destruction and prioritization.

HApplication is the basic process model and the root level controller of an client-side application. Destructing an HApplication instance destructs its UI too.

The HThemeManager service supports loading theme files (markup and css) on the fly. Automatically and transparently.

HRect and HPoint define a high-level coordinate model and include a lot of geometry calculations such as intersections, offsets and manipulation of shapes by any edge or corner.

The HView class is the high-level foundation building block for building user interfaces. Any class that inherits HView gets a high-level UI building block with themeability, animation and sub-view support for free. HViews define the bounds of all UI components.

The HControl class extends view with support for events, labels, values and a multitude of options. All interactive UI components extends HControl.

Riassence Core comes bundled with a full set of essential UI components: HButton, HCheckbox, HImageView, HPasswordControl, HRadioButton, HSplitView, HSlider, HVSlider, HStepper, HTextControl, HTextArea, HValidatorView, HProgressbar, HUploader, HStringView, HProgressIndicator, HTab and HWindow.

These components share the API of HControl and are useable as-is. They are also easily extendable for any specific tasks.

THIN SERVER_

Although you can run the client independently without the server, the server is necessary for building complex web applications and doing tasks not allowed in the web browser's sandbox. As the client can handle the user interface independently, the server does what a server needs to do, and does it efficiently.

The main responsibilities of the Riassence server is to avoid tasks the client is capable of and emphasize on the tasks the client can't do.

Session management is one of these tasks. Sessions are needed to tell the clients apart and to keep the data shared by the client and server intact, secure and up to date. Sessions also have a persistent database back-end where the session data is stored. This allows uninterrupted operation of the client even if the server becomes unreachable for a while. Session keys are by default disposable and use cryptographic hashing (SHA1) for key exchange. Data of concurrent sessions are kept separate from each other.

Another important duty of the server is to bridge the sand-boxed client environment with the I/O capabilities of the server. Some of these capabilities are access to databases, the filesystem, external interfaces and tasks that needs to be done securely.

The server-side controller model is the Plugin. The plugin does tasks like decide which user interfaces to launch, defines session data, responds to data changes etc. It's the necessary glue between back-end and GUI.

Plugins are deployed by dropping a plugin bundle into the plugins directory and un-deployed by dragging it out of the plugins directory. In debug/development mode, plugins are automatically reloaded. Each plugin is initialized as a single instance and user interaction is managed through evented messages. Messages are light-weight, disposable objects that provide the request/response cycle, access to system-wide interfaces and session-bound data.

DATA MANAGEMENT_

Client-server data is managed using the DSM (Distributed Shared Memory) approach. Values are smart data container objects that are bound to other objects. When the data changes, each bound objects synchronized with notification events. These events are automatically distributed to both server and client member objects. Synchronization is transparent and triggered automatically in both directions. This makes data handling very simple, deprecates the need of explicit notification and complex routing schemes.

Currently supported ballast value types are:

- Boolean
- Number (integer and float)
- String
- Flat array (containing any combinations of data types listed above)

2. SOMEBODY SHOULD SET THE TITLE FOR THIS CHAPTER!

This guide is also available in HTML at http://rsence.org/trac/wiki/QuickStart

INSTALLATION

1. INSTALLING SYSTEM DEPENDENCIES

1.1. Mac OS X

- Install a recent version of [http://developer.apple.com/mac/ XCode] to get at least the gnu compiler collection.

- Install the most recent version of [http://www.macports.org/ MacPorts]

- Update rubygems to the most recent version, unless you do it regularly anyway:

```
---
  sudo gem update --system
  sudo gem update
---
```

1.1.1 Mac OS X 10.4 and 10.5

- Install a these packages using Terminal:

```
---
  sudo port install ruby +thread_hooks
  sudo port install rb-rubygems
  sudo port install sqlite3-ruby
---
```

1.1.2 Mac OS X 10.6

- Install a these packages using Terminal:

```
---
  sudo port install sqlite3
---
```

1.2. Debian and Ubuntu GNU Linux

- Install these packages:

```
---
  sudo apt-get install build-essential ruby-full libsqlite3-ruby rubygems rake
---
```

1.99. Unsupported UNX-like operating system
(Applies to Linux distributions not listed above)

You'll have to figure out how to install the dependencies on your own,
but generally this is the list of software you should look for:

- Ruby 1.8.7 or newer

- Rake (ruby replacement for make)

- Sqlite

- Ruby Gems

- A standard set of compilers and build tools

2. RUBY PACKAGES (RUBY GEMS)

These ruby gems are required:

- rake (not required, if you have a system package of it installed)

- rack

- thin (or another rack-compatible http server library)

- json

- Required by persistent session storage:

  - sqlite3-ruby

  - sequel

These ruby gems are recommended. The system will run without them but manual reconfiguration might be needed.

- cssmin

- highline (not required, if you don't need the setup wizard)

- soap4r (not required, if you don't need to create SOAP services)

- rmagick (not required, but used by many projects)

2.1. Command-line installation example of the gems

```
---
  sudo gem install rake highline rack thin soap4r rmagick json sequel
---
```

3. SETTING UP A PROJECT

3.1. Create a directory for your project

This directory will contain everything needed to run your application.

```
mkdir my_riassence_project
cd my_riassence_project
mkdir conf # optional, place additional configuration files here
mkdir plugins # optional, place your local plugins here
mkdir lib # optional, place your libraries here
mkdir js # optional, create your JavaScript components here
```

3.2. Make a checkout of the Riassence Framework

- You'll need an directory called "rsence" in your project directory
  that contains the framework.

- There are three main options: release, stable, trunk. Choose one of these:
  * release: latest release version, recommended for production deployment
  * stable: the latest stable version (constant release candidate),
    recommended for production development
  * trunk: the latest bleeding edge version, recommended for developing the
    framework itself or testing out the newest features

3.2.1. The latest release version

- Download http://rsence.org/files/releases/rsence_1.1.0.tar.bz2

- Extract its contents into the project directory

3.2.2. The latest stable version

- Make a checkout of /branches/rc as rsence:

```
svn co https://svn.rsence.org/branches/rc rsence
```

3.2.3. The trunk

- Make a checkout of /trunk as rsence:

```
svn co https://svn.rsence.org/trunk rsence
```

3.3. Setup the Riassence Framework

Currently, a few additional steps are needed to get the environment set up.
In a future version, we'll simplify the whole process by just including
rsence as a gem.

3.3.1. Make the binaries executable

You may skip this step, if you downloaded the release version.
This makes the rsence process control scripts executable.

```
  chmod +x rsence/bin/*
```

3.3.2. Build the C-optimized extensions

Build the C-optimized extensions:

```
  rsence/bin/build_extensions.rb
```

3.3.3. Build the Javascript client library packages

This will create a directory named "client" in your project directory. It
contains a set of optimized Javascript and theme packages required by the
Riassence Client Framework.

```
  rsence/bin/build_client.rb
```

3.3.4. Start the server and run the configuration wizard
(doesn't apply to preconfigured projects)

3.3.4.1. Start the server in the more verbose development mode (the -d switch)

```
  rsence/bin/start -d
```

3.3.4.2. Answer the setup questions

- Press enter for each question to choose the default option.

- Eg. pressing enter at a Y/n prompt will select Y

3.3.4.3. Open your web browser to check if the server responds

- On a default installation, the riassence server framework listens to port 8001

- To test it, open the address [http://localhost:8001/]

- A brief "Loading, please wait..." -message, followed by the plugin ui's configured to run is a sign of success.

3.3.4.4. Inspect the log files

- In case of installation issues, check the log files.

- The log files are by default in the '"rsence/var/log"' directory.

3.3.4.5. Stop the server

```
---
 rsence/bin/stop
---
```

4. CONTROLLING THE RIASSENCE SERVER FRAMEWORK PROCESS

4.1. Starting the server

- Either use the "rsence/bin/start" -command or use "rsence/bin/launch.rb start"

- By default, it will look for a settings file "rsence/conf/local_conf.rb"

- If a settings file is not found, the setup wizard will run; see section 3.3.4

4.2. Starting the server

- Use the "rsence/bin/stop" -command

4.3. Checking the server status

- Use the "rsence/bin/status" -command

4.4. Restarting the server

- Use the "rsence/bin/restart" -command

4.5. Re-setting the sessions

- Start or restart the server with the "--reset-sessions" switch

4.6. Running in development mode

- Start or restart the server with the "-d" switch

4.6.1. What does development mode do?

- Plugins are reloaded for each index-page load (every time when the browser reloads the "/ -page")

- Built Javascript packages are reloaded, if changed.

- More verbose log output is generated

4.7. More command-line options

---
rsence/bin/launch.rb help
---

5. PLUGIN DEPLOYMENT

If you followed, the previous steps, you are ready to deploy some software.

- In development mode (see 4.6.), plugins are (re)loaded on each page reload

- In production mode, a server restart is required (see 4.4.)

- By default, "rsence/plugins" is the only plugin directory. Edit the configuration file to enable other directories.

- Sample plugins are available in the repository URL directory https://svn.rsence.org/contrib/plugins

5.1. Deploying plugins

- Copy or move a plugin bundle into the "plugins" directory to enable the plugin.

5.2. Un-deploying plugins

- Copy or move a plugin bundle out of the the "plugins" directory.

5.3. Temporarily disabling a plugin

- Create an empty file named "disabled" in the plugin bundle to disable it.

---
touch rsence/plugins/componentsampler/disabled
---

6. PLUGIN DEVELOPMENT

6.1. The Hello World -plugin

6.1.1. Create a plugin bundle

- Create a directory named "hello_world" inside the "rsence/plugins" directory.

- Create a ruby source file inside the "hello_world" directory named "hello_world.rb".

- The ruby source file (without the .rb extension) must match the name of the directory for the bundle to be recognized.

6.1.2. Display a "Hello World" message using the Javascript client framework

Copy/paste this ruby source into the "hello_world.rb" file created in section 6.1.1:

```
class HelloWorld < Plugin
  # Let's say hello to the web browser by just replying some javascript:
  def say_hello(msg, who)
    include_js( msg, ['controls'] )
    msg.reply "var helloApp = HApplication.nu();"
    msg.reply "var helloWinRect = HRect.nu(20,20,320,60);"
    msg.reply "helloApp.helloWin = HWindow.nu(helloWinRect, helloApp, {label: 'Hello, #{who}!'});"
  end
  def init_ui(msg)
    # Let's say hello, when the bootstrap is done:
    say_hello(msg, 'World')
  end
end
HelloWorld.new.register('hello_world')
```

6.2. The Visitor Counter Plugin bundle

Read the tutorial at: http://rsence.org/trac/wiki/Tutorials/VisitorCounter

3. CONFIGURATION

4. TROUBLESHOOTING

TUTORIAL
5. Tutorial

5. TUTORIAL

```
Step 1: Initializing a RSence project.

Welcome to the RSence Chat Tutorial.

In this tutorial you will learn the basics of RSence development.

You will need three applications to develop a project:
 - A text editor suitable for writing source code.
 - A command shell to control the process and initialize your environment.
 - A web browser to test your application.

Start by creating a new RSence project in your project folder.

In this case, we use a folder named "tutorial".

To initialize a new RSence project, in this case a chat application we'll use
for tutorial purposes, type this in your terminal application:
 rsence init chat_app

The init wizard will ask you a few basic questions about your project setup:
 - The first question is "Project Title":
   - Just enter any name for your project, like "Chat Application"
   - That name will be shown in the title bar of your web browser.
 - The second question is the database connection string.
   - The database is used as session storage, enabling persistent session
     data storage even when RSence is not running. It's also used for
     temporary storage of file uploads.
   - If you have sqlite installed, just press enter. It will work by default.
   - Any database supported by Sequel is supported by RSence too.
 - The third question is the TCP port of the RSence web server.
   - Only one process at a time can use a specific port, so be sure to enter
     one not used by other processes on the same computer.
   - In most cases, the default 8001 is fine, so just press enter.
 - The fourth question is the TCP/IP address to bind the HTTP server to.
   - The default "0.0.0.0" address binds the HTTP Server to all configured
     network interfaces. Just press enter to select the default.
   - Another safe choice is "127.0.0.1", which is the localhost interface.
     That interface is accessible only to the computer itself.
     We will use the localhost interface in this tutorial.
 - The fifth question is the HTTP Server URI prefix which is the "directory"
   RSence will respond to. The default "/" works fine in most cases, so just
   press enter to continue.
   - NOTICE: RSence does not serve any actual files or directories by default.
             The URI (and complete URL) configured by default is an access
             hook to in-memory objects of the RSence process, which will call
             installed plugin objects.
 - The sixth question is just a confirmation of the basic configuration
   parameters. Press enter to accept or press "n" followed by enter to
   re-enter the questions.
 - Finally, RSence asks whether or not to install the "welcome" plugin.
   Just press enter to install it. We will use it as a template for this
   tutorial.

---

Step 2: Inspect the project.

As a result of the project init, RSence creates a directory to hold your
application. Open this directory and look let's review which file and folder
is what.

The "conf" directory contains by default a configuration file named
"config.yaml". This configuration file contains the result of the answers
to the questions asked by the "init" command. Modify the configuration file
to set any advanced options or to change some of the basic settings.
For development, enter these two lines at the bottom of the configuration
file and save it:

:transporter_conf:
  :client_autoreload: true

This tells the RSence process to reload connected web browser clients
automatically, whenever RSence detects changes in the source files of the
project plugin bundles. It's a great convenience while writing applications,
because you do not have to reload the web browser manually between changes.
```

```
The "db" directory contains the sqlite session database. The directory is
unused, if you don't use a session storage database (it's optional) or if
you use another database engine and its associated ruby database driver
libraries.

The "log" directory contains the standard output and standard error log
files, unless configured otherwise.

The "run" directory contains process identification files.

The "plugins" directory contains all installed RSence software. RSence is able
runs by default with a few support plugins, but they do nothing particularly
interesting by their own, except initializing the RSence client when
web browsers access the main URL.

If you didn't answer "n" to the final question of the project init command,
the "plugins" directory now contains a copy of the "welcome" plugin.

It's a simple plugin that displays a greeting when you start RSence.

Lets look at its structure:

 - It's in a directory that defines its name. It's accessible by other
   installed plugins via the plugin managemnt system by that name.
 - It contains a ruby source code file named "welcome.rb".
   - If a ".rb" file is found matching the name of the directory, the
     directory is identified as a "Plugin Bundle" and the ruby source
     is loaded into memory. If the source file contains a class inherited
     from a supported Plugin interface, like Plugin, Servlet or GUIPlugin,
     it's also automatically initialized by the plugin management system.
   - If the ruby source file contains a class inherited from either
     GUIPlugin or Plugin, it will look for a file called "values.yaml".
     That file contains client-server -synchronized value container object
     specifications. More about this later.
   - If the ruby source file contains a class inherited from GUIPlugin,
     like the "welcome" plugin is, it will look for a ".yaml" file named
     similarly to the bundle directory name in the "gui" subdirectory.
     In this case, the file is named "gui/welcome.yaml". The file contains
     a "GUITree" structure, which is used for user interface rendering
     instructions. More about this later.
   - There is also a file called "info.yaml", which contains basic
     plugin-specific meta-data, such as the name, description, version
     and dependencies of the plugin bundle.

---

Step 3: Run the project

RSence supports several different ways to run a RSence server process.
In this case, we will use the "run" command, which is supported on all
platforms, including Windows. Background daemon processes require a
posix-compatible operating system, such as Mac OS X and Linux, so we'll
focus on the "run" command here.

There are various options for each command. To list all commands, run the
following command at your command line:

 rsence help

It will list all available commands on your platform. To get more information
about a specific command, such as the "run" command, do this:

 rsence help run

It will display help for the various options available for that command.
```

6. RUBYN PERUSTEET

7. JAVASCRIPT BASICS

8. YAML BASICS

TUTORIALS
9. Visitor Counter

9. VISITOR COUNTER

INTRODUCTION

Plugins are bundled in Riassence Framework. This means that the basic structure of a plugin is the same in every one. More information about the bundling can be found later from the documentation. This tutorial will however introduce the basic structure of plugins.

```
|-plugin.rb        #The server side code written in Ruby
|-js/              #UI code resides here
|-- plugin.js      #UI code written in Javascript
```

Ruby is used in the server side logic. More information about the Riassence Framework can be found from here. Javascript is used to create user interface. Values between JS and Ruby can be synchronized automatically, but JS is used solely for user interface purposes.

On this example we will build a plugin which counts page loads.

STEP 1

First let's make an empty plugin, which doesn't do anything by itself. This will however show the bare minimum which a plugin needs:

```
# basic Plugin class extension
class Counter < Plugin
# nothing here yet!
end
# registers Counter as 'counter'
Counter.new.register( 'counter' )
```

This will register the PluginManager class. From now on you can access the data of this plugin instance with the name you provide:

```
Counter.new.register( 'counter' )
```

In this case the name will be 'counter'

STEP 2

There are multiple methods to extend in a plugin class. Let's add support for bunch of methods to get some functionality:

```
# basic Plugin class extension
class Counter < Plugin
# primary initialization routine of the whole plugin, runs only once.
  def init

  end

# session initialization, whenever a new client is detected this method will be invoked
  def init_ses( msg )

  end

# restore session will be invoked when an old session in client is detected
  def restore_ses( msg )

  end

# runs every time during the synchronization (every time the server talks with the client)
  def idle( msg )

  end

# invoked once per page load when the framework has been initialized
```

```
  def init_ui( msg )

  end

end
# registers Counter as 'counter'
Counter.new.register( 'counter' )
```

This will add support for plugin initialization, session management, idle processes and user interface. This is sufficient for the visitor counter. Check the source code of Plugin for complete set of methods which can be extended.

STEP 3

Let's begin building the class by adding one instance variable. @visits_total will tell the absolute number of visitors on the page.

```
# basic Plugin class extension
class Counter < Plugin
  # primary initialization routine of the whole plugin, runs only once.
  def init
    #total number of visits on the page
    @visits_total = 0
  end

  # session initialization, whenever a new client is detected this method will be invoked
  def init_ses( msg )
    # add one to total visits
    msg.console("Hello, this is your first visit!")
    @visits_total += 1
  end

  # restore session will be invoked when an old session in client is detected
  def restore_ses( msg )
    #add one to total visits
    msg.console("Welcome back, this is visit number #{@visits_total}!"
    @visits_total += 1

  end

  # runs every time during the synchronization (every time the server talks with the client)
  def idle( msg )

  end

  # invoked once per page load when the framework has been initialized
  def init_ui( msg )

  end
end
# registers Counter as 'counter'
Counter.new.register( 'counter' )
```

Counting visitors can be easily done be extending the methods `init_ses( msg )` and `restore_ses( msg )`. These methods will be executed automatically upon the client connection. Another trick worth to note is that you can easily send information, for example for debugging purposes, by using `msg.console( String )` notation. This will send a message directly to the browser's console. Firebug is a cool debugging tool you'll want to install with Firefox if you don't have it already.

STEP 4

Some further additions will follow. Let's add

```
# basic Plugin class extension
class Counter < Plugin
  # primary initialization routine of the whole plugin, runs only once.
  def init
    # total hits are recorded as instance variable
    @hits_total = 0

    # so are visits, both unique and all visits
    @visits_total = 0
    @visits_unique = 0
```

```
  end

  # session initialization, whenever a new client is detected this method will be invoked
  def init_ses( msg )
    # unique visit as the init_ses is called so let's add one for total and unique visit
counters
    @visits_unique += 1
    @visits_total += 1

    # this is really important development tool and will
    # send a message into web browser's console, check from your Firebug for example
    msg.console( "Welcome, user! Your Session ID is #{msg.ses_id}. This is your first visit."
)
  end

  # restore session will be invoked when an old session in client is detected
  def restore_ses( msg )
    #add one to total visits
    @visits_total += 1

    # send a happy message to the browser's console
    msg.console( "Welcome back, user! This is visit number #{@visits_total}!")
  end

  # runs every time during the synchronization (every time the server talks with the client)
  def idle( msg )
    # let's add one to the hit counter as a message to server is
    # delivered even though only idle method is called
    @hits_total += 1
    msg.console( "Hit count: #{@hits_total}" )
  end

  # invoked once per page load when the framework has been initialized
  def init_ui( msg )

  end
end
# registers Counter as 'counter'
Counter.new.register( 'counter' )
```

We added two more variables recording total hits and unique visits. Unique visits are easily distinguished as they can be added only when `init_ses( msg )` is loaded. This means that the client initializes everything from the beginning and starts a new session. This can be estimated to a new client on this example. The Riassence Framework uses cookies to detect the old session. Total hits is attached to the `idle(msg)` . This means that the increment is done whenever client polls the server (with specified intervals) and idle method is run. The package is sent between server and the client whenever the client contacts the server.

The Message object instance msg is supplied as an parameter and contains all required request/response and session related data as well as the primary server-side API for communicating with the client and other plugins. For example here session id can be extracted from the `msg` with `msg.ses_id`. More about `msg`s later.

STEP 5

Let's add a session variable, which records the page loads per session.

```
# basic Plugin class extension
class Counter < Plugin
  # primary initialization routine of the whole plugin, runs only once
  def init
    # total hits are recorded with plugin wide scope
    @hits_total = 0

    # so are visits, both unique and all visits
    @visits_total = 0
    @visits_unique = 0
  end

  # session initialization, whenever a new client is detected this method will be invoked
  def init_ses( msg )
    # unique visit as the init_ses is called so let's add one for total and unique visit
counters
    @visits_unique += 1
```

```
    @visits_total += 1

    # initialization of the session
    # let's add counter for that we can keep track of how many visits
    # this particular session (== client) has had to the site
    msg.session['counter'] = {
      # defined as numeral in a hash for the key (symbol) :visits
      :visits => 1
    }

    # this is really important development tool and will
    # send a message into web browser's console, check from your Firebug for example
    msg.console( "Welcome, user! Your Session ID is #{msg.ses_id}. This is your first visit."
)
  end

  # restore session will be invoked when an old session in client is detected
  def restore_ses( msg )
    #add one to total visits
    @visits_total += 1

    # add one as well to client's session specific counter (each client has its own counter)
    msg.session['counter'][:visits] += 1

    # send a happy message to the browser's console
    msg.console( "Welcome back, user! This is your #{msg.session['counter'][:visits]}.
visit!" )
  end

  # runs every time during the synchronization (every time the server talks with the client)
  def idle( msg )
    # let's add one to the hit counter as a message to server is
    # delivered even though only idle method is called
    @hits_total += 1
    msg.console( "Hit count: #{@hits_total}" )
  end

  # invoked once per page load when the framework has been initialized
  def init_ui( msg )

  end
end
# registers Counter as 'counter'
Counter.new.register( 'counter' )
```

The session variables are stored in a hash. They can be easily accessed with symbols, such as :visits.

STEP 6

The last step is to add UI. The UI is created with javascript file stored in the 'js' directory on the plugin bundle. Ruby code first:

```
# basic Plugin class extension
class Counter < Plugin
  # primary initialization routine of the whole plugin, runs only once
  def init
    # total hits are recorded with plugin wide scope
    @hits_total = 0

    # so are visits, both unique and all visits
    @visits_total = 0
    @visits_unique = 0
  end

  # session initialization, whenever a new client is detected this method will be invoked
  def init_ses( msg )
    # unique visit as the init_ses is called so let's add one for total and unique visit
counters
    @visits_unique += 1
    @visits_total += 1

    # initialization of the session
    # let's add counter for that we can keep track of how many visits
    # this particular session (== client) has had to the site
    msg.session['counter'] = {
      # defined as numeral in a hash for the key (symbol) :visits
```

```
      :visits => 1
    }

    # this is really important development tool and will
    # send a message into web browser's console, check from your Firebug for example
    msg.console( "Welcome, user! Your Session ID is #{msg.ses_id}. This is your first visit."
)
  end

  # restore session will be invoked when an old session in client is detected
  def restore_ses( msg )
    #add one to total visits
    @visits_total += 1

    # add one as well to client's session specific counter (each client has its own counter)
    msg.session['counter'][:visits] += 1

    # send a happy message to the browser's console
    msg.console( "Welcome back, user! This is your #{msg.session['counter'][:visits]}.
visit!" )
  end

  # runs every time during the synchronization (every time the server talks with the client)
  def idle( msg )
    # let's add one to the hit counter as a message to server is
    # delivered even though only idle method is called
    @hits_total += 1
    msg.console( "Hit count: #{@hits_total}" )
  end

  # invoked once per page load when the framework has been initialized
  def init_ui( msg )
    # grab the session with 'counter' (registered as 'counter')
    ses = msg.session['counter']
    # send the basic libraries, controls (HControl, HView etc. and the default theme for
controls)
    include_js( msg, ['controls','default_theme'] )
    # include js code from the plugin 'js/', in this case 'js/counter.js'
    msg.reply require_js( 'counter' )
    msg.reply("counter1 = Counter.nu(#{extract_hvalues_from_hash(ses)});")
    msg.reply("counter1.setTotalVisitors(\"#{@visits_total}\");")
    msg.reply("counter1.setTotalHits(\"#{@hits_total}\");")
    msg.reply("counter1.setSessionVisits(\"#{msg.session['counter'][:visits]}\")")
    msg.reply("counter1.setUniqueVisits(\"#{@visits_unique}\");")
  end
end
# registers Counter as 'counter'
Counter.new.register( 'counter' )
```

Javascript:

```
Counter = HApplication.extend({
  constructor: function(_values) {
    this.base();
    this.values = _values;
    this.drawSubviews();
  },

  drawSubviews: function() {
    counterWin = HWindow.nu([50,50,550,350],this,{
      label: "Counter Window"
    });
    this.total = HStringView.nu([5,5,300,30], counterWin, {
      value: "Total visitors: not defined"
    });
    this.hits = HStringView.nu([5,40,300,30], counterWin, {
      value: "Total hits: not defined"
    });
    this.visits = HStringView.nu([5,75,300,30], counterWin, {
      value: "Session visits: not defined"
    });
    this.unique = HStringView.nu([5,110,300,30], counterWin, {
      value: "Unique visits: not defined"
    });
  },

  setTotalVisitors: function(value) {
    this.total.setValue("Total visitors " + value);
  },
  setTotalHits: function(value) {
    this.hits.setValue("Total hits " + value);
```

```
  },
  setSessionVisits: function(value) {
    this.visits.setValue("Session visits " + value);
  },
  setUniqueVisits: function(value) {
    this.unique.setValue("Unique visits: " + value);
  }

});
```

PLUGINS

10. Server Plugin Model

11. Programming Plugins

10. SERVER PLUGIN MODEL

11. PROGRAMMING PLUGINS

CLIENT
12. Client Framework
13. Client Themes

12. CLIENT FRAMEWORK

13. CLIENT THEMES

DATA MODELS
14. DATA MODELS

DATA MODEL

SERVLETS
15. Programming Servlets

15. PROGRAMMING SERVLETS

SOAP
16. Programming SOAP Services

16. PROGRAMMING SOAP SERVICES

USER INTERFACE
17. Using the GUI Description Format
18. User Interface Model
19. Advanced User Interface Programming

17. USING THE GUI DESCRIPTION FORMAT

18. USER INTERFACE MODEL

19. ADVANCED USER INTERFACE PROGRAMMING

INTEGRATION

20. Integration Examples

20. INTEGRATION EXAMPLES

CREDITS
21. Lisenssi

21. LISENSSI

Kaikki kappaleet ovat kirjoittajien tekijänoikeuden alaisia. Jos muuten ei sanota, kaikki luvut tässä käyttöoppaassa on lisensoitu **GNU General Public License version 2** mukaisesti.

Tämä dokumentaatio on vapaata dokumentaatiota: voit jakaa sitä eteenpäin ja/tai muokata sitä Free Software Foundationin GNU General Public License mukaisesti; joko lisenssin version 2, tai (tahtoessasi) minkä tahansa myöhemmän version.

Dokumentaatiota jaellaan siinä toivossa, että se on käyttökelpoisa, mutta ILMAN MITÄÄN TAKUUTA; ilman edes MYYTÄVYYDEN tai TIETTYYN KÄYTTÖÖN SOPIVUUDEN oletettua takuuta. Katso lisätietoja GNU General Public Licensestä.

Tämän dokumentaation mukana olisi pitänyt tulla kopio GNU General Public Licensestä, mikäli sitä ei tullut kirjoita osoitteeseen Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

KIRJOITTAJAT

USER INTERFACE PROGRAMMING
© Tomi Toivio 2010

CLIENT FRAMEWORK
© Tomi Toivio 2010

CLIENT THEMES
© Tomi Toivio 2010

CONFIGURATION
© Tomi Toivio 2010

CREDITS
© adam hyde 2006, 2007
Modifications:
Tomi Toivio 2010

DATA MODELS
© Tomi Toivio 2010

INSTALLATION
© Tomi Toivio 2009, 2010

INTEGRATION EXAMPLES
© Tomi Toivio 2010

INTRODUCTION
© adam hyde 2006, 2007
Modifications:
Tomi Toivio 2009, 2010

JAVASCRIPT BASICS
© Tomi Toivio 2010

PROGRAMMING PLUGINS
© Tomi Toivio 2010

PROGRAMMING SERVLETS

PROGRAMMING SOAP

RUBY BASICS

SERVER PLUGIN MODEL

TROUBLESHOOTING

TUTORIAL

USER INTERFACE MODEL

USING GUI

VISITOR COUNTER

YAML BASICS

Free manuals for free software

GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

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

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

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

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

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

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

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

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

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

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

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

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

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

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NO WARRANTY

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

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

END OF TERMS AND CONDITIONS