Making Software

Manual de instalação

Dez/2013

Índice

Introdução

Documento de instalação do Manking Software.
Nas próximas páginas estão as instruções para a instalação do software.

Resumo

Esse sistema possui duas camadas, um website Python/Django e um motor em Perl.
Através do website o usuário escolhe o que prentende no seu sistema, e quando finalizar, o motor Perl gera um sistema autônomo, completamente separado do Making Software, que pode ser instalado em outro servidor, editado, etc.

Licença de uso

Qualidade do código

Essa é uma versão de desenvolvimento, uma prova de conceito de um sistema gerador de códigos de sistemas web. Não foi feito nenhum trabalho de otimização, de utilização de melhores práticas, nem de limpeza do código.
Esse documento está na sua primeira versão, e tem o mesmo padrão de qualidade.

17/Dez/2013

Dependências

Segue abaixo a lista de dependências para rodar o Making Software e uma breve descrição de como instalar e/ou configurar as dependências.
Em caso de dúvida na instalação dessas dependências por favor consulte as urls fornecidas abaixo ou o Google, com certeza ele sabe mais que eu. Isso é sério, quem trabalha ou trabalhou com configuração de software, montagem de ambientes etc, sabe que as grandes dificuldades aparecem nas dependências, instalação de pacotes de terceiros etc.

Linux

O Making Software só roda em Linux porque ele utiliza comandos nativos do sistema operacional. Talvez ele rode em Mac OSX e com certeza não roda em Windows.
Por enquanto o sistema foi testado com Ubuntu/Debian e com Slackware.

Django

Você precisa ter o Django instalado na sua distribuição Linux. Se não tem, siga os passos abaixo e instale:
https://docs.djangoproject.com/en/dev/topics/install

Ubuntu

- sudo apt-get install python-setuptools
- sudo easy_install pip
- sudo pip install Django

Slackware

- setuptools
  wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py -O - | python
- pip
  wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py -O - | python
- Django
  pip install Django

O Django é um Framework baseado em Python. O Python vem instalado por padrão nas distribuições Linux.

Apache

Esse manual não ensina a instalar e configurar o servidor web Apache para usar com o Making Software. É utilizado o servidor web de desenvolvimento do Django.
A montagem e a configuração do ambiente como um todo é custosa, e deixar o Apache de lado facilita esse trabalho. Os impactos que essa decisão traz são:

- Não utilização do WSGI.
- Não é recomendado usar esse servidor web em ambiente de produção

- O sistema não faz deploy automaticamente. Uma vez que o usuário finalizou a criação do sistema ele não poderá clicar no botão “Publicar” do website que chama o Perl na camada 2. Ele precisa executar o Perl na mão.

Sobre esse último item, eu não sei bem o motivo. Talvez o servidor web crie alguma espécie de sandbox, mas não fui à fundo para investigar. Eu tentei executar o mesmo com usuário privilegiado mas também não funcionou.

Banco de dados - Postgres

A camada Perl utiliza o banco de dados Postgres, instale-o, crie um usuário e uma base de dados para utilizar com o Django.
Quando você instalar a aplicação Django a estrutura do banco de dados será criada, então você não precisa se preocupar em manipular o banco, exceto instalá-lo e criar um usuário.

Ubuntu

- *sudo apt-get install postgresql*
- *sudo apt-get build-dep python-psycopg2*
- *sudo apt-get install python-psycopg2*

psycopg2

Conector do Postgres para o Python. Para instalar no Slackware, ao invés de usar os fontes e scripts do Slackware, eu instalei com o pip:

- *pip install psycopg2*

Após instalar o Postgres, você pode instalar uma GUI para gerenciar o banco. Não é necessário mas ajuda em alguns casos. Uma muito boa é o pgAdmin http://www.pgadmin.org

Perl

O perl já vem nativo com todas as distribuições Linux. O Making Software utiliza alguns pacotes Perl que não vem instalado por padrão, mas para instalá-los basta usar o CPAN. Se você nunca instalou pacotes do CPAN, leia abaixo:
http://www.cpan.org/modules/INSTALL.html
Se não quiser ler, quando aparecer alguma dependência em Perl você será alertado na tela, no comando de linha. Então via de regra, basta digitar:

- *sudo cpan <nome do pacote que falta>*

Ele instala automagicamente.
Eu tive problemas para instalar o DBD::Pg no Slackware. Tentei várias vezes, instalei o DBI, DBI::DBD, DBD e nada. No final forcei a instalação e foi:

- *sudo cpan -f DBD::Pg*

Download do Making Software

1. Entre no seu diretório home (ou em qualquer outro onde queira baixar e instalar o Making Software)
   cd ~
2. Você pode fazer o dowload do arquivo compactado, ou diretamente via Git.
   a. Download do pacote.
      Entre na sua pasta home, como explicado acima e digite o comando abaixo:
      wget https://github.com/fortinbras/making_software/archive/master.zip
   b. Clone do diretório do Git
      - Verifique se você tem o Git instalado na sua distribuição, senão instale.
      - git clone https://github.com/fortinbras/making_software
3. Se você escolheu a opção “a” acima, você precisa executar os dois passos abaixo, caso contrário o download está finalizado:
   a. Descompacte o arquivo.
      unzip master.zip
   b. Renomeie o diretório. O Github inclui o master na sua nomenclatura de versionamento. Se você fez o dowload direto sem usar o Git, é necessário voltar para a nomenclatura original do sistema
      mv making_software-master making_software

Estrutura de diretórios

/home/seu_nome/making_sofware/ - Pasta base do Making Software
/home/seu_nome/making_sofware/loopware - Pasta da aplicação Python/Django
/home/seu_nome/making_sofware/making_sofware_perl - Pasta da aplicação Perl
/home/seu_nome/making_sofware/static_files - Arquivos estáticos
/home/making_sofware/projects - Onde serão criados os sistemas.

A última pasta não é criada automaticamente. Você deverá criá-la quando solicitado ao longo desse documento.

Configuração da camada 1 - Python Django

1. Configuração do settings.py

Esse arquivo deve ficar localizado em:
<home da aplicacao>/making_software/loopware/loopware/settings.py.

Utilize o template **settings.template.py** localizado nesse diretório e altere o parâmetro *INTELIFORM_PERL_DIR*:
Por exemplo, se você fez o download do making_software no seu diretório home, esse parâmetro deve ter a seguinte configuração:

- *INTELIFORM_PERL_DIR = '/home/your_home'*
- Renomeie o arquivo **settings.template.py** para **settings.py**.

2. Configuração do banco de dados

Crie o banco de dados

Digite os comandos abaixo para criar o banco de dados

1. Passe para o usuário Postgres:
    a. sudo -s
    b. su postgres
2. createdb -T template0 making_software

Se você criar o banco de dados com outro nome que não *“making_software”* , altere o settings.py e o arquivo config.pl da aplicação Perl.
Lembre-se que o Postgres precisa estar rodando.

settings.py

DATABASES

Para alterar configurações como nome do banco, usuário e senha do Postgres, procure o parâmetro *DATABASES* no arquivo settings.py.

Postgres Socket

Se você instalar o Postgres através do código fonte e compilar na sua máquina como no caso do Slackware, o socket do Postgres vai estar em /tmp/.s.PGSQL.5432.
Se você baixar compilado, como no caso do Ubuntu ou Fedora, ele vai estar em /var/run/postgresql/.s.PGSQL.5432.
Esse endereço precisa ser corretamente configurado, senão você vai ter problemas.
Por padrão o endereço configurado é
'HOST': '/var/run/postgresql/'

Problemas de autenticação

Se tiver problemas de autenticação no banco de dados, como :
"ident authentication failed for user postgres",
altere o método de autenticação do arquivo *pg_hba.conf* para “trust”.

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
local           all         postgres                              trust
```

Você precisa estar logado com o usuário *“postgres”*.
Mas antes leia esse documento para entender os métodos de autenticação no Postgres:
http://www.postgresql.org/docs/8.4/static/auth-methods.html

3. Camada 1 - primeiro teste

Com o parâmetro *INTELIFORM_PERL_DIR* configurado corretamente e o arquivo renomeado para settings.py, a aplicação web já pode ser testada.

1. No prompt do comando de linha digite o seguinte comando:
    a. *cd /home/your_home/making_software/loopware*
    b. *./manage.py runserver*
    O Django possui um servidor web compacto para efeito de testes. Se você não digitar a porta como é o caso acima, ele sobe o serviço na porta 8000.
2. Abra um navegador e digite a seguinte url
    a. http://localhost:8000

Se tudo foi instalado corretamente a página inicial da aplicação web Making Software deve ser apresentada.
Ainda não é possível fazer login, criar sistemas etc, porque a base de dados não foi criado.

4. Instalação do sistema Django

Para instalar o sistema Django, que é a camada 1 do Making Software digite os comandos abaixo. Isso vai criar a base de dados do Making Software:

1. cd *<home da aplicacao>/making_software/loopware/*
2. *./manage.py syncdb*

5. Camada 1 - segundo teste

Se tudo foi configurado corretamente, você pode acessar novamente o endereço:
http://localhost:8000 e utilizar a camada 1 do Making Software.
Crie, edite e exclua sistemas e elementos para ver se tudo está funcionando corretamente.

Configuração da camada 2 - Perl

A aplicação Perl é completamente independete da aplicação Django.
Ela é chamada pela aplicação Django quando o usuário quer publicar um sistema criado no Making Software.
Na instalação desse manual não será possível disparar o Perl através do Django. O Perl deverá ser executado na mão conforme será explicado nessa sessão.

1. Biblioteca do sistema

O sistema Perl possui apenas uma biblioteca que precisa ser instalada.

1. cd *<home da aplicacao>/making_software/making_software_perl/Classes/Sistema*
2. *./instala*

Os comandos acima instalam a biblioteca Sistema.pm, no diretório de libs Perl do seu SO.

2. Camada 2 - primeiro teste

Digite os comandos abaixo para fazer o primeiro teste de compilação do sistema. Isso devia ser tudo automático, mas não é. Então mão na massa:

1. cd *<home da aplicacao>/making_software/making_software_perl/*
2. *perl create_project.pl*

Esse primeiro teste muito provavelmente vai retornar erros de compilação, porque são necessárias algumas biblitecas Perl que você pode não ter instaladas.
Instale essas bibliotecas conforme abaixo:

3. Bibliotecas Perl/CPAN

É necessário uma série de libs Perl para o funcionamento do sistema. Todas elas podem ser baixadas diretamente do CPAN.
Se você não conhece o CPAN, ele é o repositório oficial de bibliotecas mantidas pela comunidade. www.cpan.org.
Por exemplo, se ao executar o comando do item anterior você receber a mensagem “Can´t locate DBI.pm” instale a biblioteca DBI.pm, que é uma biblioteca muito comum utilizada para a camada de acesso à banco de dados.

1. sudo cpan DBI

Quando você vai instalar alguma biblioteca, o CPAN tenta resolver as dependências da biblioteca que você está instalando. Ao longo da instalação você pode ter que responder à algumas perguntas do CPAN sobre preferências da instalação, mas o processo é praticamente todo automático.
Para cada biblioteca que ele não encontra ele reclama e você precisa instalar.

Um outro exemplo, “Can´t locate Template.pm”. Instale a biblioteca Template.pm, que é uma biblioteca utilizada para se utilizar templates na geração de documentos em Perl.

2. sudo cpan Template

Repita esse passos até resolver todas as dependências de bibliotecas Perl.
Eu tive problemas para instalar o DBD::Pg no Slackware. Tentei várias vezes, instalei o DBI, DBI::DBD, DBD e nada. No final forcei a instalação e foi:

- *sudo cpan -f DBD::Pg*

4. Configuração do hosts.template.pl

Esse arquivo deve ser alterado somente se você vai usar mais de um ambiente, por exemplor produção e desenvolvimento.
Caso contrário renomei-o para hosts.pl e deixe o conteúdo do arquivo como está.

1. cd *<home da aplicacao>/making_software/making_software_perl/*
2. *mv hosts.template.pl hosts.pl*

5. Configuração do config.template.pl - 1.a parte

As configurações abaixo, são as configurações mínimas necessárias para se fazer o segundo teste da camada 2.
Se você preferir, pode configurar tudo de uma vez.
Altere os parâmetros abaixo:

- **temp_dir**
  Diretório onde o Making Software vai criar os arquivos temporários.
- **django_admin**
  Dependendo da versão do Django o executável tem o nome de django-admin.py ou simplesmente django-admin. Ele pode estar em /usr/local/bin, ou /usr/bin.
  Preencha corretamente.
- **Postgres**
  Ver o item Postgres nas intruções sobre a camada 1 e preencha com as mesmas informações.
- **echo**
  Esse parâmetro é para responder a pergunta sobre criação de um usuário quando vai ser feito o deploy da aplicação Django criada e depende da distribuiçõa Linux a maneira como deve ser tratada a resposta dada sem interação do usuário.

Salve/renomeie o arquivo como config.pl

1. cd *<home da aplicacao>/making_software/making_software_perl/*
2. *mv config.template.pl config.pl*

6. Camada 2 - segundo teste

Entre na pasta da apilcação Perl para testar a instalação do sistema Perl.

1. cd *<home da aplicacao>/making_software/making_software_perl/*
2. *perl create_project.pl <id de um sistama criado> <home da aplicacao>*

Por exemplo se você instalou o sistema no seu diretório home, para testar com o primeiro sistema criado digite o comando abaixo:

- *perl create_project.pl 1 </home/seu home>*

Por padrão, esse ambiente de desenvolvimento está fazendo o deploy dos sistemas criados na porta **8001**. Para verificar se o sistema subiu corretamente, abra um browser e digite:

- http://localhost:8001

O mesmo usuário que você criou no Making Software é automaticamente copiado para cada novo sistema criada, dessa maneira você pode utilizar o mesmo usuário e testar o sistema que você acabou de subir.

FIM

Se você chegou até aqui e conseguiu criar um sistema no Making Software você conseguiu alcança o objetivo desse tutorial. Parabéns!!

O texto que segue abaixo ainda não está finalizado, e documenta pendências, e outras configurações não obrigatórias.

Configuração do Apache

O servidor web Apache cumpre as seguintes funções

- Servir a aplicação web da camada 1 (Python/Django) de maneira definitiva, e não através do servidor web de desenvolvimento do Django.
- Integrar a camada 1 (Python/Django) com a camada (Perl), fazendo com que o Perl seja chamado diretamente da aplicação web Django.
- Servir os sistemas criados no Making Software.

As configuração do Apache ainda serão documentadas.

É necessário criar a pasta abaixo ou dar permissão de escrita para o usuário que executa o Apache.
Isso fica para documentar depois também.

1. Pasta para os projetos criados

Essa pasta deve ser criada para acomodar os projetos que serão criados pelo Making Software. Se preferir crie no /home ou onde achar melhor. O endereço dessa pasta é o valor da variável MEDIA_ROOT. Digite os comandos abaixo para criar:

- *sudo mkdir -p /home/making_software/projetos*

Outras configurações

Essas configurações não são obrigatórias para o funcionamento do sistema.

Arquivos estáticos

Os arquivos estáticos, css, js, imagens, etc estão na pasta /static-files e serão apresentados corretamente quando utilizado o servidor web do Django.
Se você alterar o ambiente para outro servidor web, mova essa pasta e faça o direcionamento correto conforme as instruções abaixo.
https://docs.djangoproject.com/en/1.2/howto/static-files/

Verificação de e-mail

Quando um usuário cria uma conta em um sistema seu, é bom que se verifique a veracidade do e-mail utilzado através de uma checagem de e-mail. O Making Software utiliza o allauth, pacote que cuida da criação e gerenciamento de acesso das contas. Por padrão a verificação de e-mail está desabilitada porque você precisa ter um servidor SMTP instalado na sua máquina, caso contrário vai dar pau na aplicação.
Se você tem um servidor de SMTP ou sabe como habilitar um servidor dummy, descomente a linha:
#ACCOUNT_EMAIL_VERIFICATION = ("mandatory")

Veja abaixo como habilitar um servidor SMTP dummy, só para testes.
https://docs.djangoproject.com/en/dev/topics/email/#testing-email-sending

Pendências

- Resolver dependência de nomenclatura entre as versões do Django.
Por exemplo:
TEMPLATE_DIRS setting must be a tuple. Please fix your settings, as auto-correction is now deprecated.

- templates/views.py - guardian (todas as referencias)
- templates/settings.py - guardian (todas as referencias)
- Sistema.pm - retorno vazio pra Relacionamentos (return;)
- making_software_perl/templates/fb_app_account_base.html

```
{% comment%}
  <li><b><a href="/[% base.slug %]/fb_app">Voltar</a></b></li>
{% endcomment%}
  <li><b><a href="/fb_app">Voltar</a></b></li>
```

- making_software_perl/default_apps/allauth/account/views.py

```
# isso precisa comentar
      success_url = '/flat/bem-vindo/'

   # Comuta abaixo para valer para o codigo aberto ou codigo fechado
   or self.success_url)
   #or my_success_url)
```

- create_project.pl

```
&Runserver($project_name);
print ">>> Webserver is set, up and running<br>\n";

#&ConfigApache($project_name);
#print ">>> Webserver is set, up and running<br>\n";
```

colocar tratamento de erro para todas chamadas de comandos nativas do SO, pois cada distribuição varia, e mesmo no django os executáveis variam de nome de versão para versão.
Por exemplo:

- django_admin => '/usr/bin/django-admin',

Otimização

O sistema da camada 2 em Perl pode ser otimizado de várias maneiras.
Uma das tarefas do script create_project.pl é receber parâmetros e gerar arquivos fisicos no filesystem através da utilização de templates (Template Toolkit). Essa função e utilizada várias vezes, mas o código está repetido. Uma clase poderia executar essa função e reduzir o código a menos de um terço, certamente.
Outras funções que utilizam acesso à banco de dados também poderiam ser agrupadas em uma classe.