Criando e Adaptando Gemas do Ruby On Rails

Gustavo Toshi Komura

Universidade Federal do Paraná - UFPR
Ciência da Computação
Departamento de Informática
Orientador: Professor Doutor Bruno Müller Junior

Curitiba - 7 de Novembro de 2014

Sumário

Definição

Definição de Biblioteca

Conjunto de subprogramas ou rotinas que tem por função principal prover funcionalidades que geralmente são utilizadas por desenvolvedores em um determinado contexto.

Vantagens do Uso de Bibliotecas

- O uso de bibliotecas permite a reutilzação de código.
- O uso de bibliotecas permite a modularização do projeto.

História

História

- **COMPOOL** *(Communication Pool)*: *Software* desenvolvido em **JOVIAL** que tinha o objetivo de compartilhar informação entre vários programas.
- **COBOL**: Linguagen de programção que em 1959 possuía um sistema primitivo de bibliotecas.
- **FORTRAN**: Linguagen de programção que possuía um sistema que permitia a compilação de subprogramas independentemente dos outros.
- **Simula 67**: Linguagen de programção orientada a abjetos que em 1965 possibilatava o armazenamento de classes em arquivos de bibliotecas. Depois esses arquivos de bibliotecas poderiam ser incorporados em outros programas.

Classificação - Formas de Ligamento

Definição

Processo que implica em associar uma biblioteca a um programa ou outra biblioteca.

Tipos

- **Tradicional**: Os dados da biblioteca são copiados para dentro do executável do outro programa ou biblioteca.
- **Dinâmica**: A biblioteca é referênciada no outro programa ou biblioteca. Neste caso não existe tanto trabalho no momento de compilação, pois a biblioteca só será copiada quando o programa for carregado ou outra biblioteca for carregada na memória.
- **Remoto**: A biblioteca é carregada por chamadas de procedimentos remotos. A vantagem é que o programa e a biblioteca não precisam estar necessáriamente na mesma máquina, pois quando o programa requisitar a biblioteca, ela será transferida pela rede de uma máquina para outra.

Classificação - Momentos de Ligação

Definição

É o momento em que a biblioteca é carregada na memória.

Tipos

- **Carregamento em Tempo de Carregamento**: A biblioteca é carregada na memória no mesmo momento em que a aplicação esta sendo carregada.
- **Carregamento Dinâmica ou Atrasado**: A biblioteca é carregada na memória somente quando a aplicação faz a requisição da biblioteca.

Classificação - Formas de Compartilhamento

Definição

Disponiblidade da biblioteca para vários programas ao memso tempo.

Tipos

- **Compartilhamento em Disco**: Programas podem utilizar o mesmo arquivo em disco para acessar a biblioteca.
  - **Vantagens**: Economia de tempo de compilação, espaço ocupado por binários e a possibilidade de usufruir de atualizações em a necessidade de recompilar o programa.
  - **Desvantagens**: Caso um biblioteca seja excluida ou danificada, todos os programas que a utilizam ela serão afetados.
- **Compartilhamento em Memória**: Programas podem compartilhar o acesso ao mesmo código da biblioteca na memória.
  - **Vantagens**: Economia de memória e tempo de carregamento.
  - **Desvantagens**: Desempenho pode ser prejudicado por causa da obrigatoriedade do ambiente multi-tarefa e a falta de escalabilidade.

Ruby - Definição

Ruby

- Criada por **Yukihiro "Matz" Matsumoto**.
- **"Matz"** diz que "**Ruby** *é natural, não simples*" e também fala que "**Ruby** *é uma linguagem simples na aparência, mas é muito complexo internamente, assim como o corpo humano*".
- É uma linguagem dinâmica de código aberto com foco na simplicidade e produtividade.
- Possui sintaxe elegante natural de ler e escrever.
- Baseada nas melhores características de **Perl**, **SmallTalk**, **Python**, **Eiffel**, **Ada** e **Lisp**.
- Linguagem de script mais poderosa que **Perl** e mais orientada a objeto que **Python**.
- Tudo é objeto.

Ruby - História

Ruby

- Criada por **Yukihiro "Matz" Matsumoto** em 24 de fevereiro de 1993.
- O nome foi escolhido entre uma conversa entre **Yukihiro "Matz" Matsumoto** e **Keiju Ishitsuka**.
- **Ruby** poderia ter o nome **Coral**.
- **Ruby** é uma *birthstone* que coincidentemente representava o mês de aniversário de um de seus colegas.
- Sua primeira versão foi lançada oficialmente em 21 de dezembro de 1995 na "*Japanese Domestic NewsGroup*". E nos 2 dias subsequentes foram lançadas mais 3 versões junto com sua lista "*Japanese-Language ruby-list main-list*" ("*RubyTalk*").
- Em 2006 atingiu o seu auge com conferências pelas principais cidades do mundo e com mais de 200 mensagens por dia na sua principal lista.

Bibliotecas do Ruby (gemas)

gem

- A maior parte das gemas são distribuídas na forma de **gems**. Sua instalação é feita por meio da ferramenta **gem**.

Pacotes ".zip" e ".tar.gz"

- A menor parte das gemas são distribuídas na forma de arquivo ".zip" ou ".tar.gz". Motivo pelo qual a sua instalação é feita por meio da leitura dos arquivos **README** ou **INSTALL** contidos dentro da gema.

Ferramenta gem e o site RubyGems

O programa gem e o site RubyGems

- Criado em abril de 2009 por **Nick Quaranto**.
- É um sistema de pacotes do Ruby desenvolvido para facilitar a criação, o compartilhamento, e a instalação de bibliotecas.
- Possui características sinilares a ferramenta **apt-get**, mas ao invés de fazer a distribuídas de pacotes para **Debian GNU/Linux Distribution** e seus variantes, faz a distribuição de pacotes para o **Ruby**.
- Permite fazer buscas e instalações de gemas.
- A partir da versão "1.9" do **Ruby** não existe a necessidade de fazer a instalção do **gem**, pois ele vem por *default* nos pacotes.
- O site possuía o nome **Gemcutter** até a versão "*1.3.6*", sendo trocado para **RubyGems** com o objetivo de solidificar o papel do site na comunidade do **Ruby**.

Segurança do Ruby

Importância

- Perda de dados sigilosos.
- Perda de dinheiro.
- Indisponibilidade de serviços.

Esquema de Correção

- Reportar vulnerabilidade via e-mail para `security@ruby-lang.org`. Lista privada com membros que administram o **Ruby**.
- As vulnerabilidades recebidas por medidas de segurança só são divulgadas para a comunidade quando são solucionadas.
- Nas divulgações de vulnerabilidade são informados os tipos de erros, os problemas que eles podem causar, e as soluções que devem ser tomadas.

Segurança das Bibliotecas - Uso de gemas

Risco

- Toda gema utilizada na aplicação é instalada localmente no servidor.
- Caso o autor da gema seja mal intencionado, ele pode conseguir roubar dados do servidor.

Solução

- A partir da versão "*0.8.11*" o *RubyGems* disponilizou uma solução baseada no uso chaves de assinaturas criptografadas.
- Com o comando "**gem cert**" é possível criar uma par de chaves e empacotar os dados da assinatura dentro da gema.
- Com o comando "**gem install**" é possível fazer a verificação da chave de assinatura antes da sua instalação.
- Apesar do método ser benéfico, ele não é muito utilizado, pois são necessários muitos passos manuais e também não existe uma politica de segurança bem definida para essas chaves de assinautra.

Segurança das Bibliotecas - Reportar Vulnerabilidades

Esquema de Correção

- **Vulnerabilidades em gemas de outros usuários**:
  - Verificar se a vulnerabilidade ainda não é conhecida.
  - Enviar e-mail privado para o dono da gema.
  - Informar o problema e uma possível solução.
- **Vulnerabilidades nas própias gemas**:
  - Criar identificador **CVE** único, enviando e-mail para `cve-assign@mitre.org`.
  - Trabalhar em uma solução para o problema.
  - Criar um patch de correção quando o problema for corrigido.
  - Informar a comunidade sobre o problema e que essa vulnerabilidade foi corrigida no *patch* "*X*".
  - Registrar o problema em um *database open source* de vulnerabilidade (**OSVBD**) e enviar e-mail para `ruby-talk@ruby-lang.org` com *subject* "*[ANN][Security]*" informando detalhes sobre as vulnerabilidades, versões com o erro, e ações a seren tomadas.

Ferramentas Utilizadas

Ferramentas

- **VMware® Player**: aplicativo de virtualização que permite a utilização de vários SOs ao mesmo tempo sem a necessidade de reiniciar a máquina fisica.
- **RVM**: ferramenta criada em outubro de 2007 por *Wayne E. Seguin* que permite instalar, gerenciar, e trabalhar com multiplos ambiente do *Ruby* com um certo conjunto de gemas.
- **Ruby On Rails**: *framework* criado em 2003 por *David Heinemeier Hansson* e recebendo suporte do *Rails Core Team*, facilita o desenvolvimento de código, visando a produtividade sustentável.
- **Git**: ferramenta criada em 2005 por *Linus Torvalds*, permite fazer o gerenciamento de versões tanto de projetos grandes como pequenos com rapidez e eficiência.

Preparação do ambiente

Sequência de ferramentas a serem instaladas

- Instalação do **VMware® Player**.
- Instalação do **Ubuntu 12.04** no **VMware® Playe**.
- Instalação do **RVM** no terminal dentro do **Ubuntu 12.04**.
- Instalação do **Ruby** também no terminal.
- Instalação das gema básicas **rails** e **bundle** também no terminal.

Estrutura Básica

Estrutura de uma gema

- **gemspec**: arquivo que possui informações básicas da gema como nome, descrição, autor, endereço, e dependências.
- **bin**: diretório que possui os executáveis que serão carregados quando a gema for instalada.
- **lib**: diretório que possui todos os códigos **Ruby** referentes ao funcionamento da gema.
- **spec** ou **test**: diretório que possui todos os códigos de teste da gema.
- **Rakefile**: arquivo que possui código **Ruby** que faz a otimização de algumas funcionalidades por meio da ferramenta **rake**.
- **README**: arquivo que possui a documentação da gema (**RDoc** ou **YARD**).

Modelo de Criação - Criando a Estrutura

Formas de criação

- **Manual**: Implica na criação de todos os arquivo e diretórios manuallmente.
- **Automática**: Implica na execução do comando " **bundle gem "nome da gema"** ".

Modelo de Criação - Gemspec

Conteúdo do arquivo gemspec

- Nome da gema.
- Versão da gema.
- Autores da gema.
- E-mail dos autores e/ou e-mail para notificações sobre a gema.
- Breve descrição da gema.
- Descrição completa da gema.
- Localização dos arquivos e diretórios da estrtura básica da gema.
- Dependências da gema.

Modelo de Criação - Funcionalidades ou Testes

Escolha de desenvolvimento

- Depende da politica de desenvolvimento adotada no inicio do projeto.
- Desenvolver código de funcionalidade e depois desenvolver o código de teste.
- Desenvolver código de teste e depois desenvolver o código de funcionalidade.

Modelo de Criação - Versão da gema

Formas de Versionamento

- **PATH**: "*0.0.X*" para pequenas correções como correções de bugs.
- **MINOR**: "*0.X.0*" para médias alterações como alterações de funcionalidades.
- **MAJOR**: "*X.0.0*" para grandes modificações como remoção de funcionalidades.
- **PRE**: "*0.0.0-pre*" para pré-lançamentos de versões.

Modelo de Criação - Diferença entre module e class

Module

- Conjunto de métodos e constantes.
- Os métodos são estáticos.
- Sinilaridade com o conceito de **interface** do **JAVA**.

Class

- Conjunto de métodos e constantes.
- Para utlizar os métodos e constantes é necessário instanciar um objeto na memória.
- **class** é uma subclasse de **module** (“**initialize()**”, “**superclass()**”, “**allocate()**” e “**to_yank()**”).

Modelo de Criação - Código de Funcionalidade

Desenvolvimento de funcionalidades

- As funcionalidades da gema desenvolvidas em código **Ruby** devem ser inseridas no diretório **lib**.
- As funcionalidades da gema desenvolvidas em código **Javascript** devem ser inseridas no diretório **vendor**.
- Arquivo (**lib/"nome da gema".rb**).
- Modularizar a gema fazendo uso de **require**.
- Diretório (**lib/"nome da gema"/**).

Modelo de Criação - Código de Teste

Desenvolvimento de testes

- Os testes da gema desenvolvidas devem ser inseridos no diretório **test** ou **spec**.
- É recomendado nomear o arquivo de teste com " **teste_"funcionalidade a ser testada"** ".
- Uso de **asserts**,
- Utilizar o arquivo "**Rakefile**" para automatizar os testes.

Modelo de Criação - Execução de Teste

Realizar Testes

- Fazer o "**build** e a instalação da gema.
- Utilização da ferramenta **rake**.
- Realizar testes por arquivo de teste com o comando "**rake test/nome do arquivo**" ou "**rake spec/nome do arquivo**".
- Realizar teste utilizando todos os arquivos de teste executando o comando "**rake**".

Adaptando uma gema - Observações

Observações

- Adição de funcionalidades que geralmente são utilizadas.
- Funcionalidade a ser adicionada deve estar no mesmo contexto da gema.
  - **CORRETO**: Adição do cálculo da raiz quadrada de um número em uma gema que faz cáculos.
  - **ERRADO**: Adição da criação de mapas em uma gema que faz cálculos.
- Evita implementações do zero.

Adaptando uma gema - Engenharia Reversa

Engenharia Reversa

- É um processo de análise para a extração de informações de algo que já existe em um modelo de abstração de alto nível.
- As informações podem estar no formato de código fonte ou mesmo em um executável.
- O processo de análise para a extração de dados deve ser feita de forma minunciosa, pois pode ocorrer uma grande perda de recursos, caso alguma funcionalidade seja entendida de forma incorreta.
- O modelo de abstração de alto nível pode ser por exemplo um diagrama de caso de uso ou um diagrama de sequência.

Adaptando uma gema - Google Maps e sua API

Google Maps e sua API

- Criada por dois irmãos **Lars** e **Jens Rasmussen**.
- A rederização de mapas até 2005 possuía um alto custo, nessitando de servidores altamente equipados.
- O **Google Maps** foi anunciado em Fevereiro de 2005 no blog do *Google*.
- Em Junho de 2005 o *Google* anunciou a primeira **API** do **Google Maps**, pois percebeu que os usuários gostariam de incorporar os mapas em seus sites.
- Para fazer modificações na gema de exemplo foi necessário consultar o livro **Beginning Google Maps API 3** e o **Google Maps API V3**.

Adaptando uma gema - Entendimento e Adaptação

Entendimento da gema

- Verificar se realmente os diagramas estão de acordo com as características da gema.
- Encontrar os principais elementos da gema.
- Encontrar os elementos mais utilizados ou mais reaproveitadas da gema.

Adaptação da gema

- Encontrar as funcionalidades que podem ser reaproveitadas na inclusão das novas funcionalidades.
- Após implementação das novas funcionalidades, verificar o impacto que elas causaram na gema.

Trabalhos Futuros

Trabalhos Futuros

- Para a primeira gema de exemplo **gemtranslatetoenglish**:
  - Abordar mais recursos da linguagem **Ruby**.
  - Desenvolver mais funcionalidades para mostrar mais modelos de teste.
- Para a segunda gema de exemplo **Google-Maps-For-Rails adaptada**:
  - Incluir a escolha de locomoção.
  - Incluir a possibilidade de inserir pontos intermediários entre a origem e o destino.

Conclusão

Conclusão

- Proporcionou o entendimento sobre bibliotecas.
- Mostrou conceitos básicos da linguagem **Ruby**.
- Apresentou como criar uma biblioteca do **Ruby**.
- Mostrou como adaptar uma biblioteca do **Ruby**.
- Apresentou exemplos para facilitar o entendimento da explicação de como criar ou adaptar uma biblioteca do **Ruby**.
- Proporcionou a possibilidade de economizar tempo no desenvolvimento.