Home

Powered by Open Sources Licences from:
Adaptavist Theme Builder, Atlassian Jira, Atlassian Confluence, GreenHopper Plugin, Yourkit Java Profiler

PROJETO (Product Backlog, Taskboard, Burndown) | RELATORIOS (Svn, Timetracking, Changelog) | COMUNIDADE (Forum Google, Forum 3layer, Blog, Noticias) | DOWNLOADS (Latest, Nightbuild, Svn)

Introdução

Bem-Vindo.

Magoo é um framework diferenciado para o gerenciamento de permissões de usuários em aplicações java, sejam web ou desktop. Utilizando a premissa de não-intrusividade, qualquer aplicação existente pode ser gerenciada por ele, independente da sua arquitetura interna ou tecnologias utilizadas em sua construção. Tendo uma base neutra de usuários e permissões, ele é aderente ao JAAS e pode ser facilmente integrado a mecanismos legados ou serviços de diretório, como o Active Directory. Com um modelo de informações totalmente OO, permite herança de permissões e a vinculação dessas tanto a dados quanto à regras de negócio. Tendo estruturas de cache hierárquicos extremamente eficientes e com um exclusiva tecnologia de intersão de bits, sua performance é excepcional, seja qual for o modelo de persistência adotado para as permissões e usuários. Possuindo um modelo centralizado de permissões uma granularidade escalável é possível definir regras desde um simples controle em uma tela até toda uma federação de sistemas, em diversos ambientes e de forma instantânea.

Características

- Abordagem não-intrusiva, suportando aplicações legadas sem alteração de código existente
- Controle total de permissões de telas em aplicativos Web/HTML, independente de tecnologia *server side*
- Controle total de permissões de telas em aplicativos Desktop, para os frameworks Swing, SWT, AWT e Thinlet
- Identificação automática dos objetos gerenciáveis
- Tecnologia de inversão de bit, que otimiza drasticamente o volume de dados
- Alta performance, com regras baseadas em caches hierárquicos clusterizados
- Modelo neutro de permissões, usuários e perfis
- Base única para usuários e permissões
- Modelo de dados totalmente OO, com suporte à herança para permissões, usuários e perfis
- Armazenamento de regras em banco relacional, arquivos planos ou XML
- Totalmente integrável à bases de usuários legadas, serviços de diretório (como Active Directory) e JAAS
- Granularidade escalável, desde um controle em uma tela até uma federação de sistemas
- Permissões baseadas em Expressões Regulares e XPath[1]
- Vínculação de permissões à regras de negócio e dados[1]
- API baseada em padrões abertos de mercado (somente necessária em casos extremos)
- Operação standalone ou integrado em servidores de aplicação como o Tomcat, JBoss, Jetty, JRun e Geronimo
- Gerenciamento textual e visual de permissões
- Suporte comercial pela 3Layer Tecnologia

NOTA: (1) Recurso em pesquisa.

Motivação e Objetivos

De forma geral, o modelo de permissões e usuários de uma aplicação é intrinsicamente ligado ao seu código-fonte. Para dar um exemplo, considere esse fragmento de código em uma pseudo-linguagem:

```
<ui:menu>
  <ui:menuItem>File</ui:menuItem>
  <ui:menuItem>Edit</ui:menuItem>
  <if:$user.role="owner">
    <ui:menuItem>Tools</ui:menuItem>
  </if>
  <ui:menuItem enabled='<%=user.role=="admin"%>'>Configuration</ui:menuItem>
</ui:menu>
```

O objetivo desse código é exibir um conjunto de itens de menu numa tela, sendo que usuários do tipo "owner" podem ***ver*** a opção "Tools" e usuários "admin" têm ***habilitada*** a opção "Configuration".

Esse exemplo mostra claramente a dependência do código-fonte do sistema em relação à estrutura e conteúdo do controle de acesso. Nessa aplicação, caso se deseja trocar o pertil "owner" por "dono" o custo de refatoração é extremanente alto. Da mesma forma, caso fosse necessário habilitar o menu "Configuration" para todos os usuários, sem que eles se tornassem administradores, isso novamente levaria à alteração de código-fonte.

Foi pensando nesse cenário que o Magoo foi projetado. A idéia básica por trás do Magoo é retirar totalmente do código da aplicação qualquer referência ao mecanismo de controle de acesso. Assim, o código da aplicacão seria simplesmente isso:

```
<ui:menu>
  <ui:menuItem>File</ui:menuItem>
  <ui:menuItem>Edit</ui:menuItem>
  <ui:menuItem>Tools</ui:menuItem>
  <ui:menuItem>Configuration</ui:menuItem>
</ui:menu>
```

Mas como então é feito o controle dos menus "Tools" e "Configuration"?

A resposta é: De forma externalizada[2].

E para demonstrar isso, consideramos um exemplo fictício de configuração do Magoo, onde as permissões, perfis e elementos de tela são armazenados em um arquivo de texto plano, como:

```
*     : disabled  : //ui/menuItem[Configuration]
*     : invisible : //ui/menuItem[Tools]
admin : enabled   : //ui/menuItem[Configuration]
owner : visible   : //ui/menuItem[Tools]
```

Nesse arquivo, as duas primeiras linhas referem-se a todos (asterisco) os perfis, e indicam que eles têm o menu "Configuration" desabilitado e o menu "Tools" invisível. Logo abaixo, as duas últimas linhas denotam que o administrador tem o menu "Configuration" habilitado e que o perfil "owner" tem o menu "Tools" visível.

A mesma configuração poderia ser feita de forma otmizada, usando a palavra reservada `except`, como abaixo:

```
* except(admin) : disabled  : //ui/menuItem[Configuration]
* except(owner) : invisible : //ui/menuItem[Tools]
```

Isso, produz o mesmo resultado de antes, ou seja, todos os perfis, com exceção do "admin" vão ter a entrada de menu "Configuration" desabilitada. E todos os perfis, com exceção do "owner" não poderão exergar a entrada de menu "Tools".

E qual a vantagem nisso? Algumas são:

- A aplicação pode ser construída sem se preocuparmos com o controle de acesso;
- O código-fonte da aplicação fica menor e mais legível;
- E, consequentemente, uma refatoração nas regras de permissão não afeta mais o código-fonte. Por exemplo, trocar o perfil "owner" por "dono" não implica em nada além de alterar esse valor na configuração. Da mesma forma, permitir o acesso de todos os usuários ao menu "Configuration" simplesmente implica em remover ou comentar a primeira linha da configuração!

Obviamente, esse é um exemplo bem simples. Entretanto, inúmeras variações podem ser feitas, sem que a essência se perca.

Complementando, salienta-se que também o suporte à permissões baseadas em regras de negócio, e mesmo permissões sobre o conteúdo de controles (por exemplo, um usuário administrador que poderia visualizar mais itens em uma combobox do que um usuário comum) é possível e com a mesma flexibilidade.

NOTA: (2) Este mesmo tipo de permissionamento declarativo já é utilizado em ambientes java web há vários anos. Entretanto, as soluções existentes carecem do compatilhamento eficiente de usuários e perfis entre sistemas diferentes, não são capazes de aplicar permissões em nveis mais granulares (tais como um controle de tela dentro de uma página), não possuem um mecanismo inteligente de descoberta e catalogação de informações a serem gerenciadas, não suportam alterações em tempo de execução, exigem a alteração de pelo menos alguma parte (arquivos de configuração) da aplicação para que a segurança seja aplicada e outros elementos. Todas essas necessidades são transparentemente implementadas pelo Magoo.

Status do Projeto

Neste inicio de 2009, o projeto foi retomando. Com um reescalonamento de prioridades, esperamos que para o primeiro semestre de 2009 ja tenhamos uma versao operacional, com suporte para aplicacoes JSF, e modulo administrativo de gerenciamento em uma interface GWT.

Histórico

- 2009/02 - Retomada do projeto
- 2008/03 - Projeto movido para o servidor da 3Layer Tecnologia
- 2007/08 - Estudo de tecnologias - Código funcional para web, com permissões baseadas em arquivos planos.
- 2007/07 - Projeto criado - Nada operacional.

Empresa	Produtos	Usos	Mais informacoes
	• Java Profiler	Profiling e analise de memoria e processamento das aplicacoes Merlin, Magoo e outros projetos da area Incubator	Macro desconhecida:'html'