GRADEp: Guia de Instalação

GT GRADEp (2004-2005)

GRADEp: Guia de Instalação

por GT GRADEp (2004-2005)

Este documento descreve os procedimentos associados à instalação do middleware GRADEp, considerando diferentes cenários de uso e modalidades de instalação

Índice

Lista de Tabelas

Capítulo 1. Preparando o ambiente

1.1. Requisitos de Hardware

Os requisitos de hardware variam dependendo do perfil do middleware a ser carregado no dispositivo. A construção de tais perfis é influenciada não só pelas características de hardware inerentes ao dispositivo, mas também pela estratégia de organização distribuída multi-celular no estilo *super-peers* empregada no GRADEp.

Nessa perspectiva, distinguem-se dois casos:

- **Dispositivo que integra a base virtual da célula.** Corresponde a um dispositivo que hospedará a *instância celular* de algum serviço (um ou mais) do middleware;
- **Dispositivo de processamento.** Caracteriza um nodo que participa da célula disponibilizando sua capacidade de processamento, mas que não desempenha um papel específico, do ponto de vista organizacional da célula, por hospedar apenas *instâncias de nodo* dos serviços.

Em geral, independentemente das especificidades de cada um dos casos (base ou nodo de processamento), o dispositivo deve dispor de recursos de hardware suficientes para suportar alguma modalidade da plataforma Java. Além disso, equipamentos selecionados para integrar a base estarão tipicamente sujeitos a um nível de exigência mais elevado do ponto de vista de estabilidade e capacidade de processamento.

Desconexão planejada e Mobilidade

A estratégia adotada para viabilizar um comportamento mais dinâmico dos nodos de processamento, que não exija conexão permanente do dispositivo à infra-estrutura de rede, requer, por outro lado, que os dispositivos participantes da base da célula tenham um caráter estacionário. Assim, na condição atual, não é suportada a movimentação ou desconexão dos dispositivos que integram a base da célula.

1.1.1. Dispositivos que integram a base virtual da célula

Para dispositivos que participam da base da célula, a implementação atual do middleware requer que tais dispositivos disponham de recursos de hardware compatíveis com (suficientes/que suportem) a plataforma Java 2 Standard Edition (J2SE), na versão 1.3 ou superior.

Perfil de hardware mais restrito utilizado em testes:

Celeron (Mendocino) 300MHz, 128MB RAM, 20MB HD

Adicionalmente, quando dimensionando a capacidade de processamento e armazenamento do dispositivo, deve-se levar em consideração o número de instâncias celulares de serviços do middleware que serão hospedadas no mesmo. Outros aspectos que devem ser considerados incluem: o número de usuários, o número total de nodos que constituem a célula e a expectativa sobre o número de aplicações simultaneamente em execução na célula.

A capacidade de processamento do dispositivo deve ser ajustada por experimentação.

Infelizmente, não existe uma heurística geral razoável para projetar essa carga máxima a ser imposta a um dispositivo participante da base, pois ela pode ser influenciada por características específicas dos serviços hospedados e das aplicações em execução pelos usuários. Observe-se que a infra-estrutura distribuída está potencialmente sendo compartilhada entre diversas aplicações de diversos usuários.

O observado, entretanto, durante o desenvolvimento é que um computador que atenda aos requisitos mínimos de operação das versões recentes (>=1.3) da plataforma Java suporta, com alguma folga, uma célula pequena (+/- 20 nodos).

Tais características influenciam de forma diferente cada um dos serviços. A seguir, são identificados os aspectos mais significativos por serviço:

- **Serviço Logger.** Provê mecanismo de rastro de execução (logging) utilizado por outros serviços do middleware. Esse serviço suporta o redirecionamento das mensagens para arquivo, sendo, nessa modalidade de operação, afetado pela quantidade de espaço livre em disco.

O volume de informação depende do nível de logging habilitado no perfil do middleware.

O nível de logging pode variar, desde a habilitação da gravação apenas das mensagens classificadas como críticas, até a inclusão de mensagens de depuração. Nessa última modalidade, o volume de dados gerado é considerável, podendo atingir rapidamente vários MBytes.

* *Seria interessante não como casos típicos. Configuração do Logger -> Valores min, max.*

- **Serviço Worb.** Implementa um broker de comunicação inter-objetos no estilo chamada remota de método. É, na implementação atual, o mecanismo de comunicação preferencial no middleware. Considerando o pior caso, cada chamada remota corresponde a um socket aberto e a uma thread em execução no dispositivo local além de, possivelmente, a alocação de uma série de objetos em memória. O dispositivo deve, então, dispor de capacidade de processamento e memória para suportar o número máximo de chamadas remotas simultâneas esperado para a célula.

O número de conexões abertas, simultâneas, não é infinito!

Tipicamente, o número de conexões abertas (**sockets**), simultâneas, em um processo é limitado pelo sistema operacional. Contudo, mesmo antes de atingir esse limite o dispositivo pode entrar em uma condição de sobrecarga devido à natureza específica do processamento exigido pelas requisições correntemente sendo tratadas.

Assim, em caso de saturação *o sistema deve ser ajustado por experimentação*, seja modificando o limite atribuído pelo sistema operacional, seja substituindo o dispositivo por um de maior capacidade de processamento.

- **Serviço CellInformationBase.** Aglutina informações dinâmicas e estáticas sobre a composição da célula: usuários registrados, aplicações em execução, nodos participantes, serviços disponíveis etc.

Dessa forma, o volume de informações armazenado na base é proporcional às dimensões da célula e determina a capacidade de memória requerida para o dispositivo que hospeda o serviço.

Back-ends de armazenamento: Hashtable ou LDAP?

A instância celular do serviço CellInformationBase distribuída atualmente com o middleware suporta duas alternativas de back-end de armazenamento: tabelas hash em memória (sem persistência) ou um servidor de diretórios LDAP externo.

No primeiro caso, o dispositivo deve dispor de memória suficiente para acomodar todas as informações da célula nas tabelas hash em memória. No segundo caso, são *adicionalmente* aplicáveis as restrições especificadas pela implementação de servidor LDAP em uso.

Escalabilidade: o servidor LDAP pode estar hospedado em outro nodo!

Caso opte-se pelo uso do servidor LDAP, este não precisa ser hospedado necessariamente no mesmo nodo que hospeda o serviço CellInformationBase. Essa separação pode ser usada como estratégia de distribuição de carga visando a escalabilidade da célula.

- **Serviço CodeRepository (BDA).** Provê o acesso ao repositório de código pervasivo a partir do qual o código dos componentes das aplicações em execução é buscado e instalado sob demanda nos dispositivos da célula. A demanda de espaço em disco requerido por este serviço é determinado pelo volume de código armazenado no repositório.
- **Serviço VirtualEnvironment (AVU).** Provê o repositório pervasivo de dados de usuário (entre estes estão as aplicações instaladas pelo usuário em seu *ambiente virtual*), os quais são, na implementação atual, servidos a partir de arquivos armazenados no sistema de arquivos do dispositivo local. Dessa forma, o serviço VirtualEnvironment requer espaço em disco compatível com o número de usuários da célula e o volume de dados armazenado por cada um destes no seu ambiente virtual.

Aplicações instaladas no ambiente virtual do usuário não ocupam espaço!

Apenas descritores de disparo são mantidos no ambiente virtual do usuário, e não o código das aplicações em si, o qual é mantido no repositório de código. Dessa forma, o consumo de espaço por um dado usuário é pouco influenciado pela aplicações em seu ambiente virtual.

1.1.2. Dispositivos de processamento

Dispositivos sem caráter especial dentro da organização da célula correspondem à condição de consumo mínimo de recursos por parte do middleware. Nessa situação, os requisitos aplicáveis são aqueles referentes ao suporte à plataforma Java em geral, como também aqueles fixados pelas aplicações que se pretende executar naquele dispositivo. O middleware GRADEp, tipicamente, não acrescenta demandas adicionais significativas.

Perfis de hardware mais restritos utilizados em testes

- *PDA:* Sharp Zaurus 7600, processador Intel PXA250 400MHz, 32MB SRAM
- *Desktop:* i686 Celeron (Mendocino) 300MHz, 128MB RAM

1.2. Requisitos de Software

As dependências exatas de software para instalação do middleware GRADEp dependem da modalidade de instalação selecionada e do perfil do middleware a ser utilizado no dispositivo. A lista mínima de dependências, a qual é comum a todas as modalidades, inclui:

Para dispositivos integrantes da base da célula: Java Runtime Environment versão 1.3 superior.

Recomendável J2SE 1.4.2 ou superior

Apesar da compatibilidade esperada com a versão 1.3 da plataforma Java, os testes durante o período de desenvolvimento concentraram-se na versão 1.4.2, então esta deve ser a opção mais estável.

Para dispositivos de processamento:

a. *dispositivos portáteis (PDAs)*: J2ME, perfil CDC;

J2ME / Jeode

Os testes do sistema para J2ME tem se concentrado no PDA Sharp Zaurus 7600, cuja plataforma Java, Jeode, implementa a especificação Personal Java.

b. *dispositivos desktop*: Java Runtime Environment versão 1.2 ou superior.

Java Web Start

Optando-se pelo disparo via Java Web Start, é recomendável utilizar-se J2SE 1.4.2 ou superior.

A seguir, são detalhadas as dependências específicas relacionadas a diversos aspectos da instalação.

1.2.1. Back-ends para a Base de Informações da Célula

A instância celular do serviço CellInformationBase distribuída atualmente com o middleware suporta duas alternativas de back-end de armazenamento:

- `HashtableBackEnd`, o qual não provê persistência e emprega tabelas hash em memória para o armazenamento das informações; e
- `LdapBackEnd`, o qual provê persistência, empregando, para isto, um servidor LDAP externo ao middleware para o armazenamento das informações.

Caso opte-se pela utilização de um servidor LDAP como back-end de armazenamento das informações gerenciadas pela serviço CellInformationBase, acrescente-se como requisito de software:

- Instalação do servidor LDAP em *um* dos nodos do sistema

Servidor LDAP

Recomendado: Openldap (http://www.openldap.org), versão 2.0.27 ou superior.

Distribuição Rocks Linux 3.0 não inclui pacote para servidor LDAP

Optando-se por esse sistema, é necessário compilar o serviço a partir dos fontes ou reaproveitar os `.rpm` disponibilizados para Red Hat.

1.2.2. Web Start

Uma das alternativas de instalação do middleware para os *nodos de processamento* de uma célula está no uso de um mecanismo de disparo a partir da Web. Por conveniência, o disparador Web Start do middleware está encapsulado em um descritor JNLP da tecnologia Java Web Start™[1]. Ressalta-se, entretanto, que o GRADEp Web Start não depende estritamente do Java Web Start para operar.

Caso opte-se pela utilização do mecanismo de disparo via a tecnologia Java Web Start, acrescente-se como requisito de software:

- um servidor Web o qual possa servir os arquivos (código e configuração) para os clientes Web Start;

Servidor WWW

Recomendado: Apache (http://www.apache.org) versão 2.0.54 ou superior.

Não é necessário que o servidor seja de uso exclusivo do middleware.

- suporte Java Web Start habilitado nas máquinas clientes as quais se pretende incorporar como nodos de processamento à célula.

Instalação do Java Web Start

O JWS é instalado automaticamente com as versões 1.4.2 ou superiores do Java Runtime Environment.

1.2.3. Ferramentas auxiliares utilizadas em tempo de instalação

A seguir são descritas as ferramentas auxiliares necessárias a cada uma das modalidades de instalação. *Tais ferramentas não são necessárias posteriormente quando da execução do sistema*.

Acesso direto ao repositório CVS. Uma das modalidades para obtenção do código fonte/módulos do middleware é via a ferramenta CVS. Optando-se por esta modalidade de obtenção, acrescente-se como dependências em tempo de instalação:

- cliente CVS (http://www.cvshome.org), versão 1.11 ou superior;
- cliente SSH (Secure Shell), com suporte ao protocolo SSH v2.

Acesso ao repositório CVS atualmente é restrito

O acesso diretamente ao repositório CVS está restrito aos desenvolvedores cadastrados, não sendo, no momento, permitido acesso anônimo, mesmo que somente para leitura.

Compilação dos módulos Java a partir dos fontes. Optando-se pela compilação a partir dos fontes, ao invés da utilização das classes Java pré-compiladas, tem-se então como dependências em tempo de instalação:

- Java2 SDK, versão 1.4 ou superior;
- Apache ANT (http://ant.apache.org), versão 1.6 ou superior.

Compilação do monitor nativo a partir dos fontes. O middleware disponibiliza, adicionalmente ao monitor independente de plataforma, um monitor nativo, que extrai métricas diretamente a partir do sistema operacional do nodo hospedeiro. Para a compilação do monitor nativo distribuído com o middleware são necessárias as seguintes ferramentas:

- compilador GNU **gcc/g++** (http://www.gnu.org/software/gcc/gcc.html), versão 2.95 ou 3.2 ou superior;
- GNU **autoconf** (http://www.gnu.org/software/autoconf/autoconf.html), versão 2.59;
- GNU **make** (http://www.gnu.org/software/make/make.html), versão 3.80 ou superior,

Java Native Interface: GNU gcc/g++ x Java SDK

A seleção da versão do compilador **gcc/g++** a ser utilizado deve levar em consideração o Java Runtime Environment que será utilizado para executar o sistema. Isto deve-se ao fato de diferentes gerações (2.95.x e 3.x) do **gcc/g++** ligarem o código compilado com versões diferentes da `libc`. Para que a biblioteca nativa seja carregada com sucesso pela JVM, ambas (biblioteca nativa e JVM) devem utilizar a mesma versão da `libc`.

Em específico, caso o JRE selecionado como alvo tenha sido compilado com o compilador **gcc** 2.95, essa mesma versão deve ser utilizada para compilar o monitor nativo. Caso tenha sido compilada com a familia 3.x do compilador **gcc**, a versão 3.2 (ou superior) deve ser usada para compilar o monitor nativo.

Até pouco tempo atrás, o JRE/JDK Linux disponibilizado pela Sun (versão 1.4.x) (http://java.sun.com) era compilado com **gcc** 2.95. Aqueles que preferissem utilizar o **gcc** 3.x deveriam utilizar uma distribuição alternativa da plataforma Java, como aquela disponibilizada pela Blackdown (http://www.blackdown.org), a qual já era compilada com **gcc** 3.2.

Em caso de dúvida, consulte a documentação da plataforma Java que tenha selecionado.

Monitor Nativo: plataformas suportadas

Atualmente, as plataformas suportadas são:

- i386/Linux (kernel 2.4.x e 2.6.x);
- arm/Linux (kernel 2.2); e
- sparc/Solaris (versão 8 ou superior).

1.3. Casos práticos de instalações com sucesso

O GRADEp foi instalado com sucesso nos seguintes sistemas utilizando os pacotes disponibilizados nas seguintes distribuições Linux:

- Gentoo Linux 2004.x e 2005.0;
- Debian Linux sarge (Ago/2005);
- Fedora Core release 3 (Heidelberg);
- Rocks Linux 3.3.0 (Makalu). Nesse caso, foi necessário instalar o servidor LDAP manualmente, assim como o JDK, já que não existem os pacotes correspondentes diretamente na distribuição.

Notas

Java e Java Web Start são marcas registradas da Sun Microsystems Inc.

Capítulo 2. Instalando o middleware

Este capítulo aborda a construção da estrutura de diretórios no sistema de arquivos do dispositivo e a população da mesma com os arquivos do middleware GRADEp. Ao término desta etapa, estarão disponíveis no dispositivo os arquivos executáveis (scripts, classes Java e bibliotecas compartilhadas) necessários à execução do middleware, assim como esqueletos dos arquivos de configuração, cuja personalização é abordada no capítulo seguinte.

Nota:

Os procedimentos de instalação aqui descritos foram depurados mais intensivamente em ambiente *NIX.

2.1. Visão Geral

Diferentes estratégias podem ser empregadas para a instalação do GRADEp. A estratégia de instalação mais adequada em uma dada situação varia dependendo da perspectiva de uso do middleware naquela situação.

São consideradas, nesse documento, três perspectivas de uso do middleware, as quais são caracterizadas a seguir.

Perspectiva dos *desenvolvedores de sistema*. Tais desenvolvedores estarão a cargo tanto da expansão do GRADEp, pela proposição de novos módulos, quanto da manutenção dos já existentes. É importante, portanto, que estes tenham acesso ao código fonte do middleware.

Perspectiva dos *desenvolvedores de aplicação*. Estes não têm interesse particular em modificar os serviços existentes, mas sim em construir aplicações que fazem uso dos serviços disponibilizados pelo middleware. Tais desenvolvedores precisam conhecer as APIs definidas para os serviços e a semântica associada a sua operação sem, no entanto, entrar nos detalhes de como os mesmos foram programados. Prescindem, portanto, do acesso ao código fonte.

Perspectiva dos *usuários finais*. Estes têm interesse em executar aplicações que fazem uso dos serviços disponibilizados pelo middleware, mas não requerem conhecimento das APIs definidas para os serviços ou da forma como os mesmos foram programados. Para usuários finais, é suficiente que a instalação do sistema esteja operacional.

Nesse sentido, duas estratégias de instalação são aqui abordadas:

- geração dos módulos do sistema a partir da compilação do código fonte do middleware, a qual é direcionada aos desenvolvedores de sistema; e
- utilização de módulos previamente compilados, a qual é aplicável a desenvolvedores de aplicação e usuários finais.

Usuários finais podem ainda optar pela estratégia Web Start

Essa estratégia é abordada no capítulo seguinte, no qual são descritos alguns cenários possíveis de configuração para um célula GRADEp.

2.2. Instalando a partir do código fonte

2.2.1. Obtendo o código fonte

A obtenção do código fonte do sistema pode ser feita tanto através do acesso direto ao sistema de controle de versões que gerencia o código fonte, utilizando-se a ferramenta **cvs**, quanto pelo download dos pacotes `.jar` disponibilizados no site do projeto, os quais são atualizados periodicamente a partir do código mantido no repositório CVS.

O acesso ao repositório CVS está restrito aos desenvolvedores registrados

No momento, o acesso anônimo não está disponível. Desenvolvedores não registrados devem, obrigatoriamente, utilizar o download dos pacotes `.jar` disponibilizados no site do projeto.

Alternativa 1: via acesso direto ao repositório CVS. Execute o comando **cvs**, conforme ilustrado a seguir, para obter o módulo `gradep-dev`. Lembre-se de substituir *`user@gradep.mirror:/cvsroot`* pelo endereço do repositório CVS, e correspondente usuário a ser utilizado, onde o código fonte do GRADEp está sendo disponibilizado (e.g., **`elvis@saloon.inf.ufrgs.br:/cvsroot/gradep`**). Consulte o site do projeto para uma lista atualizada dos mirrors disponíveis.

```
# export CVS_RSH=ssh
# cvs -d :ext:user@gradep.mirror:/cvsroot checkout gradep-dev
```

O resultado do comando acima, quando completado com sucesso, é a criação de um diretório `gradep-dev` no dispositivo local. O arquivo `gradep-dev/modules-conf.properties` controla os módulos que serão incluídos no processo de compilação e empacotamento do middleware. A sintaxe desse arquivo segue o formato java.properties. Em específico, a linha:

```
module.N.build.enabled = true
```

controla quando o módulo *`N`* deverá ou não ser incluído no processo de compilação. Linhas iniciando por # correspondem a comentários. Caso não tenha sido realizada a personalização de uma dada propriedade, o valor definido em

```
module.default.property_name = value
```

é utilizado. Fazendo uso do editor de textos de sua preferência, habilite os módulos de interesse no arquivo `modules-conf.properties`, removendo a marca de comentário das linhas correspondentes, e execute o target **fecth** com a ferramenta **ant** para obter, dentro do diretório `gradep-dev/src/`, o código fonte correspondente àqueles módulos:

```
# cd gradep-dev
# vim modules-conf.properties
# ant fetch
```

Dica:

Posteriormente, caso seja necessário incluir mais algum módulo, é suficiente habilitá-lo no arquivo `modules-conf.properties` e executar novamente o **ant fetch**.

Alternativa 2: via download dos pacotes `.jar` a partir do site do projeto. Realize o download do arquivo `gradep-dev-`*`versão`*`.jar`, assim como de todos os módulos opcionais que sejam de interesse (identificados como `gradep-`*`módulo`*`-src-`*`versão`*`.jar`), a partir do site do projeto, utilizando o browser de sua preferência. Concluído o download, crie um diretório para a instalação e expanda o arquivo `gradep-dev-`*`versão`*`.jar`, conforme ilustrado na seqüência de comandos a seguir:

```
# mkdir gradep-dev
# cd gradep-dev
# jar -xvf /caminho/para/gradep-dev-versão.jar
```

Então, dentro do diretório `gradep-dev/src/`, extraia o fonte dos módulos selecionados:

```
# cd src
# jar -xvf /caminho/para/gradep-módulo1-src-versão.jar
# jar -xvf /caminho/para/gradep-módulo2-src-versão.jar
...
# jar -xvf /caminho/para/gradep-móduloN-src-versão.jar
```

Posteriormente, novos módulos podem ser adicionados à árvore local, repetindo-se o procedimento descrito acima, isto é, extraindo-se os `.jar` correspondentes dentro do diretório `src`.

2.2.2. Organização do código fonte

Qualquer que seja o mecanismo empregado para obtenção do código fonte do middleware, após uma operação com sucesso, uma estrutura de diretórios similar à descrita a seguir deverá estar disponível no dispositivo.

```
# ls -1 -F gradep-dev
CVS/ ❶
build/
bin/ ❷
docs/ ❸
lib/ ❹
src/ ❺
util/ ❻
var/ ❼
build.xml ❽
README
modules-conf.properties ❾
```

Organização do código fonte:

❶ Diretórios auxiliares utilizados, respectivamente, pela ferramenta **cvs**, para implementação do sistema de controle de versões, e pelo processo de compilação e empacotamento (ferramenta **ant**), para armazenamento de arquivos temporários relativos à compilação do sistema.

❷ Diretório que contém os scripts (executáveis) de disparo e controle do sistema, assim como arquivos de configuração do perfil de execução do middleware.

❸ Diretório que armazena a documentação gerada a partir do código fonte.

❹ Diretório que contém os `.jar` correspondentes aos módulos do sistema, gerados como resultado do processo de compilação do sistema.

❺ Diretório onde é mantido o código fonte dos módulos do sistema.

❻ Diretório que contém arquivos e ferramentas auxiliares utilizados no processo de compilação e empacotamento do middleware. Entre eles, estão extensões para a ferramenta **ant**, responsáveis pela geração automática de targets a serem incluídos no processo de compilação e empacotamento, diretamente a partir das especificações definidas no arquivo `modules-conf.properties`.

❼ Diretório que contém arquivos gerados ou necessários (potencialmente modificáveis) durante a execução. Entre estes, está o rastro (log) de operação do middleware. Tipicamente, na configuração de bases de célula, o armazenamento persistente correspondente ao repositório de código e ao ambiente virtual do usuário também ficam hospedados sob a árvore `var`.

❽ Parametriza a operação da ferramenta ant, definindo os targets (ações) disponíveis. Tipicamente, não é necessário modificar esse arquivo.

❾ Controla quais módulos serão incluídos no processo de building.

2.2.3. Compilando o código fonte

O processo de compilação e empacotamento do middleware é controlado por meio da ferramenta **ant**, a qual permite automatizar as tarefas relativas à compilação e ao empacotamento dos módulos do sistema, assim como as relativas à geração da documentação da API correspondente (javadoc).

O processamento realizado pela ferramenta **ant** é orientado por targets (alvos ou tarefas), definidos no arquivo `build.xml`. Cada target define um conjunto de ações correspondentes à execução de uma tarefa de mais alto nível. O comando **ant -p** lista os targets públicos (main targets) definidos no arquivo `build.xml`, juntamente com uma breve descrição do seu objetivo:

```
# ant -p
Buildfile: build.xml

     GRADEp Development tree

Main targets:

  build   Incremental build of enabled modules ❶
  deploy  Ship the jars to the execution system ❷
  docs    Build the project documentation ❸
  fetch   Get the latest code from the CVS tree for enabled modules ❹
  main    build and deploy ❺
Default target: main ❻
```

❶ Compila os fontes `.java` armazenados sob `src` habilitados no arquivo `modules-conf.properties`. Os arquivos `.class` resultantes do processo são armazenados sob o diretório `build`.

❷ Empacota os arquivos `.class` gerados no diretório `build` por uma execução anterior do target **build**, gerando os arquivos `.jar` correspondentes sob o diretório `lib`.

❸ Gera a documentação HTML (javadoc) correspondente às APIs dos módulos do sistema a partir dos fontes armazenados sob `src`.

❹ Obtém, via **cvs**, o código fonte correspondente aos módulos habilitados no arquivo `modules-conf.properties`.

Nota:

Esse target só tem sentido para aqueles que optaram pelo uso da ferramenta **cvs** como estratégia de obtenção dos fontes do middleware.

❺ Atalho para a execução em seqüência dos targets **build** e **deploy**.

❻ Indica o target que será executado por default caso a ferramenta **ant** seja chamada sem parâmetros.

A execução de um target pela ferramenta **ant** ocorre informando-se o nome do target como parâmetro:

```
# ant target1 target2 ... targetN
```

Assim, estando-se dentro do diretório `gradep-dev`, executa-se o target **build** para a compilação dos fontes do middleware,

```
# ant build
```

e, posteriormente, o target **deploy**,

```
# ant deploy
```

para a geração dos pacotes `.jar` correspondentes aos módulos do middleware.

Geração da documentação da API

Para a geração da documentação da API, execute o target **docs**.

```
# ant docs
```

A documentação (HTML) gerada estará disponível no diretório `gradep-dev/docs/javadoc`.

Concluída esta etapa com sucesso, o diretório `lib` conterá os pacotes `.jar` correspondentes aos módulos do sistema. Os scripts de controle do middleware automaticamente detectarão todos os módulos armazenados nesse diretório, incluindo-os no CLASSPATH da JVM quando da execução do sistema.

2.3. Instalando a partir de módulos pré-compilados

2.3.1. Obtendo os módulos pré-compilados

A obtenção de módulos pré-compilados do sistema pode ser feita tanto através do acesso direto ao sistema de controle de versões, utilizando-se a ferramenta **cvs**, quanto pelo download dos pacotes `.jar` disponibilizados no site do projeto, os quais são atualizados periodicamente a partir dos módulos mantidos no repositório CVS.

O acesso ao repositório CVS está restrito aos desenvolvedores registrados

No momento, o acesso anônimo não está disponível. Desenvolvedores não registrados devem, obrigatoriamente, utilizar o download dos pacotes `.jar` disponibilizados no site do projeto.

Alternativa 1: via acesso direto ao repositório CVS. Execute o comando **cvs**, conforme ilustrado a seguir, para obter o módulo `gradep`. Lembre-se de substituir *`user@gradep.mirror:/cvsroot`* pelo endereço do repositório CVS, e correspondente usuário a ser utilizado, onde os módulos pré-compilados do GRADEp estão sendo disponibilizados (e.g., **`elvis@saloon.inf.ufrgs.br:/cvsroot/gradep`**). Consulte o site do projeto para uma lista atualizada dos mirrors disponíveis.

```
# export CVS_RSH=ssh
# cvs -d :ext:user@gradep.mirror:/cvsroot checkout gradep
```

O resultado do comando acima, quando completado com sucesso, é a criação de um diretório `gradep` no dispositivo local.

Dica:

Posteriormente, caso seja necessário atualizar a instalação, é suficiente executar o comando

```
cvs update
```

dentro do diretório `gradep`

Alternativa 2: via download dos pacotes `.jar` a partir do site do projeto. Realize o download do arquivo `gradep-`*`versão`*`.jar`, assim como de todos os módulos opcionais que sejam de interesse (identificados como `gradep-`*`modulo-versão`*`.jar`), a partir do site do projeto, utilizando o browser de sua preferência. Concluído o download, crie um diretório para a instalação e expanda o arquivo `gradep-`*`versão`*`.jar`, conforme ilustrado na seqüência de comandos a seguir:

```
# mkdir gradep
# cd gradep
# jar -xvf /caminho/para/gradep-versão.jar
```

Então, instale no diretório `gradep/lib/`, os arquivos `.jar` correspondentes aos módulos do sistema cujo download foi realizado anteriormente.

```
# cd lib
# java -jar /caminho/para/gradep-modulo1-versão.jar
# java -jar /caminho/para/gradep-modulo2-versão.jar
...
# java -jar /caminho/para/gradep-moduloN-versão.jar
```

Evite manter múltiplas versões de um mesmo módulo no diretório `gradep/lib`!

É uma atitude saudável remover versões antigas dos módulos que tenham sido atualizados no diretório `gradep/lib`. Isso deve evitar efeitos colaterais indesejáveis causados pela inclusão de tais versões desatualizadas no CLASSPATH do middleware.

2.3.2. Organização de diretórios da instalação a partir de módulos pré-compilados

Qualquer que seja o mecanismo empregado para instalação a partir de módulos pré-compilados do middleware, após uma operação com sucesso, uma estrutura de diretórios similar à descrita a seguir deverá estar disponível no dispositivo.

```
# ls -1 -F gradep-dev
CVS/ ❶
bin/ ❷
docs/ ❸
lib/ ❹
extras/ ❺
var/ ❻
```

Organização do código fonte:

❶ Diretórios auxiliares utilizados pela ferramenta **cvs**, para implementação do sistema de controle de versões.

❷ Diretório que contém os scripts (executáveis) de disparo e controle do sistema, assim como arquivos de configuração do perfil de execução do middleware.

❸ Diretório que armazena a documentação gerada a partir do código fonte.

A documentação da API não está incluída na instalação padrão

A documentação da API não é incluída na instalação a partir de módulos pré-compilados padrão. É necessário fazer o download do arquivo correspondente do site do projeto. Sugere-se utilizar o diretório `gradep/docs/javadoc` para armazenamento da documentação expandida.

❹ Diretório que contém os `.jar` correspondentes aos módulos do sistema, gerados como resultado do processo de compilação do sistema.

❺ Diretório que inclui arquivos adicionais necessários à instalação do sistema. Em específico, os esquemas LDAP utilizados para a instalação do back-end LDAP da Base de Informações da Célula são disponibilizados nesse diretório.

❻ Diretório que contém arquivos gerados ou necessários (potencialmente modificáveis) durante a execução. Entre eles, está o log de operação do middleware. Tipicamente, na configuração de bases de célula, o armazenamento persistente correspondente ao repositório de código e ao ambiente virtual do usuário também ficam hospedados sob a árvore `var`.

Capítulo 3. Configurando o middleware

Este capítulo aborda o processo de configuração do middleware. Nesse sentido, inicialmente, o documento de especificação de perfis de execução é detalhado. A seguir, o processo de configuração é ilustrado pela construção de perfis de execução para diferentes cenários.

3.1. Perfil de Execução do GRADEp

3.1.1. Estrutura do Documento de Especificação de Perfis

A especificação de perfis de execução (profiles) para o middleware GRADEp dá-se através de um documento XML cujo formato é apresentado na Figura 3-1 utilizando-se XML Schema.

Figura 3-1. Especificação XML Schema do formato do documento de definição de perfis

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:attribute name="name" type="xsd:string"/>
  <xsd:attribute name="value" type="xsd:string"/>
  <xsd:attribute name="impl" type="xsd:string"/>
  <xsd:attribute name="loadPolicy">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="boot"/>
        <xsd:enumeration value="demand"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:element name="PROFILESET">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="PROFILE" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="PROFILE">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="PROP" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element ref="SERVICE" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute ref="name" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="SERVICE">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="PROP" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute ref="name" use="required"/>
      <xsd:attribute ref="impl" use="required"/>
      <xsd:attribute ref="loadPolicy" use="required"/>
    </xsd:complexType>
  </xsd:element>
```

```
  <xsd:element name="PROP">
    <xsd:complexType>
      <xsd:attribute ref="name"  use="required"/>
      <xsd:attribute ref="value" use="required"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

O documento de definição de perfis de execução define um conjunto de perfis através do elemento `PROFILESET`. Dentro de `PROFILESET`, um elemento `PROFILE` especifica a composição de um perfil de execução, sendo caracterizado por um nome (atributo `name`).

Por exemplo:

```
<PROFILESET>
   <PROFILE name="profile1">
   ... elementos internos do profile
   </PROFILE>
   <PROFILE name="profile2">
   ... elementos internos do profile
   </PROFILE>
</PROFILESET>
```

define um conjunto de perfis contendo dois perfis: *`profile1`* e *`profile2`*.

Dentro de um perfil (elemento `PROFILE`), podem ser especificadas zero ou mais propriedades (i.e., pares {nome,valor}) globais àquele perfil através de elementos `PROP`. Deve-se, ainda, definir ao menos um serviço (elemento `SERVICE`) por perfil. A especificação do serviço é caracterizada pelo nome do serviço (atributo `name`), que identifica a interface exportada pelo serviço, e associa ao serviço referenciado uma implementação real, cuja classe Java é identificada no atributo `impl` e uma politica de carga, (definida no atributo `loadPolicy`). No que se refere à politica de carga do serviço, dois valores são suportados: `boot`, indicando que o serviço deve ser carregado já durante a fase bootstrapping do middleware, e `demand`, indicando que o a carga do serviço deverá ocorrer sob demanda, sendo postergada até a ocorrência da primeira requisição direcionada aquele serviço.

Por exemplo:

```
... declarações externas omitidas
   <PROFILE name="profile1">
     <PROP name="p1" value="v1"/>
     <PROP name="p2" value="v2"/>

     <SERVICE name="service1"
              impl="somepackage.S1Impl"
              loadPolicy="boot"/>

     <SERVICE name="service2"
              impl="otherpackage.S2Impl"
              loadPolicy="demand"/>
   </PROFILE>
...
```

define duas propriedades *`p1`* e *`p2`* globais ao perfil *`profile1`*, assim como define dois serviços, *`service1`* e *`service2`*, associando a estes, respectivamente, as classes Java `somepackage.S1Impl`

e `otherpackage.S2Impl`. Além disso, indica que o primeiro serviço deverá ser carregado durante o bootstrapping, enquanto que a carga do segundo ocorrerá sob demanda.

Dentro de um elemento `SERVICE` podem ser definidas propriedades que parametrizam a operação daquela implementação específica do serviço utilizando-se elementos `PROP`, de forma análoga ao que ocorre com as propriedades globais do perfil.

Por exemplo:

```
... declarações externas omitidas
     <SERVICE name="service1"
              impl="somepackage.S1Impl"
              loadPolicy="boot">
       <PROP name="ps1" value="vs1"/>
       <PROP name="ps2" value="vs2"/>
     </SERVICE>
...
```

define duas propriedades *ps1* e *ps2*, cujos valores (*vs1* e *vs2* respectivamente) poderão ser recuperados em tempo de execução pela classe `somepackage.S1Impl` para parametrização de sua operação.

3.1.2. Compondo o Perfil de Execução

Nessa seção é construída uma visão incremental de composição do perfil de execução do middleware, relacionado diferentes níveis de funcionalidade aos serviços resposáveis por sua disponibilização.

1. **O perfil mínimo.** A funcionalidade mínima concebida para um nodo conectado em rede no GRADEp está na capacidade receber e executar requisições de ações remotas, seja na forma explixíta de ações, seja na forma de chamadas remotas as operações exportadas por algum dos serviços hospedados no nodo. O perfil mínino correspondente a provisão desta funcionalidade inclui os serviços Worb, CellInformationBase, Executor e CodeRepository. Esse perfil habilita a execução de ações remotas no nodo, mas não instanciação de componentes gerenciados (OXs) das aplicações.
2. **Acrescentando suporte a instânciação remota e migração de (OXs).** Para suporte a essa funcionalidade deve-se acrescentar ao perfil mínimo os serviços OXManager e ObjectSeed. Nesse nível de funcionalizade, entretanto, a seleção do nodo a hospedar o componente sendo instânciado ou migrado deve ser feita manualmente pelo programador da aplicação. Em específico, não é promovida nenhuma estratégia de balanceamento de carga.
3. **Acrescentando balanceamento de carga quando da instanciação remota ou migração de componentes.** Com essa finalidade, os serviços Scheduler e Collector devem ser adicionados ao profile. O primeiro utiliza informações de monitoração coletadas pelo segundo para orientar as operações de instanciação remota e migração realzadas pelo serviço Executor.
4. **Acrescentando capacidade de acesso a dados armazenados no ambiente virtual do usuário.** O serviço VirtualEnvironment é quem está a cargo da provisão desta funcionalidade.
5. **Acrescentando capacidade de login de usuário e disparo (externo ao middleware) de aplicações.** Essa funcionalidade de interface do middleware com sistemas legados é provida pelo serviço Gatekeeper.
6. **Acrescentando funcionalidade de descoberta de recursos no ambiente da grade pervasiva.** O serviço Discoverer é resposável pela implemetação de mecanismo de descoberta de recursos,

o qual opera executando um casamento entre os atributos definidos para cada recurso e uma lista de atributos de interesse fornecida na requisição de pesquisa.

7. **Acrescentando capacidade de reconhecimento de contextos de alto nível e adaptação em função a modificaçãoes no seu estado.** Essa funcionalidade é provida pelos serviçoes ContextManager e AdaptEngine. Enquanto o primeiro é responsável pela produção da finfação de contexto a partir da filtragem dos dados obtidos junto ao serviço Collector, o segundo provê facilidades para a utilização da mesmo, como o agendamento de acões a serem executadas quando um elemento de contexto de interesse da aplicação assumir um determindo estado.
8. **Acrescentando políticas de controle de acesso aos recursos de processamento.** Essa é uma funcionalidade de perspectiva celular, estando relacionada a operação dos serviçoes ResourceBroker e Gateway.

Funcionalidade do ResourceBroker/Gateway ainda não disponível

Estes serviços, até a data da redação deste documento, ainda não estão integrados a versão estável da plataforma.

3.1.3. Propriedades Globais do Perfil

A operação do middleware é influenciada atualmente por quatro (4) propriedades, relacionadas a identificação do nodo na célula, que não são específicas de um determinado serviço. Tais propriedades, apresentadas na tabela a seguir, são definidas globalmente no perfil de operação do middleware.

Tabela 3-1. Propriedas globais definidas

Propriedade	Semântica
localhost.id	Identificação *única* do nodo no escopo da *célula local*
localhost.cell	Identificação *única* da célula a qual o nodo pertence no escopo da *grade pervasiva*
localhost.name	Nome (ou descrição) do nodo. Tipicamente, o FQDN do nodo. Não é necessário que seja único.
localhost.inetaddr	Endereço Internet (IP) a ser utilizado como endereço de contato do nodo na célula.

Uma estratégia que pode ser empregada para garantir a unicidade do nome da célula, especificado na propriedade localhost.cell, consiste em agregar ao nome selecionado para a célula (único do escopo do ambiente local) um *sufixo construído de forma hieráquica* no escopo do sistema distribuído (único no escopo do ambiente distribuído).

O domínio Internet (DNS) pode ser empregado para gerar nomes únicos para as células

Essa técnica consiste em agregar ao nome (simples) escolhido para a célula, o nome do domínio Internet (DNS) ao qual a célula pertence. Por exemplo, considerando que deseje-se criar uma célula correspondendo ao laboratório 205 do Instituto de Informática da Universidade Federal do Rio Grande do Sul, cujo domínio Internet é `inf.ufrgs.br`, um possível nome para essa célula seria `lab205.inf.ufrgs.br`.

A tabela a seguir define o formato esperado e os valores padrão assumidos em caso de omissão de cada uma das propriedades definidas acima.

Tabela 3-2. Formato e valores padrão das propriedades globais do perfil

Propriedade	Formato	Valor Padrão
localhost.id	inteiro positivo (base 10)	*Autodetectado*. Gerado automaticamente a partir do endereço IP do nodo.
localhost.cell	string alfanumérico [A-Za-z0-9._]*	"acell", **deve ser sobreescrito.**
localhost.name	string alfanumérico [A-Za-z0-9._]*	*Autodetectado*. É utilizada resolução reversa (DNS) do endereço IP do nodo.
localhost.inetaddr	endereço IP (d.d.d.d)	*Autodetectado*. O processo de auto-detecção privilegia IPs reais (i.e. não pertencentes as subredes `127.0.0.0/8`, `10.0.0.0/8` e `192.168.0.0/16`), caso algum esteja associado a alguma das interfaces do nodo.

O processo de auto-detecção do endereço IP pode falhar

Em algumas situações, mais freqüentemente quando o nodo dispõe de múltiplas interfaces de rede, a auto-deteção, que é um processo heurístico, pode não selecionar o endereço IP adequado. Nessa situação, a propriedade localhost.inetaddr deve ser explicitamente definida no perfil para indicar o qual endereço de rede (IP) o sistema deverá utilizar para acessar o nodo.

3.2. Configurando a base da célula

Nos cenários de configuração apresentados a seguir, para efeito de ilustração, vamos assumir que uma implementação da instância celular do serviço ServiceX é provida pela classe `ServiceXSuperPeerImpl`. De forma análoga, assumiremos que uma implementação da instância de nodo deste mesmo serviço será provida pela classe `ServiceXNodePeerImpl`.

Observe-se, ainda, que nem todos os serviços disporão de implementações diferenciadas para base e nodos da célula. Nessa situação, referenciaremos a classe que provê uma implementação para este serviço simplesmente por `ServiceXPeerImpl`

Nesse sentido, a Tabela 3-3 a seguir provê uma síntese das diferenciações *típicas* entre as instâncias celular e de nodo de cada um dos serviços do middleware.

Tabela 3-3. Implementações típicas disponíveis para cada serviço

Serviço	Implementação	Critério de Diferenciação
Logger	`PeerImpl`	-
CellInformationBase	`SuperPeerImpl`	Agrega informações relativas a todos os nodos integrantes da célula local.

Serviço	Implementação	Critério de Diferenciação
	`NodePeerImpl`	Mantém (cache) informações relativas ao nodo local.
Worb	`PeerImpl`	-
Executor	`PeerImpl`	-
CodeRepository	`SuperPeerImpl`	Repositório de todas as aplicações disponíveis na célula. Caching intercelular para código proveniente de outras células.
	`NodePeerImpl`	Mantém (cache) código das aplicações em execução no nodo.
ObjectSeed	`PeerImpl`	-
OXManager	`SuperPeerImpl`	Gerencia os HomeOXHandles.
	`NodePeerImpl`	Gerencia os MasterOXHandles e ProxyOXHandles.
Scheduler	`SuperPeerImpl`	Agrega os índices de carga dos nodos da célula, produzindo um ranking de ocupação dos nodos que será considerado nas decisões de escalonamento na célula.
	`SchedulerNodePeerImpl`	Produz índices de carga que descrever o estado do nodo local.
Collector	`PeerImpl`	-
VirtualEnvironment	`SuperPeerImpl`	Armazenamento persistente (alta capacidade) dos dados que compõem o ambiente virtual do usuário. Coordena as operações intercelulares caso, por exemplo, o dado esteja sendo utilizado fora da célula home do usuário.
	`NodePeerImpl`	Caching do dados recentemente em uso para potencializar a autonomia do dispositivo caso este venha a entrar no modo de operação desconectado.
GateKeeper	`PeerImpl`	-
Discoverer	`SuperPeerImpl`	Controle de leases global da célula. Coordenação das operações de descoberta multi-celular.
	`NodePeerImpl`	Controle de leases correspondentes aos recursos registrados no nodo local.
ContextManager	`SuperPeerImpl`	Produção de informação de contexto que envolve dados de monitoração provenientes de diversos nodos simultaneamente.
	`NodePeerImpl`	Produção de informação de contexto que envolve unicamente sensores disponíveis diretamente no nodo local.
AdaptEngine	`PeerImpl`	-
ResourceBroker	`SuperPeerImpl`	Implementa políticas de controle de acesso e alocação de recursos globais à célula.
	`NodePeerImpl`	Implementa políticas de controle de acesso a alocação específicas do nodo local.

Serviço	Implementação	Critério de Diferenciação
Gateway	`PeerImpl`	-

Implementações reais atualmente disponíveis

Um mapeamento das implementações simbólicas descritas acima para classes reais atualmente existentes no middleware é apresentado no anexo XXX

* *TODO: montar esse anexo.*

3.2.1. Base mono-nodo

Nesse cenário, todas as instâncias celulares de serviços do middleware, as quais em conjunto consituem a base virtual da célula, encontram-se hospedadas em um mesmo nodo físico. Um perfil que atende este cenário é apresentado na Figura 3-2.

Figura 3-2. Perfil de execução do middleware para base mono-nodo

```
<PROFILESET>
  <PROFILE name="base-mono-nodo">
    <PROP name="localhost.cell" value="celula.exemplo"/>
    <SERVICE name="logger"
             impl="LoggerPeerImpl" loadPolicy="boot"/>
    <SERVICE name="worb"
             impl="WorbPeerImpl" loadPolicy="boot"/>
    <SERVICE name="cib"
             impl="CellInformationBaseSuperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="bda"
             impl="CodeRepositorySuperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="executor"
             impl="ExecutorPeerImpl" loadPolicy="boot"/>
    <SERVICE name="oxmanager"
             impl="OXManagerSuperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="objseed"
             impl="ObjectSeedPeerImpl" loadPolicy="demand"/>
    <SERVICE name="scheduler"
             impl="SchedulerSuperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="collector"
             impl="CollectorPeerImpl" loadPolicy="demand"/>
    <SERVICE name="avu"
             impl="VirtualEnvironmentSuperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="gk"
             impl="GatekeeperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="discoverer"
             impl="DiscovererSuperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="logger"
             impl="ContextManagerSuperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="logger"
             impl="AdaptEnginePeerImpl" loadPolicy="demand"/>
    <SERVICE name="rb"
             impl="ResourceBrokerSuperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="gw"
```

```
            impl="GatewayPeerImpl" loadPolicy="boot"/>
  </PROFILE>
</PROFILESET>
```

Política de carga para instâncias celulares (SuperPeerImpl) deve, tipicamente, ser "boot"

Deversas implementações de instâncias celulares dos serviços relizam operações relacionadas a preparação do ambiente de execução da célula. Dessa forma, para correta operação da célula, estas precisam ser disparadas durante o bootstrap da base. Contudo isso não é uma obrigatoriedade.

Em caso de dúvida, consulte a documentação da implementação específica do serviço para determinar quando este exige ou não a política de carga "boot".

3.2.2. Base multi-nodo

Nesse cenário, os serviços que constituem a base virtual da célula, encontram-se fisicamente distribuídos em diversos equipamentos.

Em especifico, no exemplo a ser trabalho nesta seção, a base virtual será constituída de 4 nodos físicos. As instancias celulares dos serviços do middleware serão distribuídas entre estes nodos conforme esquema de alocação apresentado na Tabela 3-4.

Tabela 3-4. Alocação das instancias celulares dos serviços por nodo

Nodo	Serviços
#1	CellInformationBase Discoverer
#2	CodeRepository VirtualEnvironment
#3	Scheduler ResourceBroker Gateway
#4	OXManager ContextManager

O perfil do middleware para cada nodo é construído então complementando as instâncias celulares selecionadas por instancias de nodo para os demais serviços do middleware. A Figura 3-3 exemplifica este procedimento

Figura 3-3. Perfil de execução do middleware para base mono-nodo

```
<PROFILESET>
  <PROFILE name="base-multi-nodo-1">
    <PROP name="localhost.cell" value="celula.exemplo"/>
    <SERVICE name="logger"
             impl="LoggerPeerImpl" loadPolicy="boot"/>
    <SERVICE name="worb"
             impl="WorbPeerImpl" loadPolicy="boot"/>
```

```
  <SERVICE name="cib"
           impl="CellInformationBaseSuperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="bda"
           impl="CodeRepositoryNodePeerImpl" loadPolicy="boot"/>
  <SERVICE name="executor"
           impl="ExecutorPeerImpl" loadPolicy="boot"/>
  <SERVICE name="oxmanager"
           impl="OXManagerNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="objseed"
           impl="ObjectSeedPeerImpl" loadPolicy="demand"/>
  <SERVICE name="scheduler"
           impl="SchedulerNoderPeerImpl" loadPolicy="boot"/>
  <SERVICE name="collector"
           impl="CollectorPeerImpl" loadPolicy="demand"/>
  <SERVICE name="avu"
           impl="VirtualEnvironmentNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="gk"
           impl="GatekeeperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="discoverer"
           impl="DiscovererSuperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="logger"
           impl="ContextManagerNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="logger"
           impl="AdaptEnginePeerImpl" loadPolicy="demand"/>
  <SERVICE name="rb"
           impl="ResourceBrokerNodePeerImpl" loadPolicy="demand"/>
</PROFILE>

<PROFILE name="base-multi-nodo-2">
  <PROP name="localhost.cell" value="celula.exemplo"/>
  <SERVICE name="logger"
           impl="LoggerPeerImpl" loadPolicy="boot"/>
  <SERVICE name="worb"
           impl="WorbPeerImpl" loadPolicy="boot"/>
  <SERVICE name="cib"
           impl="CellInformationBaseNodePeerImpl" loadPolicy="boot"/>
  <SERVICE name="bda"
           impl="CodeRepositorySuperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="executor"
           impl="ExecutorPeerImpl" loadPolicy="boot"/>
  <SERVICE name="oxmanager"
           impl="OXManagerNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="objseed"
           impl="ObjectSeedPeerImpl" loadPolicy="demand"/>
  <SERVICE name="scheduler"
           impl="SchedulerNodePeerImpl" loadPolicy="boot"/>
  <SERVICE name="collector"
           impl="CollectorPeerImpl" loadPolicy="demand"/>
  <SERVICE name="avu"
           impl="VirtualEnvironmentSuperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="gk"
           impl="GatekeeperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="discoverer"
           impl="DiscovererNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="logger"
           impl="ContextManagerNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="logger"
```

```
            impl="AdaptEnginePeerImpl" loadPolicy="demand"/>
  <SERVICE name="rb"
            impl="ResourceBrokerNodePeerImpl" loadPolicy="demand"/>
</PROFILE>

<PROFILE name="base-multi-nodo-3">
  <PROP name="localhost.cell" value="celula.exemplo"/>
  <SERVICE name="logger"
            impl="LoggerPeerImpl" loadPolicy="boot"/>
  <SERVICE name="worb"
            impl="WorbPeerImpl" loadPolicy="boot"/>
  <SERVICE name="cib"
            impl="CellInformationBaseNodePeerImpl" loadPolicy="boot"/>
  <SERVICE name="bda"
            impl="CodeRepositoryNodePeerImpl" loadPolicy="boot"/>
  <SERVICE name="executor"
            impl="ExecutorPeerImpl" loadPolicy="boot"/>
  <SERVICE name="oxmanager"
            impl="OXManagerNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="objseed"
            impl="ObjectSeedPeerImpl" loadPolicy="demand"/>
  <SERVICE name="scheduler"
            impl="SchedulerSuperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="collector"
            impl="CollectorPeerImpl" loadPolicy="demand"/>
  <SERVICE name="avu"
            impl="VirtualEnvironmentNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="gk"
            impl="GatekeeperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="discoverer"
            impl="DiscovererNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="logger"
            impl="ContextManagerNodePeerImpl" loadPolicy="demand"/>
  <SERVICE name="logger"
            impl="AdaptEnginePeerImpl" loadPolicy="demand"/>
  <SERVICE name="rb"
            impl="ResourceBrokerSuperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="gw"
            impl="GatewayPeerImpl" loadPolicy="boot"/>
</PROFILE>

<PROFILE name="base-multi-nodo-4">
  <PROP name="localhost.cell" value="celula.exemplo"/>
  <SERVICE name="logger"
            impl="LoggerPeerImpl" loadPolicy="boot"/>
  <SERVICE name="worb"
            impl="WorbPeerImpl" loadPolicy="boot"/>
  <SERVICE name="cib"
            impl="CellInformationNodePeerImpl" loadPolicy="boot"/>
  <SERVICE name="bda"
            impl="CodeRepositoryNodePeerImpl" loadPolicy="boot"/>
  <SERVICE name="executor"
            impl="ExecutorPeerImpl" loadPolicy="boot"/>
  <SERVICE name="oxmanager"
            impl="OXManagerSuperPeerImpl" loadPolicy="boot"/>
  <SERVICE name="objseed"
            impl="ObjectSeedPeerImpl" loadPolicy="demand"/>
```

```
    <SERVICE name="scheduler"
             impl="SchedulerNodePeerImpl" loadPolicy="boot"/>
    <SERVICE name="collector"
             impl="CollectorPeerImpl" loadPolicy="demand"/>
    <SERVICE name="avu"
             impl="VirtualEnvironmentNodePeerImpl" loadPolicy="boot"/>
    <SERVICE name="gk"
             impl="GatekeeperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="discoverer"
             impl="DiscovererNodePeerImpl" loadPolicy="demand"/>
    <SERVICE name="logger"
             impl="ContextManagerSuperPeerImpl" loadPolicy="boot"/>
    <SERVICE name="logger"
             impl="AdaptEnginePeerImpl" loadPolicy="demand"/>
    <SERVICE name="rb"
             impl="ResourceBrokerNodePeerImpl" loadPolicy="demand"/>
  </PROFILE>
</PROFILESET>
```

3.2.3. Conectando células

Vizinhança estática e dinâmica

3.3. Configurando nodos de processamento

Montando o perfil de execução

Figura 3-4. Perfil de execução do middleware para nodo de processamento

```
<PROFILESET>
  <PROFILE name="base-multi-nodo-1">
    <PROP name="localhost.cell" value="celula.exemplo"/>
    <SERVICE name="logger"
             impl="LoggerPeerImpl" loadPolicy="boot"/>
    <SERVICE name="worb"
             impl="WorbPeerImpl" loadPolicy="boot"/>
    <SERVICE name="cib"
             impl="CellInformationNodePeerImpl" loadPolicy="boot"/>
    <SERVICE name="bda"
             impl="CodeRepositoryNodePeerImpl" loadPolicy="boot"/>
    <SERVICE name="executor"
             impl="ExecutorPeerImpl" loadPolicy="boot"/>
    <SERVICE name="oxmanager"
             impl="OXManagerNodePeerImpl" loadPolicy="demand"/>
    <SERVICE name="objseed"
             impl="ObjectSeedPeerImpl" loadPolicy="demand"/>
    <SERVICE name="scheduler"
             impl="SchedulerNoderPeerImpl" loadPolicy="boot"/>
    <SERVICE name="collector"
             impl="CollectorPeerImpl" loadPolicy="demand"/>
    <SERVICE name="avu"
```

```
                impl="VirtualEnvironmentNodePeerImpl" loadPolicy="demand"/>
      <SERVICE name="gk"
                impl="GatekeeperPeerImpl" loadPolicy="boot"/>
      <SERVICE name="discoverer"
                impl="DiscovererNodePeerImpl" loadPolicy="demand"/>
      <SERVICE name="logger"
                impl="ContextManagerNodePeerImpl" loadPolicy="demand"/>
      <SERVICE name="logger"
                impl="AdaptEnginePeerImpl" loadPolicy="demand"/>
      <SERVICE name="rb"
                impl="ResourceBrokerNodePeerImpl" loadPolicy="demand"/>
    </PROFILE>
  </PROFILESET>
```

conectando nodos à base

3.4. Construindo um célula Web Start

preparando o descritor de disparo

Apêndice A. Respostas para Perguntas Freqüentes

1. Dúvidas gerais

1.1. Onde obtenho o middleware?

1.2. É mesmo necessário instalar uma base para fazer os primeiros testes?

1.3. Como posso testar o funcionamento mínimo do middleware após sua instalação?