Code Management

From Matchi Wiki

Contents

Code Management through Subversion

We currently use Subversion (a.k.a. SVN) to manage all source code.

Code Repositories and Code Trees

The trunk code repositories as of August 2016 are:

- `usr` - Utility scripts and batch jobs that deployed to the target system directories under `/usr/local/`.
- `php` - All PHP code
- `sql` - Database-related SQL scripts, which includes table, view, trigger and stored procedure groups
- `rabbit` - RabbitMQ messanger scripts

USR Code Tree

```
usr
└── local
    ├── bin
    ├── etc
    │   └── cron
    ├── lib
    ├── sbin
    └── share
        ├── icons
        └── templates
```

Deployment of the USR code tree

`[sandbox]/usr/locl/bin` contains both server-side processes and client-side developer utilities. Deploy the contents of this directory on both servers and development workstations to the local `/usr/locl/bin` directory. Do the same for the directories `[sandbox]/usr/locl/lib` and `[sandbox]/usr/locl/share` directories.

`[sandbox]/usr/local/sbin` contains daemon processes. These need to be deployed only on servers that will execute these daemons in the respective `/usr/local/sbin` directories.

`[sandbox]/usr/locl/etc/cron` contains cron job scripts. Some of the cron job scripts invoke processes in `/usr/local/bin` and `/usr/local/sbin` directories. Cron job scripts are also only deployed on servers that are assigned to execute them. A deployed cron jobs is assigned to executing to the certain execution pattern by adding a symbolic link to the cron script in any of the directories `/etc/cron.hourly`, `/etc/cron.daily`, `/etc/cron.weekly` or `/etc/cron.monthly`.

PHP Code Tree

```
php
├── com_innovation
│   ├── assets
│   │   ├── actionicons
│   │   │   ├── 220x220
│   │   │   ├── 32x32
│   │   │   ├── 48x48
│   │   │   └── 80x80
│   │   ├── fileicons
│   │   │   ├── 128x128
│   │   │   ├── 160x160
│   │   │   ├── 180x180
│   │   │   ├── 32x32
│   │   │   └── 80x80
│   │   ├── layout
│   │   │   └── css
│   │   │       └── images
│   │   ├── planicons
│   │   │   ├── 32x32
│   │   │   ├── 48x56
│   │   │   ├── 64x75
│   │   │   ├── 80x94
│   │   │   └── 90x105
│   │   ├── roleicons
│   │   │   ├── 32x32
│   │   │   ├── 64x75
│   │   │   ├── 80x94
│   │   │   └── 90x105
│   │   ├── stateicons
│   │   │   ├── 220x220
│   │   │   ├── 32x32
│   │   │   ├── 48x48
│   │   │   └── 80x80
│   │   └── util
│   ├── controllers
│   ├── helpers
│   │   ├── challenges
│   │   ├── emails
│   │   ├── messages
│   │   │   ├── innovationdetailedviews
│   │   │   ├── innovationlistviews
│   │   │   └── innovationreports
│   │   ├── ratings
│   │   │   └── innovationdetailedviews
│   │   ├── reports
│   │   └── videos
│   ├── language
│   │   └── en-GB
│   ├── models
│   │   ├── fields
│   │   └── forms
│   └── views
│       ├── innovationdetailedviews
│       │   └── tmpl
│       │           └── includes
│       ├── innovationforms
│       │   └── tmpl
│       │       └── includes
│       ├── innovationlistviews
│       │   └── tmpl
│       │       └── includes
│       └── innovationreports
│           └── tmpl
│               └── includes
```

Deployment of the PHP code tree

(TODO)

SQL Code Tree

```
sql
├── charts
├── data
├── design
├── etl
├── hacks
├── storedprocs
├── tables
├── triggers
└── views
```

Deployment of the SQL code tree

(TODO)

Rabbit Code Tree

(not is use yet)

Subversion Methods

More details about Subversion here: http://svnbook.red-bean.com

Creating a Sandbox

Create a new Sandbox directory when starting with a new sub-project, e.g. for the PHP code tree do this:

```
$ mkdir TEC-1107
$ cd TEC-1107
$ svn checkout http://mapp01/svn/matchi/trunk/php
```

Updating Code into your Sandbox

You can update an entire directory and its child directories by going to the relevant directory and doing an update from there:

```
$ svn update
```

Or your can update a specific file only into your sandbox:

```
$ svn update [file]
```

Adding a file to the code base

This step only marks the file(s) or directory for later checking-in, you still need to perform a check-in / commit before the file is properly lodged in Subversion.

```
$ svn add [file1] [file2] [file3] ...
```

You should also include in a comment section in each file a keyword string that automatically expands the check-in version details when the file is eventually committed:

- Add the following in a comment in the top of each file:

For PHP Code:

```
// $Id: $
```

For SQL scripts:

```
-- $Id: $
```

For BASH and Perl scripts:

```
# $Id: $
```

This will expand on check-in to something similar to this:

```
# $Id: scan_appcode.sh 3467 2016-03-05 13:47:16Z gerrit $
```

Also set the "Id" property so that Subversion knows that the keyword string "$Id: $" needs to be expanded:

```
$ svn propset svn:keywords "Id" [file]
```

Special case: For executable files like batch scripts and cron scripts, the executable flag needs to be set and needs to persist in Subversion, so for BASH and Perl command-line scripts, we also need to do this:

```
$ svn propset svn:executable on [file]
```

A utility exists that sets this correct properties on all files in the current directory: `/usr/local/bin/svnpropset.sh`. Simply run it before checking brand new files in:

```
$ svnpropset.sh
```

Checking in / Committing code changes

Code changes need to be associated with a Change Issue Id that is recorded in JIRA. See http://jira.matchi.biz. The Change Issue Id is in the form `TEC-[numnber]` and needs to be included in the comment when one or more files are checked into Subversion:

```
$ svn ci -m "TEC-[xxxx]  [optional further comment]" [file1] [file2] [file3] ...
```

It is necessary to specify the actual file names: If you don't all files that have been changed will automatically be selected and checked-in.

```
$ svn ci -m "TEC-[xxxx]  [optional further comment]"
```

Compare two versions of a file that are already checked-in in Subversion

```
$ svn diff -r [release_1_id]:[release_2_id] [filename]
```

Compare the current working file to the last version that was checked in

You might want to do this

```
$ svn diff [file]
```

If you want to highlight the differences in colour, get Subversion to use the diff-tool of your choice rather than the default one and then pipe it through a colourizer like colourdiff:

```
$ svn diff --diff-cmd  /usr/bin/diff -x --ignore-all-space  [file]  | colordiff
```

A utility program exists in `/usr/local/bin/svndiff.sh` that does the same thing.

Compare a file to the previous checked in version

```
$ svn diff -r PREV [file]
```

If you want to highlight the differences in colour, get Subversion to use the diff-tool of your choice rather than the default one and then pipe it through a colourizer like colourdiff:

```
$ svn diff --diff-cmd  /usr/bin/diff -x --ignore-all-space -r  PREV [file]  | colordiff
```

A utility program exists in `/usr/local/bin/svnprev.sh` that does the same thing.

Retrieved from "http://wiki.matchi.info/index.php?title=Code_Management&oldid=1475"

Category: Pages with syntax highlighting errors