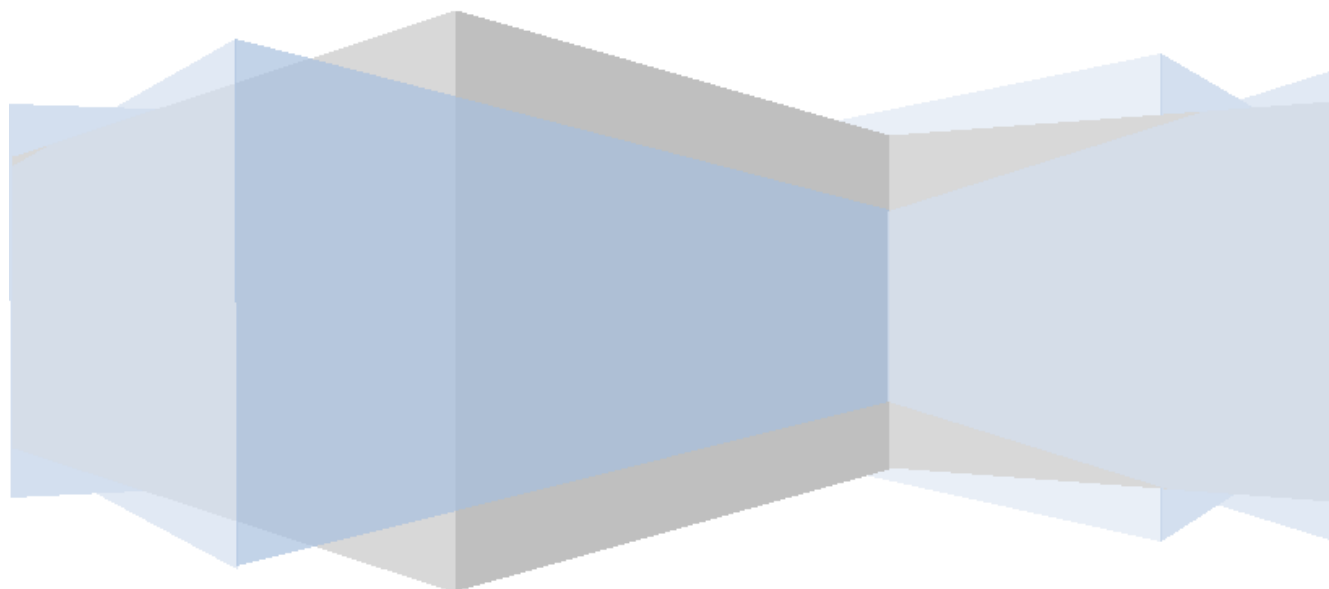


# Início em OpenACS

## Tutorial da Ferramenta de Desenvolvimento de Software OpenACS

César Clavería – Rocael Hernández – Tradução em português: Iuri Sampaio



**INÍCIO EM OPENACS:  
SEU GUIA RÁPIDO PARA UMA PODEROSA FERRAMENTA DE SOFTWARE.**

**CONTEÚDO**

Iniciando um aprendizado com OpenACS Guia para uma poderosa Ferramenta de Desenvolvimento de Software ..... 2

Como este tutorial está organizado. .... 2

Dia 1 .. .

Objetivos .. .

Seção 1 .. .

    Nesta sessão você aprenderá .. .

Instalação .. .

    Linux: Instalando OpenACS no sistema Operacional Linux Ubuntu .. .

    Instalador do Windows .. .

TCL Básico .. .

    O que é TCL .. .

    Porquê TCL .. .

    Exemplos e Fontes .. .

Minha Primeira Página .. .

    Minha Primeira ADP .. .

    Exemplos .. .

    Passando variáveis entre páginas .. .

DIA 2 .. .

Objetivos .. .

Seção 2 .. .

    Nesta sessão você aprenderá .. .

    Meu primeiro pacote .. .

    A aplicação “To Do” .. .

    Criar um novo pacote .. .

    Montar um novo pacote .. .

Seção 3 .. .

    Nesta sessão você aprenderá .. .

    O modelo de dados da aplicação To Do .. .

    Adicionando algumas entradas .. .

Dia 3 .. .

Objetivos .. .

Seção 4 .. .

    Nesta sessão você aprenderá .. .

    Criando páginas acessíveis ao usuário .. .

    Removendo itens .. .

# OpeACS.org

Atualizando o estado de um item .....	
Seção 5 .....	
Nesta sessão você aprenderá .....	
Adicionando um API TCL num pacote .....	
Dia 4. ....	
Objetivos .....	
Seção 6 .....	
Nesta sessão você aprenderá .....	
Objetos ACS .....	
Como utilizar os objetos ACS .....	
Usando os objetos ACS .....	
Mudanças nas páginas do pacote .....	
Seção 7 .....	
Nesta sessão você aprenderá .....	
Integrando com outros serviços .....	
Permissões .....	
Finalizando a transição .....	
Conclusões .....	
Anexo 1 .....	
Anexo 2 .....	
Anexo 3 .....	



### COMO ESTE TUTORIAL ESTÁ ORGANIZADO

Este tutorial está dividido em 4 dias (De fato 3 dias e meio! Embora é conhecido que muitas pessoas o terminem antes disso). Cada dia possui duas seções, no total existem sete seções. Cada seção guiará você através de um conjunto de atividades, que o manterão aprendendo as características poderosas de desenvolvimento com OpenACS.

Este tutorial é baseado na utilização de um sistema OpenACS que funciona sobre o sistema operacional Linux Ubuntu. Desta forma, alguns exemplos de comandos bash e caminhos de diretórios são baseados numa instalação Ubuntu. Todavia, isto não limita o alcance deste tutorial a apenas sistemas Linux. É possível acompanhar o tutorial utilizando outros sistemas operacionais tais como: Windows (XP e Vista), Solaris, Mac OS2.

Ao final deste tutorial você encontrará uma tabela com importantes diretórios listados para ambos sistemas Windows e Linux. Isto poderá ajudá-lo no caso de utilizar o Windows como sistema operacional base.

Em cada seção você encontrará elementos diferentes. Uma seção tipicamente consiste em:

- O número do dia em que a sessão é esperada para ser concluída
- O que você aprenderá em cada sessão
-  Notas extras sobre tópicos relacionados
-  Arquivos que serão trabalhados
- Um ícone chamado “Try it” para práticas e testes

Algumas vezes após a descrição de um exemplo você encontrará um ícone “Try it!”. Ele sugere que você coloque em prática o que acabou de ler no tutorial, além de treinar e testar simultaneamente os exemplos de código apresentados.

Normalmente é dado um endereço URL. Ele referencia o local em que o sistema está instalado. Caso o endereço não funcione, basta voltar atrás no navegador, na última seção em que estava trabalhando.

### DIA 1

Objetivo: Neste primeiro dia é mostrado como instalar, configurar e executar um sistema OpenACS. Além de criar a primeira página web utilizando OpenACS

### SESSÃO 1

NESTA SEÇÃO VOCÊ APRENDERÁ:

- ✓ **COMO CONFIGURAR UM SISTEMA WEB OPENACS**
- ✓ **O QUE É TCL E ONDE APRENDER MAIS SOBRE ELE**
- ✓ **COMO CRIAR PÁGINAS WEB SIMPLES COM OPENACS**

### INSTALAÇÃO

LINUX:

Você consegue colocar todo o sistema pronto, configurado e rodando em poucos minutos, seguindo as etapas abaixo para instalar OpenACS no Ubuntu. O instalador foi criado para trabalhar sobre a versão Ubuntu 8.04 (Hardy Heron). Porém ele também funciona facilmente em outras versões com apenas algumas modificações.

Siga estas etapas para Ubuntu 8.04 e Ubuntu 8.10:

1. Adicione o seguinte repositório utilizando o gerenciador de pacotes ou adicione manualmente a linha abaixo no arquivo `/etc/apt/sources.list`:

- `deb http://debian.adenu.ia.uned.es/apt hardy main`

2. Atualizar o repositório. Execute o comando:

- `sudo apt-get update`

3. Instalação do PostgreSQL. Execute o comando:

- `sudo apt-get install postgresql-8.2`

4. Instalação do OpenACS. Execute o comando:

## OpeACS.org

- sudo apt-get install openacs

5. Iniciar ou Reiniciar o serviço OpenACS. Execute o comando:

- sudo /etc/init.d/openacs start

Siga estas etapas para Ubuntu 9.04:

1. Adicione o seguinte repositório utilizando o gerenciador de pacotes ou adicione manualmente a linha abaixo no arquivo `/etc/apt/sources.list`:

- deb <http://debian.adenu.ia.uned.es/apt> hardy main
- deb <http://archive.ubuntu.com/ubuntu/> intrepid universe

2. Execute o comando:

- sudo apt-get update

3. Instalação do Postgresql. Execute o comando:

- sudo apt-get install postgresql-8.2

4. Instalação do OpenACS. Execute o comando:

- sudo apt-get install openacs aolserver4-nscache

5. Iniciar ou Reiniciar o serviço OpenACS. Execute o comando:

- sudo /etc/init.d/openacs start

Você encontra maiores informações e instruções atualizadas na página Wiki oficial de instalação do OpenACS: <http://openacs.org/xowiki/ubuntu>

Além da instalação no sistema operacional Ubuntu, no anexo C segue o script utilizado para instalação do OpenACS no sistema operacional Debian Etch 4

Após realizar a instalação do sistema abra o navegador de internet e digite o endereço <http://localhost:8000> e preencha o formulário que será apresentado na página e confirme. Isto iniciará o processo de instalação do sistema na máquina. Basicamente os scripts de instalação carrega o banco de dados e configura automaticamente o sistema. Após terminado o processo de instalação é preciso reiniciar o serviço openacs. Para isso, basta executar o comando

- sudo /etc/init.d/openacs restart

## OpeACS.org

O usuário Ubuntu provavelmente não tem permissão de escrita nos diretórios openacs. Para corrigir este problema, basta executar os comandos no terminal:

- `sudo usermod -a -G www-data ubuntuuser`
- `sudo chmod -R 775 /usr/share/openacs/www`

### INSTALADOR WINDOWS:

Você pode baixar um programa instalador para windows (XP e Vista 32 bits). O instalador apresentará caixas de diálogo para instalação e configuração do sistema. Para instruções detalhadas visite:

[http://www.friendlybits.com/en/inf\\_tec\\_en/win32openacs\\_en/](http://www.friendlybits.com/en/inf_tec_en/win32openacs_en/)

No caso de utilizar o instalador do windows, mantenha-se alerta para:



- O instalador instala 2 serviços. Por favor, utilize o serviço nomeado como OpenACS
- Antes de iniciar o serviço openacs, inicie o banco de dados primeiro
- No Windows Vista você precisará iniciar e parar o serviço de banco de dados e o serviço openacs com o usuário que possua privilégios de administrador

## TCL BÁSICO

### O que é TCL

Originalmente do inglês “Tool Command Language”, Tcl é uma linguagem de programação altamente flexível e fácil de usar. Tcl é a linguagem que será utilizada no trabalho com OpenACS. Então se você ainda não a conhece, este é um momento bastante oportuno para praticar e aprender um pouco mais sobre ela.

### Porquê TCL

O conjunto de ferramentas contido em OpenACS é utilizado com o servidor HTTP AOLSERVER. TCL é a linguagem embutida e utilizada nos scripts do AOLServer. Desta forma é natural que TCL seja a linguagem de programação utilizada na escrita dos códigos das aplicações.

## OpeACS.org

Exemplos e Fontes:

O livro de referências altamente recomendado está disponível em <http://philip.greenspun.com/tcl/> . Ele é gratuito e de fácil leitura. Com ele, você aprenderá TCL em apenas alguns minutos. E pelo fato de ser um livro de referências mantenha-o sempre ao lado enquanto desenvolve suas aplicações em OpenACS.

### MINHA PRIMEIRA PÁGINA

- Arquivos ADP

ADP consiste em páginas dinâmicas do AOLServer. Elas são muito semelhantes às páginas HTML, porém incluem mais algumas tags e extensões que permitem a manipulação de informações dinâmicas provenientes das páginas TCL associadas.

O processo de criação de uma página dinâmica visível ao usuário em OpenACS é dividido em dois ou mais arquivos ou páginas: um arquivo com extensão .tcl e um arquivo .adp. Consequentemente, para uma página nomeada como “teste” você terá os arquivos: teste.adp e teste.tcl

A página ADP é apenas uma página HTML com as marcações necessárias para mostrar os dados dinâmicos

- Arquivos TCL

O arquivo .tcl trata de permissões, lógica de programação, cálculos e disponibiliza os dados para as páginas ADP. É nesta página que programador realmente escreve os códigos, scripts, utilizando a linguagem de programação Tcl.

- Arquivos XQL

Existe um terceiro tipo de arquivo envolvido no contexto de uma página em OpenACS. No arquivo xql, ex. teste.xql, são definidas as consultas de banco de dados utilizadas na recuperação de dados disponíveis no banco de dados associado ao sistema. Este arquivo possui extensão xql



A nomenclatura destas páginas é realmente importante.

Lembre-se de que Tcl é uma linguagem de programação Case Sensitive. Ou seja, TESTE.adp ,



## OpeACS.org

teste.adp , TesTe.adp , tEstE.adp são consideradas páginas distintas pelo interpretador de nomes de páginas.

Exemplos:

Num exemplo bem básico de uso de Tcl na criação de páginas, você pode criar uma variável, atribuir um valor a esta variável e recuperá-la no arquivo ADP associado. Neste momento esta variável torna-se uma fonte de dados para a página ADP.

Para editar os arquivos, abra um editor de texto (Notepad++, Gedit, etc.) e siga os exemplos seguintes.

Exemplo 1: Hello World



helloworld.tcl

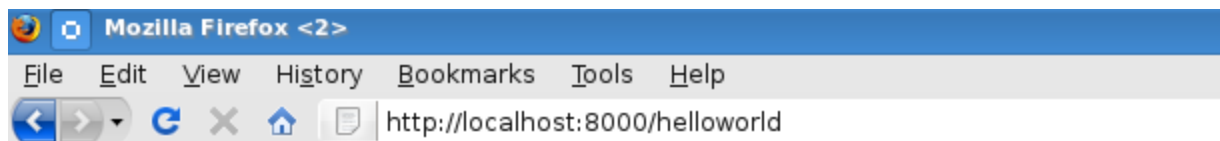
```
set hello "Hello World!"
```



helloword.adp

```
Hey, @hello@
```

Estas duas linhas bem simples geram a primeira página em OpenACS. Você pode criar ambos arquivos dentro do diretório `/usr/share/openacs/www` no seu sistema de teste e pode visualizar sua página simplesmente abrindo no seu navegador o endereço url <http://localhost:8000/helloworld> . Note que não é necessário colocar a extensão do arquivo. Você deverá visualizar algo como mostrado na figura:



Hey, Hello World!

## OpeACS.org

### Exemplo 2: Informação de Conexão

Neste exemplo são mostradas algumas informações básicas sobre conexão utilizando a API chamada `ad_conn`. Uma das funções prontas existentes na biblioteca do OpenACS . As informações são mostradas numa lista não ordenada simples.



connection.tcl

```
set user_id [ad_conn user_id]
set url [ad_conn url]
set session_id [ad_conn session_id]
set IP [ad_conn peeraddr]
```



connection.adp

```
<h2>Basic Connection Information:</h2>

<ul>
<li>User Id: @user_id@</li>
<li>This URL: @url@ </li>
<li>This session: @session_id@ </li>
<li> IP Address: @IP@ </li>
</ul>
```

Abra seu navegador de internet e visite <http://localhost:8000/connection> e você poderá visualizar algo como:

### **Basic Connection Information:**

- User Id: 568
- This URL: /connection
- This session: 30002
- IP Address: 64.191.80.135

## OpeACS.org

Perceba que os arquivos ou páginas criadas não são atrativas. Elas não possuem nenhum apelo visual e não existe título nas páginas. Elas são bastante simples e existe uma forma muito fácil de melhorar a forma como elas aparecem em seu navegador, simplesmente adicionando algumas linhas de código no arquivo ADP. Vamos melhorar a página mostrando um exemplo adicionando uma tag chamada “master” no início do arquivo connection.adp

Exemplo 3: Melhorando a apresentação da página de informações básicas de conexão connection.adp:



connection.adp

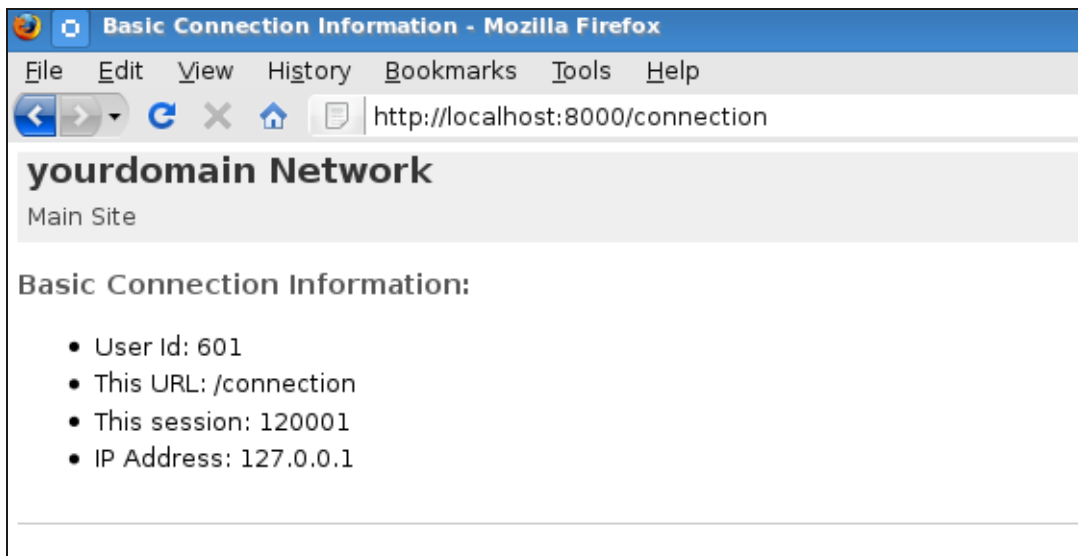
```
<master>
<property name="doc(title)">Basic Connection Information </property>

<h2>Basic Connection Information:</h2>

<ul>
<li>User Id: @user_id@</li>
<li>This URL: @url@ </li>
<li>This session: @session_id@ </li>
<li> IP Address: @IP@ </li>
</ul>
```

Você deverá visualizar a página com uma moldura. Esta moldura é o padrão utilizado pela tag “<master>”. Esta tag inclui o cabeçalho (header) e rodapé (footer) na página. Além disso ela inclui folhas de estilo CSS e outras propriedades.

Desta forma você consegue criar um site com uma aparência e estrutura unificada e bem definida. Além disso, existe a tag “<property>” . Com ela você pode adicionar novos atributos como título, contexto e etc. características a própria página.



Seu site deve parecer com a figura acima. A aparência dos sites podem mudar de sistema para sistema. Depende da moldura (template), escolhido para o site. Uma instalação básica deve parecer com esta figura.

---

Passando variáveis entre páginas pelo url (link).

---

Para certificar-se da presença das informações requisitadas na página TCL, OpenACS utiliza um sistema de contrato simples e poderoso entre páginas TCL e ADP. Isto é estabelecido pelo procedimento `ad_page_contract` na página TCL. Será construído um exemplo bem simples de uma página para ilustrar.

O exemplo apresenta um formulário simples a ser preenchido em uma página e uma outra página para apresentar as informações. O formulário contém as informações de nome e idade a serem preenchidas e apresentadas na outra página, o nome, a idade e se a pessoa é adulto.

 nameage.tcl

nothing here

## OpeACS.org



nameage.adp

```
<master>

<form action="display-name-age">
  <label for="name">Name</label>
  <input type="text" name="name"/><br />
  <label for="age">Age</label>
  <input type="text" name="age"/><br />
  <input type="submit">
</form>
```



display-name-age.tcl

```
ad_page_contract {
  This page will display the name and age entered on the nameage page.
  And this page is role is pretty much just to accept and validate the data before displaying it.
} {
  age:integer
  name
}
```



display-name-age.adp

```
<master>
<property name="title">Page for @name@</property>

<ul>
  <li>Name: @name@</li>
  <li>Age: @age@</li>
</ul>

<if @age@ ge 18>
  @name@ is an adult.
</if>
<else>
  @name@ is still a minor.
</else>
```

Objetivos: Agora que você já sabe como criar páginas. Neste dia aprenderemos com criar um novo pacote e como trabalhar com ele dentro do sistema.

## SESSÃO 2

NESTA SEÇÃO VOCÊ APRENDERÁ:

- ✓ **COMO CRIAR UM PACOTE**
- ✓ **COMO CONFIGURAR UM PACOTE**
- ✓ **COMO MONTAR UM PACOTE**
- ✓ **QUAL A APLICAÇÃO CRIADA NESTE TUTORIAL**

## MEU PRIMEIRO PACOTE

Mesmo quando o sistema consiste basicamente de páginas simples e diretórios você pode construir eficientes peças com OpenACS. O grande poder desta ferramenta fundamenta-se com este sistema de pacotes. Para ilustrar, construiremos um pacote simples no sistema. Uma aplicação “To Do” que iremos integrá-la como um novo pacote OpenACS.

---

### A aplicação “To Do List”

---

Os requisitos para a aplicação são bem simples. É necessário criamos uma lista básica de atividades chamadas também de itens, com as seguintes características.

- ✓ Permitir que usuários mantenham uma lista de itens que eles precisam cumprir antes de uma data determinada.
- ✓ Os usuários devem visualizar e modificar somente suas listas respectivas.

## OpeACS.org

- ✓ As informações que cada item contém são: Título, Descrição da tarefa, data-limite, estado corrente do item.

É necessário dar ao usuário uma interface para:

- ✓ Visualização de suas listas de itens
- ✓ Adicionar um novo item na lista
- ✓ Editar um item

---

### Criando um novo pacote

---

No nível mais básico um pacote OpenACS nada mais é do que um novo diretório dentro do diretório packages. Basta verificar em `/usr/share/openacs/packages/`, com os arquivos TCL e ADP necessários e um arquivo com metadados sobre o próprio pacote.

Não é preciso escrever todos estes arquivos. O caminho apropriado para criar o pacote é seguindo para o gerenciador de pacotes e selecionando a opção ou link para criar o pacote.

Visite a seção de administração do site <http://lcoalhost:80/acs-admin/>

Então, siga o link “Developer's Admin” e clique no gerenciador de pacotes “Package Manager” <http://lcoalhost:80/acs-admin/apm/>

Obtém-se uma lista numa página de todos os pacotes instalados no sistema, que corresponde aos sub-diretórios do diretório packages localizado em `/usr/local/aolserver/servers/openacs/`

Use a barra de rolagem e desça até o final da página e encontrará um link para criar um novo pacote “Create a new package”.

## OpeACS.org

profile-provider	Profile Provider	2.3.1	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>
ref-timezones	Reference Data - Timezone	5.3.2	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>
rss-support	RSS Support	0.3	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>
search	Search	5.3.2	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>
static-portlet	Static Portlet	2.3.1	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>
theme-zen	Zen Theme	2.3.1	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>
user-profile	User Profile	2.3.1	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>

- [Create a new package.](#)
- Write new specification files for all installed, locally generated packages
- Load a new package from a URL or local directory.
- Install packages.


**Help**

A package is **enabled** if it is scheduled to run at server startup and is deliverable by the request processor.

Após seguir o link, você será levado para uma página para entrar com as informações do novo pacote. Preencha o formulários com as seguintes informações.

<b>Package Key</b>	todo
<b>Package Name</b>	To Do List
<b>Package Plural</b>	To Do Lists
<b>Package Type</b>	Application
<b>OpenACS Core</b>	Leave it unchecked
<b>Singleton</b>	Leave it unchecked
<b>Auto-Mount URI</b>	Leave it blank
<b>Package URL</b>	Leave the default
<b>Initial Version</b>	0.1d
<b>Version URL</b>	Leave the default
<b>Summary</b>	A little application to keep track of items on a "To do list"
<b>Description</b>	A little application to keep track of items on a "To do list"
<b>Primary Owner</b>	Leave the default
<b>Primary Owner URL</b>	Leave the default
<b>Secondary Owner</b>	not necessary
<b>Secondary Owner URL</b>	not necessary
<b>Vendor</b>	not necessary
<b>Vendor URL</b>	not necessary

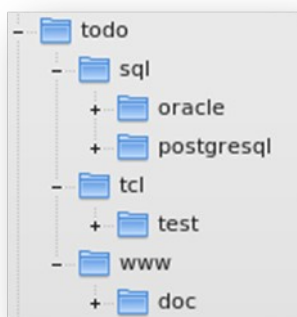


 Certifique-se de deixar marcadas a opção: **“Write a package specification file for this package”**

Agora, após você ter clicado no link **“Create Package”**, o pacote será criado e instalado no seu sistema.

É possível verificar que o pacote foi criado visitando o diretório correspondente a sua instalação OpenACS chamado “packages”, localizado em `/usr/local/aolserver/servers/openacs/`, e listar seus sub-diretórios. Você deve verificar um sub-diretório chamado “todo”, localizado em:  
`/usr/local/aolserver/servers/openacs/packages/todo/`

Dentro deste sub-diretório você encontrará um arquivo `todo.info` e a seguinte estrutura:



Esta estrutura é padrão para todos os pacotes do OpenACS. Cada arquivo e diretório tem seu propósito especial.

---

`todo.info`

---

Este é um arquivo de especificação do pacote. Dentro deste arquivo estão as informações que fazem deste diretório um pacote dentro do OpenACS. O nome, parâmetros, dependências são definidos neste arquivo. Normalmente, você não precisa modificar este arquivo diretamente, pois ele é gerado pelo gerenciador de pacote quando há alguma mudança no arquivo.

### /sql

---

Este diretório contém o modelo de dados para o pacote. As definições iniciais e mudanças no modelo de dados são definidas neste diretório.

Existem dois sub-diretórios dentro de /sql. /oracle e /postgresql. Eles permitem a definição de códigos específicos de cada banco de dados. Neste tutorial será usado somente o banco de dados postgresql.

---

### /tcl

---

O diretório TCL contém a biblioteca de arquivos com as definições de procedimentos relacionados ao pacote.

---

### /www

---

Este diretório contém as páginas públicas do pacote. Em /www são definidas as páginas que o usuário interage. Um serviço especial do OpenACS chama processador de requisições que mapeia cada requisição feita para uma instância do pacote correspondente às páginas. Ou seja, você pode ter seu pacote montado em diferentes URLs, e todos funcionando com o mesmo código.

---

## Montando o novo pacote

---

Agora que o pacote está criado, uma instância será montada no mapa do site “Site Map”. Montar um pacote significa que as páginas no diretório /www podem ser acessadas pelo usuário.

Visite a página de administração, <http://localhost:8000/acs-admin/> na seção “Subsite Administration”, clique em “Main Site”, <http://localhost:8000/admin/>

Uma vez que você encontra-se nesta página, procure a seção “Advanced Features” para a opção “Site Map”

## OpeACS.org



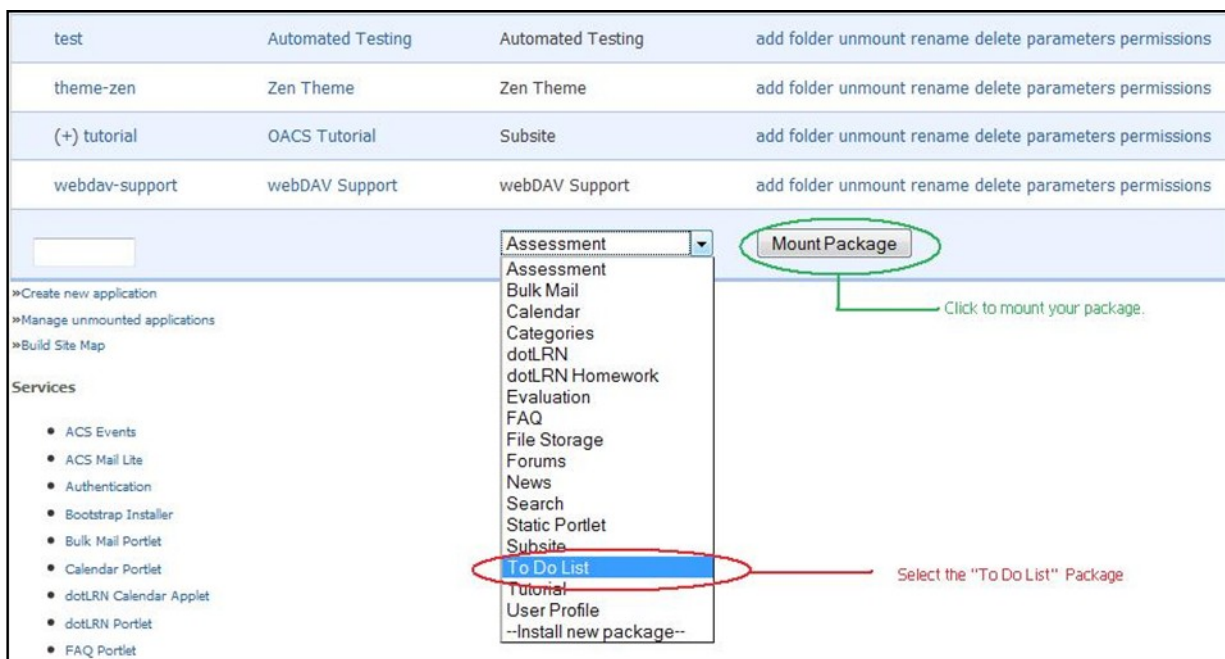
Normalmente o URL para esta página seria algo como `http://localhost:8000/admin/site-map/`

Nesta página você pode ver os “site-nodes” diferentes que existem em seu site, packages, subsite etc.

Desça até o final da página e veja um pequeno formulário que permite voce montar um pacote.

Você pode até escolher um nome qualquer que desejar no momento em que o pacote for montado.

Deixando em branco o pacote terá o nome definido pelo padrão do sistema.



Após clicar em “Mount Package”, a nova instância do pacote aparecerá na lista do mapa do site “Site Map”. Você pode até navegar por ele, porém este da figura não possui conteúdo ainda.

theme-zen	Zen Theme	Zen Theme	add folder unmount rename delete parameters permissions
todo	To Do List	To Do List	add folder unmount rename delete permissions
(+) tutorial	OACS Tutorial	Subsite	add folder unmount rename delete parameters permissions
webdav-support	webDAV Support	webDAV Support	add folder unmount rename delete parameters permissions

Assessment

Você pode criar um arquivo simples “index.adp” ou “index.html” dentro do diretório /www do novo pacote para testar como funciona.

## SESSÃO 3

NESTA SEÇÃO VOCÊ APRENDERÁ:

- ✓ **COMO DEFINIR UM MODELO DE DADOS EM OPENACS**
- ✓ **COMO EXECUTAR UM ARQUIVO SQL NA INSTALAÇÃO DO PACOTE**
- ✓ **O QUE CONTÉM O MODELO DE DADOS “TODO”**

---

### Modelo de Dados “To Do”

---

Será preciso criar pelo menos uma tabela no banco de dados para armazenar a informação da lista de tarefas “To Do”.

É preciso armazenar:

- ✓ Título da tarefa
- ✓ Descrição da tarefa

## OpeACS.org

- ✓ Data-limite
- ✓ Estado corrente do item, tarefa (pendente, completado, cancelado)
- ✓ A qual usuário-criador pertence o item
- ✓ Quando foi criada
- ✓ quando foi modificada ou atualizada
- ✓ a qual instância do pacote o item pertence

Execute o comando para criar uma tabela chamada “todo\_item” no banco de dados:

```
create table todo_item (  
  item_id integer,  
  title varchar(200),  
  description text,  
  status char(1),  
  owner_id integer,  
  due_date date default now(),  
  creation_date date default now(),  
  constraint to_do_list_pk primary key (item_id),  
  constraint to_do_owner_fk foreign key (owner_id) references users  
);
```



Então salve o comando no arquivo

`/usr/local/aolserver/servers/openacs/packages/todo/sql/postgresql/todo-create.sql`

Ao salvar o arquivo com o nome todo-create.sql, você certifica-se que sempre que instalar o pacote “todo” no sistema OpenACS. Ele criará a tabela “todo\_item” durante o processo de instalação. Este arquivo não será usado novamente. Mas será bastante útil se for preciso instalar o pacote novamente em um outro sistema.

## OpeACS.org

Se você precisa de ajuda para executar o script do arquivo, verifique o apêndice B ao final deste tutorial.



Você pode também especificar uma arquivo com instruções para serem executadas quando desejar desinstalar o pacote. Você deve salvar este comando num arquivo chamado `/usr/local/aolserver/servers/openacs/packages/todo/sql/postgresql/todo-drop.sql`

```
DROP TABLE todo_item;
```

Uma simples comando de drop table funcionará por agora.

---

### Adicionando algumas entradas de teste

---

Para testar a tabela são adicionados alguns itens no banco de dados. Você precisará saber o identificador do usuário, `user_id`. Para isso, visite a sessão “Users Administration”, página <http://localhost:8000/acs-admin/users> e procure página de detalhes do usuário, onde você encontrará o `user_id`.

Então, execute os comandos abaixo na linha comando do prompt.

```
insert into todo_item values(acs_object_id_seq.nextval, 'My first item', 'My first item
description', 'p', 568);
insert into todo_item values(acs_object_id_seq.nextval, 'My second item', 'My second item
description', 'p', 568);
insert into todo_item values(acs_object_id_seq.nextval, 'Another item', ' item description',
'p', 568);
```



**TRY  
IT!**

Em psql, execute ambos os blocos de criação de tabela e inserção dos valores de teste. Veja o apêndice B para maiores detalhes.

Objetivos: Neste terceiro dia com OpenACS, será realizada a finalização do pacote criado anteriormente. Será construída a interface com o usuário que poderá utilizar a aplicação “To Do”.

## **SESSÃO 4**

NESTA SEÇÃO VOCÊ APRENDERÁ:

- ✓ **COMO CRIAR UM FORMULÁRIO WEB COM OPENACS**
  - **COMO UTILIZAR O FORMULÁRIO AD\_FORM**  
**INSERIR, EDITAR DADOS, MOSTRAR INFORMAÇÕES DE UM ITEM**
  
- ✓ **COMO CRIAR RELATÓRIOS**
  - **COMO UTILIZAR LISTAS**
    - **MOSTRAR INFORMAÇÕES NUMA LISTA**
    - **ADICIONAR ORDENAÇÃO NUMA LISTA**
    - **ADICIONAR AÇÕES NUMA LISTA**

---

Criando páginas acessíveis ao usuário

---

A próxima etapa é a criação da interface com o usuário do pacote. É necessário criar as páginas web em que os usuário irão navegar. Lembre-se de salvar todas as páginas que você deseja que os usuários estejam aptos a acessar no diretório www do novo pacote. Isto é muito importante para o pacote funcionar corretamente.

### Página para adicionar e editar itens

---

Esta página será criada para adicionar um item. Será feita de uma forma que seja possível reutilizar a mesma página para editar o item.

É possível fazer um formulário HTML simples e cuidar das entradas do usuário manualmente, porém não há satisfação alguma neste processo. Utilizando OpenACS para fazer resolver este problema basta utilizar o procedimento pré-construído para formulário chamado `ad_form`, que permite a especificação dos campos do formulário, as ações a serem executadas em diferentes estados de um formulário, validação de entradas e etc. Para maiores informações sobre a API `ad_form` visite o link [http://www.openacs.org/api-doc/proc-view?proc=ad\\_form](http://www.openacs.org/api-doc/proc-view?proc=ad_form)

Primeiramente, vamos iniciar com a página ADP. Você pode perceber que a página é bem simples.



todo-ae.adp

---

```
<master>
<property name="doc(title)">@page_title@</property>
<formtemplate id="todo_item_form"></formtemplate>
```



todo-ae.tcl

---

Vamos começar com o bloco `ad_page_contract` e declarar algumas variáveis que usaremos posteriormente.

```
ad_page_contract {
This page allows the users to add new items to their to do list or edit existing items.
} {
item_id:optional
```



```
}  
  
set page_title "Add/Edit Todo Item"  
set user_id [ad_conn user_id]
```

Percebe-se que foi declarado `item_id` como parâmetro opcional desta página. Isto permite verificar facilmente se o formulário está em modo de edição ou adição de um novo item. Se `item_id` está presente então o modo é de edição. Se `item_id` não está presente então um novo item deve ser inserido. O formulário `ad_form` é esperto o bastante para reconhecer esses dois modos de execução. Além disso a API `ad_conn` foi utilizada para recuperar informações do usuário no sistema.

---

### FORM WIDGETS:

---

Continuando na construção do formulário, observem este trecho de código:

```
ad_form -name todo_item_form -export { user_id } -form {  
  item_id:key  
  {title:text {label "Task Title" } }  
  {description:text(textarea) {label "Description"}}  
  {due_date:date(date) {label "Due Date: " } {format {MONTH DD YYYY} } }  
  {status:text(select) {label "Status"}  
    {options { {"Pending" "p"}  
              {"Complete" "c"}  
              {"Canceled" "x" }  
            } }  
  }  
}
```

Na primeira linha foi definido o formulário `ad_form`. O parâmetro “name” atribui um nome ao formulário. Desta forma, é possível referencia-lo na página ADP ou posteriormente ao longo do mesmo arquivo TCL. Foi usado também o parâmetro “export” para exportar algumas variáveis externas ao formulário `ad_form`.

## OpeACS.org

```
ad_form -name todo_item_form -export {user_id } -form {
```

Ele pega as variáveis declaradas no arquivo TCL e coloca-as dentro do formulário ad\_form como campos escondidos.

```
item_id:key
```

Após o parâmetro “-form” são definidos os widgets (campos) do formulário. Inicialmente é definido o widget (campo) chave. Este normalmente é o identificador único do item que está sendo inserido ou editado.

Em seguida, observa-se um widget (campo) de texto simples.

```
{title:text {label "Task Title" } }
```

A próxima, define um widget (campo) do tipo “text area” para guardar a descrição do item.

```
description:text(textarea) {label "Description" }
```

O próximo é um widget (campo) de data. Este widget insere 2 “selects” e um campo de texto. Assim usuário pode escolher o mês, o dia e entrar com o ano. Este widget de data substitui a declaração de 3 widgets por apenas 1 campo de informação. É especificado também o formato desejado para usar com a data.

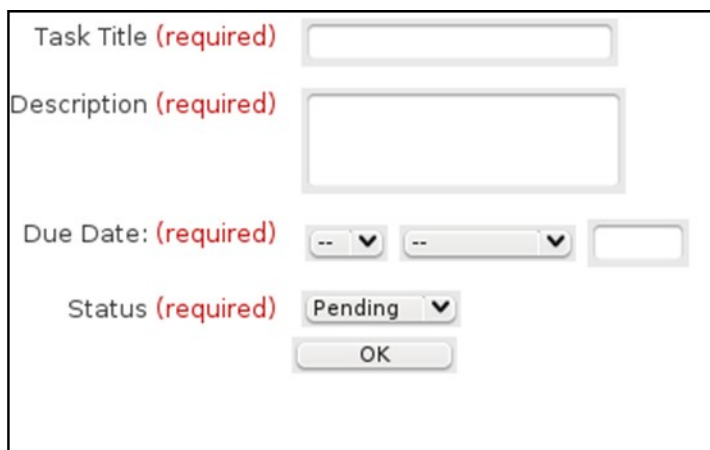
```
{due_date:date(date) {label "Due Date: " } {format {MONTH DD YYYY} } }
```

O último widget é um “select” que contém “options”, uma lista de possíveis valores e rótulos que este select mostrará ao usuário. Cada opção é uma lista de dois elementos: {<rótulos> <valor> }

## OpeACS.org

```
{status:text(select) {label "Status"}
  {options { {"Pending" "p"}
            {"Complete" "c"}
            {"Canceled" "x"}
            }}
}
```

Neste ponto, se você visitar a página no navegador, é possível observar algo como:

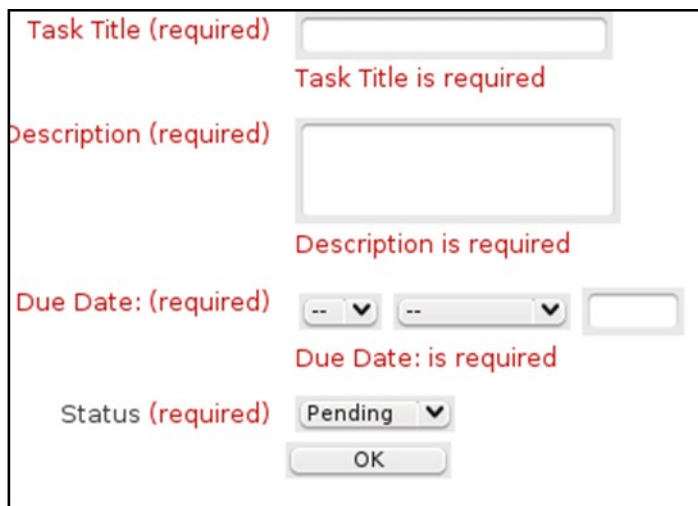


The image shows a web form with the following fields and labels:

- Task Title (required) [text input]
- Description (required) [text area]
- Due Date: (required) [date picker]
- Status (required) [dropdown menu with "Pending" selected]
- OK [button]

Você pode perceber o rótulo “required” ao lado de cada campo do formulário. Isto significa que `ad_form` está fazendo algumas básicas validações pelo programador. Se algum dos campos requeridos não possuem valor então o formulário será bloqueado e uma mensagem de erro será mostrada próximo a cada campo como mostra a figura.

## OpeACS.org



The screenshot shows a form with four fields, each with a red error message below it:

- Task Title (required)**: An empty text input field with the error message "Task Title is required".
- Description (required)**: An empty text area with the error message "Description is required".
- Due Date: (required)**: A date picker with two dropdown menus and an empty input field, with the error message "Due Date: is required".
- Status (required)**: A dropdown menu with "Pending" selected and an "OK" button below it.

Para fazer que um campo seja opcional, você apenas precisa modificar um pouco a declaração dos widgets (campos) no formulário. É mostrado abaixo um exemplo com o campo descrição

```
{description:text(textarea),optional {label "Description"}}
```



<http://localhost:8000/todo/todo-ae>

---

### Inserindo novos dados

---

A próxima seção do formulário define o que será feito com novos dados quando é inserido um novo item na lista de atividades. Existe um bloco na declaração do formulário `ad_form` chamado “new-data”.

## OpeACS.org

Este bloco contém códigos que serão executados somente quando um novo item é inserido. Normalmente são utilizados atrelados a requisições de banco de dados para armazenar as informações.

```
-new_data {
db_dml insert_item "
insert into todo_item
(item_id, title, description, status, due_date, owner_id)
values
(:item_id, :title, :description, :status, to_date(:due_date,'YYYY MM DD HH24 MI SS'),
:user_id)
"
}
```

No bloco acima, para o campo `due_date` é usada a função “`to_date`”, pois o widget “`date`” cria uma lista contendo ano, mês, dia, hora, minuto, segundos. Desta forma, temos de modificar a lista para o tipo de dado apropriado do banco de dados.

Assim, o formulário `ad_form` deve ser similar ao exemplo abaixo:

```
ad_form -name todo_item_form -export {user_id} -form {
item_id:key
{title:text {label "Task Title"} }
{description:text(textarea),optional {label "Description"}}
{due_date:date(date) {label "Due Date: "} {format {MONTH DD YYYY}} }
{status:text(select) {label "Status"}
{options { {"Pending" "p"}
{"Complete" "c"}
{"Canceled" "x"}
}}
}
} -new_data {
db_dml insert_item "
insert into todo_item
(item_id, title, description, status, due_date, owner_id)
values
(:item_id, :title, :description, :status, to_date(:due_date,'YYYY MM DD HH24 MI SS'),
:user_id)
"
}
```

É interessante realizar alguns testes de inserção de um novo item no banco de dados. É necessário executar algumas consultas no prompt ou linha de comando do psql, para verificar as informações armazenadas no banco de dados.

```
openacs=> select * from todo_item;
```

```
item_id | title | description | status | owner_id | due_date | creation_date | last_modified_date
-----+-----+-----+-----+-----+-----+-----+-----
2298 | first test | first test description | p | 568 | 2009-06-24 | 2009-04-26 | 2009-04-26
(1 row)
```

E para visualizar a página com o formulário visite:

<http://localhost:8000/todo/todo-ae>

Neste ponto, existe uma página que efetivamente armazena novos itens no banco de dados, na qual o usuário pode interagir. E ainda pode ser feito muito mais. Em seguida, é mostrado como editar um item existente. E na próxima seção você verá como mostrar as informações dos itens inseridos no banco de dados.

---

### Editando um ítem existente

---

Para editar um item existente são necessárias 2 condições: recuperar toda a informação existente do item e preencher todos os widgets ou campos do formulário `ad_form`, e atualizar o banco de dados quando o formulário for submetido.

## Recuperando os dados existentes

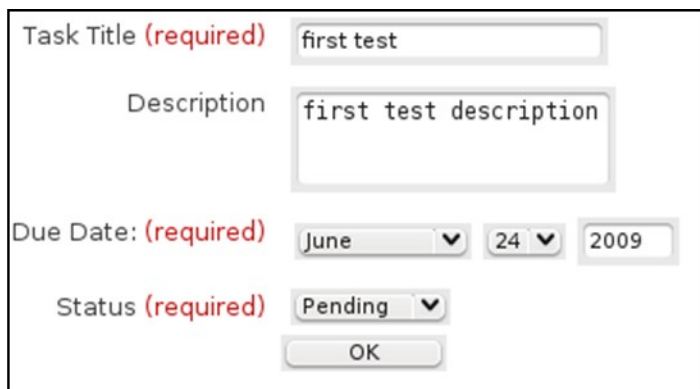
---

A informação existente é recuperada usando o bloco “select\_query”. No corpo deste bloco estão a consultas SQL que retornam linhas (rows) contendo os valores de todas as informações permitidas. Estes valores são transferidos para os widgets do formulário. Este bloco select\_query somente pode ser usado se e somente se o formulário contém o identificador chave do item.

Esta é a consulta de seleção do formulário:

```
-select_query {
  select title,
         description,
         to_char(due_date, 'YYYY MM DD') as due_date,
         status
  from   todo_item
  where  item_id = :item_id
         and owner_id = :user_id
}
```

Recupere o identificador item\_id do banco de dados (como foi feito algumas seções anteriores) e acrescente-o ao URL da página. Será então possível visualizar os campos preenchidos com os dados correspondentes ao item\_id.



The screenshot shows a web form with the following fields and values:

- Task Title (required)**: first test
- Description**: first test description
- Due Date: (required)**: June 24 2009
- Status (required)**: Pending

An "OK" button is located at the bottom of the form.

### Atualizando o ítem

---

Agora que os dados encontram-se no formulário, são possíveis editá-los. Porém é necessário salvar as alterações feitas no ítem.

```
-edit_data {
db_dml update_item "
  update todo_item
  set title = :title,
      description =:description,
      status = :status,
      due_date = to_date(:due_date,'YYYY MM DD'),
  where item_id = :item_id and owner_id = :user_id "
}
```



Lembre-se de adicionar a cláusula **WHERE** para certificar-se de que as modificações são feitas somente no ítem que deseja alterar. Neste caso, existe uma outra validação para certificar-se de que o ítem modificado pertence ao usuário correspondente.

---

### Após concluir um formulário

---

Algumas vezes, após o processamento correto do formulário, pode ser desejado realizar alguns cálculos ou simplesmente redirecionar o usuário para uma outra página. Isto pode ser feito facilmente no bloco “after\_submit”. Ele será usado agora em seguida, para redirecionar o usuário para a página inicial do pacote ou aplicação.

```
-after_submit {
ad_returnredirect index
ad_script_abort
}
```



### Modo de execução do formulário

---

Nesta seção, dois modos de execução do formulário serão descritos. É mais uma vantagem do formulário `ad_form`.

No modo de apresentação são apresentadas aos usuários as informações do item de uma forma não editável. Para é necessário apenas adicionar um parâmetro na declaração da chamada do formulário `ad_form`.

No modo de edição as informações do ítem pode ser modificadas. Os widgets estão disponíveis em modo de edição e o usuário é livre para alterar os campos de entrada.

```
ad_form -name todo_item_form -export { user_id } -mode $form_mode -form {
```

A variável “`form_mode`” foi adicionada com o valor padrão “`edit`” como parte do “`ad_page_contract`” no início do arquivo TCL. Ela pode também ter o valor “`display`” para somente apresentar os dados de forma não editável.



`todo-ae.tcl`

---

O arquivo TCL completo `todo-ae.tcl` segue abaixo.

```
ad_page_contract {  
This page allows the users to add new items to their to do list or edit existing items.  
} {  
item_id:optional  
{due_date ""}  
{form_mode "edit" }  
}
```

```
set page_title "Add/Edit Todo Item"
set user_id [ad_conn user_id]

ad_form -name todo_item_form -export {user_id} -mode $form_mode -form {
  item_id:key
  {title:text {label "Task Title"} }
  {description:text(textarea),optional {label "Description"}}
  {due_date:date(date) {label "Due Date: "} {format {MONTH DD YYYY}} }
  {status:text(select) {label "Status"}
    {options { {"Pending" "p"}
              {"Complete" "c"}
              {"Canceled" "x"}
            } }
  }
} -select_query {
  select title,
         description,
         to_char(due_date, 'YYYY MM DD') as due_date,
         status
  from   todo_item
  where  item_id = :item_id
        and owner_id = :user_id
} -new_data {
db_dml insert_item "
insert into todo_item
  (item_id, title, description, status, due_date, owner_id, package_id)
values
  (:item_id, :title, :description, :status, to_date(:due_date, 'YYYY MM DD'), :user_id)"
} -edit_data {
db_dml update_item "
update todo_item
  set title = :title,
      description = :description,
      status = :status,
      due_date = to_date(:due_date, 'YYYY MM DD'),
  where item_id = :item_id and owner_id = :user_id "
} -after_submit {
ad_returnredirect index
ad_script_abort
}
```

## Página Inicial

---

Será criada uma página inicial com uma lista de todos os itens . Todas as atividades criadas anteriormente.

Para criar a lista é utilizado o List Builder. Uma poderosa ferramenta do sistema de modelagem OpenACS. Ele permite facilmente a criação de listas com com boas funcionalidades. O principal procedimento do list builder é `template::list::create`. Muito pode ser feito com este procedimento, porém neste tutorial será mostrado somente o básico que esta API realiza.

O objetivo é criar uma página com uma tabela para mostrar as seguintes informações:

Title	Due Date	Status	Actions
The title for the item goes here.	April 28 2009	Pendi ng	View Edit Delete Mark Completed
another item	April 1 2010	Pendi ng	View Edit Delete Mark Completed
Finish the tutorial	April 27 2009	Pendi ng	View Edit Delete Mark Completed

A forma com que o list builder é utilizado é similar a utilização do formulário `ad_form` da seção anterior. É simplesmente uma chamada ao procedimento com parâmetros diferentes que afetam a saída e o comportamento da lista. Além do mais, existe um bloco de código chamado “`multirow`” associado a lista. O `multirow` será criado usando a função “`db_multirow`” proveniente da API do banco de dados. A página ADP é muito simples. Segue abaixo um exemplo:



`index.adp`

---

```
<master>  
<property name="doc(title)">@title@</property>
```

## OpeACS.org

```
<listtemplate name="todo_list"></listtemplate>
```

### O bloco `ad_page_contract`:

Até o momento o bloco `ad_page_contract` está vazio. Porém tão logo sejam criadas as funcionalidades na lista serão adicionadas uma ou duas modificações

```
ad_page_contract {  
  This page will display a list of to do items belonging to the current user.  
} {  
}
```

### Definindo variáveis:

Serão definidas algumas variáveis para serem utilizadas posteriormente. Elas são variáveis de contexto para a consulta SQL e a página ADP

```
set title "My To Do List"  
set user_id [ad_conn user_id]
```

### Definindo a lista:

Será iniciada pela declaração da lista e alguns primeiros parâmetros:

```
template::list::create -name todo_list \  
-multirow todo_list_mr \  

```

A primeira linha chama o procedimento `template::list::create`, dá um nome a lista e define o `multirow`. O nome é necessário para permitir que a página ADP chame a lista. O `multirow` é a fonte de todos os elementos da lista.

### Definindo os elementos:

Os elementos da lista serão definidos dentro do bloco “elements”. Cada elemento segue o mesmo formato básico.

```
element_name {  
  element_option option_value  
  element_option option_value
```

## OpeACS.org

```
}
```

Existem mais opções que não necessariamente são usadas. É possível ler com mais detalhes no link: [template::list::element::create on the OpenACS API](#).

Inicialmente é adicionado um elemento.

Um simples:

```
title {  
  
}
```

funcionaria, porque a ferramenta list builder reconhece como saída o valor referente ao campo de título de cada coluna retornada pela multirow, assim, não há necessidade de opções adicionais, todavia o elemento ainda pode ser melhorado.

Para adicionar o cabeçalho de uma coluna é usado a opção “label”

```
label "Task"
```

Para criar o conteúdo do elemento, neste caso o título da atividade como link para uma outra página, existe a opção “link\_url\_col”

```
link_url_col <name of a variable with the url>
```

Para adicionar opções extras no link é usado a opção “link\_html”

```
link_html {title "You can create a title text for the link here" }
```

O elemento para o título da atividade é formado:

```
title {  
  label "Task"  
  link_url_col item_url  
  link_html {title "Click to view this item details" }  
}
```

O restante dos elementos podem ser definidos de forma similar. Em seguida, será mostrada a chamada completa para o list builder construir a lista:

```
template::list::create -name todo_list \  
-multirow todo_list_mr \  
-elements {  
  title {  
    label "Task"  
    link_url_col item_url  
    link_html {title "Click to view this item details" }  
  }  
  due_date_pretty {  
    label "Due Date"  
  }  
  status_text {  
    label "Status"  
  }  
  creation_date_pretty {  
    label "Creation Date"  
  }  
  view {  
    display_template "View"  
    link_url_col item_url  
  }  
  delete {  
    display_template "Delete"  
    link_url_col delete_url  
  }  
  completed {  
    display_template "Mark Completed"  
    link_url_col completed_url  
  }  
  cancel {  
    display_template "Cancel"  
    link_url_col cancel_url  
  }  
}
```

Porém, ainda não é o fim da declaração da lista. É necessário definir a fonte de dados e o multirow. Para isto é usado o procedimento `db_multirow`.

## OpeACS.org

A fonte de dados da lista:

O procedimento **db\_multirow** cria um objeto com múltiplas linhas. Este objeto contém os resultados da consulta ao banco de dados, que será passada.

O caminho trivial para usar db\_multirow é descrito:

```
db_multirow -extend { <lista de variáveis extras que fazem parte da multirow> }
<multirow_name> <multirow statement name> {
consulta sql para recuperar os dados
} {
bloco de código TCL que será executado para cada coluna do resultado. É útil para utilizar
as variáveis para o parâmetro “extend”
}
```

O **db\_multirow** é definida da seguinte forma:

```
db_multirow -extend { item_url delete_url cancel_url completed_url status_text }
todo_list_mr todo_list_mr \
"select item_id,
    title,
    due_date,
    to_char(due_date, 'Month DD YYYY ') as due_date_pretty,
    creation_date,
    to_char(creation_date, 'Month DD YYYY ') as creation_date_pretty,
    status
from todo_item
where owner_id = :user_id"
```

O multirow foi estendido com as variáveis item\_url, delete\_url, cancel\_url, completed\_url e status\_text. Para tornar a lista mais útil será definido o bloco de código TCL opcional.

O URL do item é bem simples. Lembrando da variável definida anteriormente “form\_mode”, que determina o modo de execução do formulário de inserção ou edição do item.

A declaração da variável “form\_mode” foi feita propositalmente para agregar esta funcionalidade a página de adição e edição de itens. Agora, somente é necessário passá-la como parâmetro no url para a página de adição e edição de itens para mostrar a informação no modo de execução “display”.

## OpeACS.org

```
set form_mode display
set item_url "todo-ae?[export_vars -url { item_id form_mode }]"
```

O procedimento `export_vars` é responsável pela criação da string que contém o URL. Ela possui uma lista de variáveis que serão passadas como parâmetro do link.

O estado de um item é armazenado no banco de dados como um único carácter. Para apresentá-lo de forma legível será criada uma nova variável que contém uma melhor descrição do estado. Um bloco de código TCL é muito útil para isto.

```
switch $status {
  p {set status_text "Pending"}
  c {set status_text "Completed"}
  x {set status_text "Canceled"}
  default {set status_text "Unknown" }
}
```

Os links `delete`, `completed` and `cancel` são feitos de forma da mesma forma. Eles apontam para uma outra página que faz a real atualização e então a lista é retornada. As outras páginas serão criadas posteriormente. Somente os links para estas páginas serão criados por agora.

```
set return_url [util_get_current_url]
set delete_url "todo-delete?[export_vars -url { item_id return_url}]"

if { $status != "c" } {
  set new_status completed
  set completed_url "todo-update-item?[export_vars -url { item_id new_status
return_url}]"
}
if { $status != "x" } {
  set new_status canceled
  set cancel_url "todo-update-item?[export_vars -url { item_id new_status return_url}]"
}
```

Se tudo funcionar corretamente, é possível visualizar a lista como segue abaixo:



Task	Due Date	Status	Creation Date				
<a href="#">First Todo Item</a>	April 17 2009	Pending	April 26 2009	<a href="#">View</a>	<a href="#">Delete</a>	<a href="#">Mark Completed</a>	<a href="#">Cancel</a>
<a href="#">install test system</a>	April 26 2009	Completed	April 26 2009	<a href="#">View</a>	<a href="#">Delete</a>	<a href="#">Mark Completed</a>	<a href="#">Cancel</a>
<a href="#">Dinner</a>	April 26 2009	Pending	April 26 2009	<a href="#">View</a>	<a href="#">Delete</a>	<a href="#">Mark Completed</a>	<a href="#">Cancel</a>
<a href="#">finish up tutorial</a>	April 27 2009	Pending	April 26 2009	<a href="#">View</a>	<a href="#">Delete</a>	<a href="#">Mark Completed</a>	<a href="#">Cancel</a>



Visite: <http://localhost:8000/todo/>

Neste ponto, já existe uma lista funcional de itens associados as atividades da aplicação “To Do”. Porém, ela ainda pode ser melhorada se adicionada a funcionalidade de ordenação da lista por coluna.

### Adicionando Ordenação:

Para adicionar a habilidade de ordenação de diferente colunas, é preciso fazer algumas mudanças:

1. Adicionar uma nova variável ao bloco `ad_page_contract`.

É adicionada uma variável opcional que armazenará em qual coluna a ordenação está sendo feita. Assim o bloco `ad_page_contract` será parecido com o exemplo abaixo:

```
ad_page_contract {
  This page will display a list of to do items belonging to the current user.
} {
  orderby:optional
}
```

2. Adicionando o bloco de ordenação “`orderby`” no list builder.

Cada elemento no bloco `orderby` segue o mesmo formato: `<nome do elemento> { <cláusula orderby > }` O list builder é responsável por reconhecer quando a ordenação é ascendente ou descendente

```
-orderby {
```

```
title {orderby title}
due_date_pretty {orderby due_date}
status_text {orderby status}
creation_date_pretty {orderby creation_date}
}
```

Maiores informações sobre o bloco de ordenação no link: [orderby on the OpenACS API](#).

### 3. Adicionando a cláusula orderby na função de consulta db\_multirow.

Duas Etapas estão envolvidas:

#### 1. Recuperando a ordem por cláusula

Isto é feito com a verificação da variável “orderby” no bloco ad\_page\_contract. Se não está vazia então uma coluna está sendo ordenada. A cláusula de ordenação será armazenada numa variável para posteriormente ser utilizada na consulta SQL. Se a variável “orderby” está vazia então a lista será mostrada numa ordem padrão.

```
if {[exists_and_not_null orderby]} {
  set orderby_clause "ORDER BY [template::list::orderby_clause -name todo_list]"
} else {
  set orderby_clause "ORDER BY due_date asc"
}
```

#### 2. Adicionando ordenação na consulta SQL.

É simplesmente adicionar a cláusula de ordenação na consulta SQL. Assim a função db\_multirow é similar a mostrada no evento:

```
db_multirow -extend { item_url delete_url cancel_url completed_url status_text }
todo_list_mr todo_list_mr \
"select item_id,
  title,
  due_date,
  to_char(due_date, 'Month DD YYYY ') as due_date_pretty,
  creation_date,
  to_char(creation_date, 'Month DD YYYY ') as creation_date_pretty,
  status
from todo_item
```

## OpeACS.org

```
where owner_id = :user_id
$orderby_clause
" {
  ...
}
```

Após escrito este código é possível verificar que o cabeçalho de cada coluna é mostrado como um link e ao clicar neles, a lista será ordenada respectivamente. Ao clicar novamente no mesmo link a lista muda a ordem de ascendente para descendente e vice-versa.

<a href="#">Task</a> ↕	<a href="#">Due Date</a> ↕	<a href="#">Status</a> ↕	<a href="#">Creation Date</a> ↕
<a href="#">First Todo Item</a>	April 17 2009	Pending	April 26 2009

### Adicionando um Botão de Ação:

Numa lista os botões de ação localizam-se no topo e normalmente disponibilizam uma forma mais fácil de alcançar uma funcionalidade que afete a lista. Neste exemplo será adicionado um botão de ação para inserir um novo item na lista.

Para adicionar o item é necessário um outro bloco na declaração do list builder chamado “actions”. O formato para cada ação é bem simples, porém com muitas ações torne-se de difícil leitura. Segue o formato:

```
-actions {
  <Action label> <Action URL> <Action title text>
  <Action label> <Action URL> <Action title text>
}
```

Um exemplo de uso do bloco “actions”:

```
-actions {
  "Add New Task" "todo-ae" "Click here to add a new item to the list"
}
```

O botão de ação será mostrado no topo da lista e leva o usuário para a página todo-ae. Quando

## OpeACS.org

posicionado o cursor do mouse sobre o botão é mostrada uma mensagem de descrição da ação "click here to add a new item to the list", contida no código TCL da declaração da ação.

<a href="#">Add New Task</a>	<a href="#">Task</a> ▾	<a href="#">Due Date</a> ▾	<a href="#">Status</a> ⚡	<a href="#">Creation Date</a> ⚡			
	<a href="#">First Todo Item</a>	April 17 2009	Pending	April 26 2009	<a href="#">View</a>	<a href="#">Delete</a>	<a href="#">Mark Completed</a>
	<a href="#">install test system</a>	April 26 2009	Completed	April 26 2009	<a href="#">View</a>	<a href="#">Delete</a>	Mark Completed
	<a href="#">Dinner</a>	April 26 2009	Pending	April 26 2009	<a href="#">View</a>	<a href="#">Delete</a>	<a href="#">Mark Completed</a>
	<a href="#">finish up tutorial</a>	April 27 2009	Pending	April 26 2009	<a href="#">View</a>	<a href="#">Delete</a>	<a href="#">Mark Completed</a>

Neste ponto existe uma lista simples e funcional usando apenas algumas das ferramentas disponibilizadas pelo OpenACS. Em seguida é mostrado a versão final do arquivo TCL `index.tcl` com todas as funcionalidades construídas nesta seção.



`index.tcl`

Este é o arquivo `index.tcl` no pacote "To Do"

```
ad_page_contract {
  This page will display a list of to do items belonging to the current user.
} {
  orderby:optional
}

set page_title "My To Do List"
set user_id [ad_conn user_id]

template::list::create -name todo_list \
  -multirow todo_list_mr \
  -elements {
    title {
      label "Task"
      link_url_col item_url
      link_html {title "Click to view this item details" }
    }
    due_date_pretty {
```

```
label "Due Date"
}
status_text {
label "Status"
}
creation_date_pretty {
label "Creation Date"
}
view {
display_template "View"
link_url_col item_url
}
delete {
display_template "Delete"
link_url_col delete_url
}
completed {
display_template "Mark Completed"
link_url_col completed_url
}
cancel {
display_template "Cancel"
link_url_col cancel_url
}
} -orderby {
title {orderby title}
due_date_pretty {orderby due_date}
status_text {orderby status}
creation_date_pretty {orderby creation_date}
} -actions {
"Add New Task" "todo-ae" "Click here to add a new item to the list"
}

if {[exists_and_not_null orderby]} {
set orderby_clause "ORDER BY [template::list::orderby_clause -name todo_list]"
} else {
set orderby_clause "ORDER BY due_date asc"
}

db_multirow -extend { item_url delete_url cancel_url completed_url status_text }
todo_list_mr todo_list_mr \
"select item_id,
title,
due_date,
to_char(due_date, 'Month DD YYYY ') as due_date_pretty,
```

```
        creation_date,
        to_char(creation_date, 'Month DD YYYY ') as creation_date_pretty,
        status
    from todo_item
    where owner_id = :user_id
    $orderby_clause

" {

    set form_mode display
    set item_url "todo-ae?[export_vars -url { item_id form_mode }]"

    switch $status {
    p {set status_text "Pending"}
    c {set status_text "Completed"}
    x {set status_text "Canceled"}
    default {set status_text "Unknown" }
    }

    set return_url [util_get_current_url]
    set delete_url "todo-delete?[export_vars -url {item_id return_url}]"

    if { $status != "c" } {
        set new_status completed
        set completed_url "todo-update-item?[export_vars -url {item_id new_status
return_url}]"
    }
    if { $status != "x" } {
        set new_status canceled
        set cancel_url "todo-update-item?[export_vars -url {item_id new_status return_url}]"
    }
    }
}
```



<http://localhost:8000/todo/>

### REMOVENDO ITENS

Nesta parte do tutorial será criada uma página simples para remoção de itens. O item é removido pelo `item_id` referenciado e o usuário retorna a página anterior a remoção. É necessário

## OpeACS.org

somente o arquivo TCL para esta página.



todo-delete.tcl

```
ad_page_contract {
  This page will delete an item from the todo list package and then return the user
  to the specified return URL.
} {
  item_id
  return_url
}

set user_id [ad_conn user_id]
db_dml delete_item "delete from todo_item where item_id = :item_id and owner_id =
:user_id"
ad_returnredirect $return_url
```

### ATUALIZANDO O STATUS DO ITEM:

A página para atualizar o estado de um item funciona de forma bem similar a remoção. Apenas a operação dml é diferente desta vez.

```
ad_page_contract {
  This page will update an item's status from the todo list package and then return the user
  to the specified return URL.
} {
  item_id
  new_status
  return_url
}

set user_id [ad_conn user_id]
switch $new_status {
  "pending" {set status_code p}
  "completed" {set status_code c}
  "canceled" {set status_code x }
  default {set status_code p }
}

db_dml update_status "update todo_item
  set status = :status_code
```

```
where item_id = :item_id
and owner_id = :user_id"
```

```
ad_returnredirect $return_url
```

## SEÇÃO 5

NESTA SEÇÃO VOCÊ APRENDERÁ:

- ✓ **COMO DEFINIR PROCEDIMENTOS EM OPENACS**
- ✓ **COMO CRIAR UM ARQUIVO QUE CONTÉM OS PROCEDIMENTOS**
- ✓ **COMO FAZER COM QUE O OPENACS CARREGUE OS ARQUIVOS/PROCEDIMENTOS NA MEMÓRIA**

ADICIONANDO UMA API TCL AO PACOTE:

OpenACS não está limitado as páginas mostradas ao usuário. Em adição cada pacote pode definir seus próprios procedimentos. Lembrando o diretório /tcl contido em cada pacote. Eles serão usados neste momento.

Será criado um novo arquivo TCL chamado **todo-procs.tcl** em **/usr/share/openacs/packages/todo/tcl/** e colado o seguinte trecho de código TCL.



todo.tcl

```
ad_library {
    Procs for the To Do list package.
}
```

```
namespace eval todo {}
```

```
ad_proc -public todo::get_status_label { status } {
```

This procedure receives a status code and returns the corresponding label.

@param status is the status code received.

@return Label corresponding to the status code or Unknown if the status code is not valid

```
} {
    switch $status {
```



## OpeACS.org

```
p {
    set status_text "Pending"
}
c {
    set status_text "Completed"
}
x {
    set status_text "Canceled"
}
default {
    set status_text "Unknown"
}
}
return $status_text
}

ad_proc -public todo::get_status_code { status_text } {
This procedure returns the status code of the task by the label it has.
@param status_text is the status's label.
@return One character status code.

} {
set status_text [string tolower $status_text]
switch $status_text {
    "pending" {
        set status_code p
    }
    "completed" {
        set status_code c
    }
    "canceled" {
        set status_code x
    }
    default {
        set status_code p
    }
}
return $status_code
}
```

A primeiras linhas do arquivo TCL o definem como uma biblioteca e disponibilizam documentação com o propósito de utilizar a [ad library procedure](#). Em seguida é definido um novo “namespace” para os procedimentos. Desta forma certifica-se de que não existam conflitos entre qualquer dos procedimentos de outros pacotes que possuem o mesmo nome, mas com “namespaces”

## OpeACS.org

diferentes.

Em seguida será definido dois procedimentos diferentes para encapsular alguns códigos escritos anteriormente. É definido também um escopo como público para que qualquer outro pacote utilize os procedimentos.

Porém ainda não é possível utilizar o pacote. É necessário alertar ao gerenciador pacotes que existem um novo arquivo de funções ou procedimentos e que deve ser carregado no sistema. Visite a página do gerenciador de pacotes <http://localhost:8000/acs-admin/apm> e busque pelo pacote na lista. Em seguida clique em “reload changed”.

<a href="#">Tcl Support</a>	<a href="#">Tcl Support</a>	0.3	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>
<a href="#">search</a>	<a href="#">Search</a>	5.4.3	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>
<a href="#">todo</a>	<a href="#">To Do List</a>	0.1.d	Enabled	Locally	<a href="#">view files</a>   <a href="#">watch all files</a>   <a href="#">reload changed</a>

Assim, os procedimentos serão carregados no sistema e podem ser usados. É uma boa idéia marcar o arquivo para ser assistido, no caso de ainda continuar modificando o arquivo.

Marked the following file for reloading:

- packages/todo/tcl/todo-procs.tcl ([watch this file](#))

If you know you're going to be modifying one of the above files frequently, select the "watch this file" changed.

Agora é possível substituir o código do arquivo TCL index.tcl e o arquivo TCL todo-update-status.tcl para uma chamada ao procedimento correspondente. Encontre o lugar correto e substitua-o:



on index.tcl:

```
set status_text [todo::get_status_label $status ]
```



on todo-status-update.tcl:

```
set status_code [todo::get_status_code $new_status]
```

#### Objetivos:

Neste quarto dia de trabalho com OpenACS você aprenderá sobre o sistema ACS Objects e como seus pacotes e ítems podem fazer parte dele. Desta forma, suas aplicações utilizem todas as vantagens dos poderosos serviços e dos módulos que OpenACS apresenta. Você aprenderá como integrar ACS Objects com a sua aplicação.

### SESSÃO 6

#### NESTA SEÇÃO VOCÊ APRENDERÁ:

- ✓ **O QUE SÃO OBJETOS ACS**
- ✓ **COMO CRIAR E USAR OBJETOS ACS**
- ✓ **QUAIS OS BENEFÍCIOS EM UTILIZAR OBJETOS ACS**
- ✓ **COMO TRANSFORMAR ÍTEMS DE UMA APLICAÇÃO NUM OBJETO**
- ✓ **ADICIONAR FUNCIONALIDADES DE UM OBJETO NUM PACOTE**

### ACS OBJECTS

OpenACS oferece a opção de usar seu sistema de objetos para integrar aplicações e utilizar as vantagens dos muitos e diferentes serviços disponibilizados pelo OpenACS.

A motivação por trás da existência e uso de ACS Objects está no fato de que enquanto você desenvolve aplicações e módulos com OpenACS, você percebe que muitos deles compartilham algumas características e requisitos em comum, tais como:

- Informações de usuário relacionadas a uma determinada entrada do banco de dados.
- Relacionamentos e permissões sobre estas entradas do banco de dados

## OpeACS.org

- Algumas podem precisar de um sistema controle de versão das entradas
- Notificações
- Comentários relacionados, taxas e outros possíveis serviços
- Múltiplas Instâncias de uma aplicação para ser utilizada por diferentes grupos dentro de uma mesmo subsite

---

### Como utilizar um objeto

---

A utilização de ACS Objects é simples, porém requer que você realize algumas etapas extras enquanto define o seu modelo de dados. Você precisa:

1. Criar um novo tipo de objeto
  - Tipos de objeto são similares a classes numa linguagem de programação orientada a objetos
2. Definir uma tabela para sua aplicação
  - Esta tabela armazena todas as informações específicas de seu objeto relativas a sua aplicação
  - A chave primária desta tabela obrigatoriamente deve referenciar o identificador do objeto. A coluna `object_id` da tabela `acs_objects`. Esta é a tabela central que gerencia os ACS Objects presentes no sistema.
3. Definir procedimentos específicos da aplicação
  - Estes procedimentos podem ser funções em TCL ou funções armazenadas no banco de dados
4. Tornar os objetos visíveis a sua aplicação
  - Com poucas mudanças é possível tornar a aplicação ciente da existência dos objetos presentes no sistema

Vamos iniciar as modificações de uma aplicação para utilizar ACS Objects e integra-la da melhor forma possível com OpenACS.

### Utilizando um objeto:

---

Preparando o pacote:

Nós trabalharemos com o mesmo pacote criado nas últimas seções. Novas funcionalidades serão adicionadas e será explorada também a criação e manipulação de diferentes instâncias de uma mesma aplicação.

Uma das primeiras ações a serem tomadas é remoção da tabela principal da aplicação. Não se preocupe, a tabelas será reconstruída posteriormente. Para isso é preciso executar o comando:

```
drop table todo_item;
```

---

### Definição do modelo de dados

---

Lembrando que os arquivos referentes ao modelo de dados estão armazenados no diretório `/openacs/packages/todo/sql/postgresql/`



`todo-create.sql`

---

Este arquivo será carregado pelo sistema no momento da instalação do pacote, no caso de uma nova instalação do OpenACS. Devido ao pacote da aplicação já estar instalado no sistema, o arquivo será carregado manualmente no banco de dados.

Este arquivo tem dois propósitos:

- ✓ Criar a tabela referente ao modelo de dados da aplicação.
- ✓ Registrar o novo tipo de objeto no sistema.

A primeira seção do arquivo define a nova tabela. É possível notar que muitos dos campos definidos e utilizados anteriormente foram eliminados. E isto porque o sistema de objetos manterá o registro do identificador do pacote, usuário de criação, data de criação e etc. Além disso é possível notar que o identificador de item, `item_id`, agora aponta para a tabela `acs_objects`.

## OpeACS.org

A segunda parte é mais interessante. Uma função será criada e executada para registrar o novo tipo de objeto que irá defini-lo como o objeto que usará a tabela “todo\_item” como a tabela que armazena suas informações.

```
create table todo_item (  
  item_id integer,  
  title varchar(500),  
  description text,  
  status char(1),  
  due_date date default now(),  
  constraint todo_item_pk primary key (item_id),  
  constraint todo_item_fk foreign key (item_id) references acs_objects(object_id)  
);  
  
-- Here is the function that will register a new object type for our package, in order the  
-- parameters are:  
-- Name of the new object_type  
-- "Pretty" name of the object type, this is an human readable name for our new type  
-- Pretty plural, our pretty name in plural form  
-- Supertype, this object parent type, usually it is acs_object for a new kind of object  
-- Table name, the name that holds the data model for this object  
-- ID column, the column of our data model table primary key  
-- Package Name, our package's name  
-- The last 3 parameters are not relevant at this time  
  
create function inline_0 ()  
returns integer as '  
begin  
  PERFORM acs_object_type__create_type (  
    "todo_item",  
    "To Do Item",  
    "To Do Items",  
    "acs_object",  
    "todo_item",  
    "item_id",  
    "todo",  
    "f",  
    null,  
    null  
  );  
  
  return 0;
```

```
end;' language 'plpgsql';  
  
select inline_0 ();  
  
drop function inline_0 ();  
  
\i packages/todo/sql/postgresql/todo-package.sql
```

A última linha é uma chamada à um novo arquivo sql “todo-package.sql”, onde serão definidas algumas funções para lidar com o novo objeto e a tabela `acs_objects`

### todo-package.sql



Serão definidas duas funções para manipular os objetos. Uma para criar um novo objeto e outra para remove-los. Estas funções devem existir de acordo com o padrão OpenACS estabelecido, para o comportamento do sistema de objetos

- ✓ `todo_obj_item__new`: Esta função possui como parâmetros os campos necessários para manipular a lista de itens da aplicação. Ela utiliza alguns deles para criar um novo objeto e deixa a aplicação apenas com alguns campos específicos para serem adicionados a tabela `todo_item` que foi definida anteriormente. Nota-se que é necessário declarar na função que cria o novo objeto, qual o tipo de objeto que está sendo criado.
- ✓ `todo_obj_item__delete`: Esta função possui apenas um argumento o identificador do item que deseja-se remover. Primeiramente, ela remove os dados do modelo de dados da aplicação e então chama um outra função que remove o objeto do sistema. Nunca se deve tentar remover somente o objeto da tabela `acs_objects`, pois esta função existe para garantir que os objetos sejam removidos de forma segura.

```
CREATE OR REPLACE FUNCTION todo_item__new (integer,varchar,text,char,  
integer,date,varchar,integer)  
RETURNS integer AS '  
DECLARE  
    p_item_id ALIAS FOR $1;
```

```
p_title    ALIAS FOR $2;    -- default null
p_description  ALIAS FOR $3; -- default null
p_status    ALIAS FOR $4;  -- default null
p_creation_user ALIAS FOR $5; -- default null
p_due_date  ALIAS FOR $6;
p_creation_ip ALIAS FOR $7;  -- default null
p_context_id ALIAS FOR $8;  -- default null

v_id integer;
v_type varchar;
BEGIN

    v_type := "todo_item";

    v_id := acs_object__new(
        p_item_id,
        v_type,
        now(),
        p_creation_user,
        p_creation_ip,
        p_context_id::Integer,
        true
    );

    insert into todo_item
        (item_id, title, description, status, due_date)
    values
        (p_item_id, p_title, p_description, p_status, p_due_date);

    return v_id;

END;
' LANGUAGE 'plpgsql';

CREATE OR REPLACE FUNCTION todo_item_delete (integer)
RETURNS VOID AS '
DECLARE
    p_item_id ALIAS FOR $1;
BEGIN
    delete from todo_item where item_id = p_item_id;
    PERFORM acs_object__delete(p_item_id);
END;
' LANGUAGE 'plpgsql';
```



### Modificações em páginas da aplicação

---

O modelo de dados da aplicação foi modificado para utilizar os objetos. Mas ainda é preciso certificar-se de que a aplicação reconhece os objetos e os utiliza. Para isso, é necessário fazer algumas pequenas mudanças nas páginas visíveis ao usuário.

---

### Adicionar e Editar com objetos

---

Inicialmente as modificações serão feitas na página todo-ae. Existem três condições necessárias para certificar-se de que:

Quando adicionar novos itens, devem ser adicionados como objetos.

Quando editar um item, deve ser editado como objeto

Quando recuperada e apresentada a informação, ela deve ser apresentada como objeto

---

### Modificações na recuperação da informação

---

Quando a informação for recuperada, tanto para edição quanto apresentação dos detalhes do item, é necessário modificar a consulta do bloco “select\_query” na definição do formulário ad\_form, simplesmente reescrevendo a consulta para:

```
select todo.title,  
       todo.description,  
       to_char(todo.due_date, 'YYYY MM DD') as due_date,  
       todo.status  
from   todo_item todo,  
       acs_objects obj
```

```
where todo.item_id = :item_id  
  and obj.object_id = todo.item_id  
  and obj.creation_user = :user_id  
  and obj.context_id = :package_id
```

Como você pode ver, as modificações principais nesta parte foram para recuperar algumas informações da tabela `acs_objects`.

Um dos campos incluído na cláusula `WHERE`, `context_id`, determina em qual instância do pacote em que o item foi criado e `creation_user` o campo que determina id do usuário que criou este item.

Estes campos são utilizados para certificar-se de que estamos apresentando a informação que pertence a instância correta do pacote e ao usuário correto.

Lembrando de quando o `site-map` foi utilizado para montar o primeiro pacote. O que foi feito lá foi criar uma nova instância do pacote “`todo`”. Você pode criar quantas instâncias desejar. Por exemplo, poderia montar uma instância chamada “`todo-work`”, e outra chamada “`todo-school`”. Para manter registro das atividades do trabalho e da escola em instâncias separadas. OpenACS manipulará a lógica por trás de `cena`, criando URL, permissões, usuários e etc.

Para criar múltiplas instâncias de um mesmo pacote siga as etapas:

1. Visite a página `site-map`

## OpeACS.org

<a href="#">doc</a>	<a href="#">Documentation</a>	Documentation	<a href="#">add folder</a> <a href="#">unmount</a> <a href="#">rename</a> <a href="#">delete</a> <a href="#">permissions</a>
<a href="#">test</a>	<a href="#">Automated Testing</a>	Automated Testing	<a href="#">add folder</a> <a href="#">unmount</a> <a href="#">rename</a> <a href="#">delete</a> <a href="#">parameters</a> <a href="#">permissions</a>
<a href="#">todo</a>	<a href="#">To Do List</a>	To Do List	<a href="#">add folder</a> <a href="#">unmount</a> <a href="#">rename</a> <a href="#">delete</a> <a href="#">permissions</a>

Search

[»Create new application](#)  
[»Manage unmounted applications](#)  
[»Build Site Map](#)

2. Seguindo o mesmo procedimento feito quando montado o pacote da aplicação “todo” pela primeira vez, selecione “To Do List” do menu dropdown e escolha qual o novo nome da instância que deseja criar. Neste caso, chamamos-a de “todo-work”

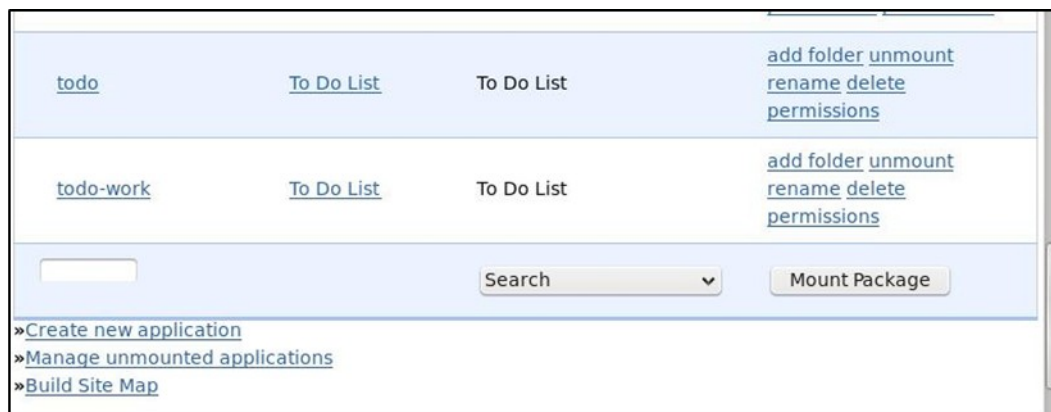
<a href="#">doc</a>	<a href="#">Documentation</a>	Documentation	<a href="#">add folder</a> <a href="#">unmount</a> <a href="#">rename</a> <a href="#">delete</a> <a href="#">permissions</a>
<a href="#">test</a>	<a href="#">Automated Testing</a>	Automated Testing	<a href="#">add folder</a> <a href="#">unmount</a> <a href="#">rename</a> <a href="#">delete</a> <a href="#">parameters</a> <a href="#">permissions</a>
<a href="#">todo</a>	<a href="#">To Do List</a>	To Do List	<a href="#">add folder</a> <a href="#">unmount</a> <a href="#">rename</a> <a href="#">delete</a> <a href="#">permissions</a>

To Do List

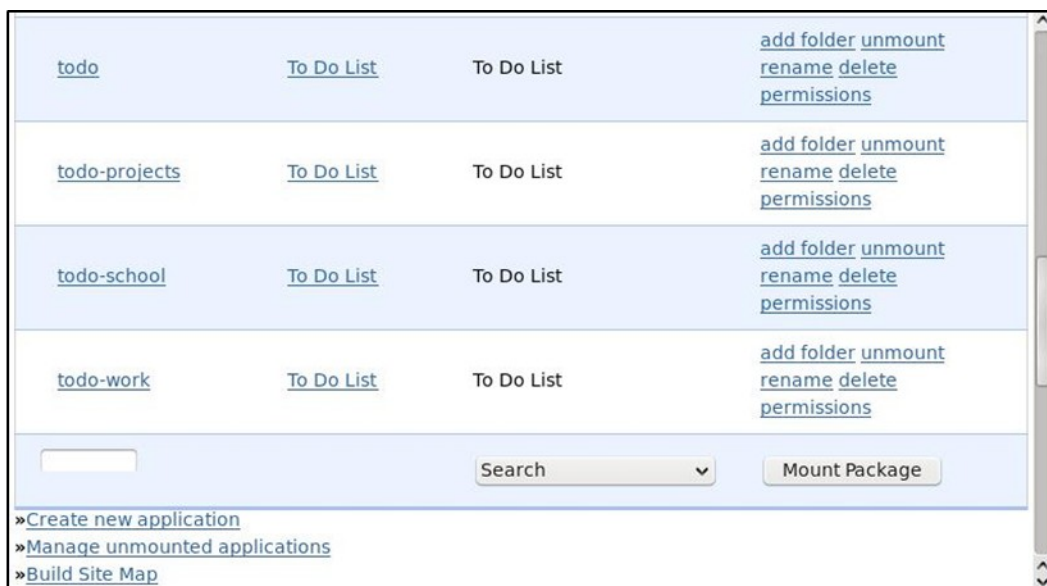
[»Create new application](#)  
[»Manage unmounted applications](#)  
[»Build Site Map](#)

## OpeACS.org

3. Agora clique em Mount Package, a página será recarregada e você terá uma nova instância da nossa aplicação “To Do List”, ou pacote montado em <http://localhost:8000/todo-work>

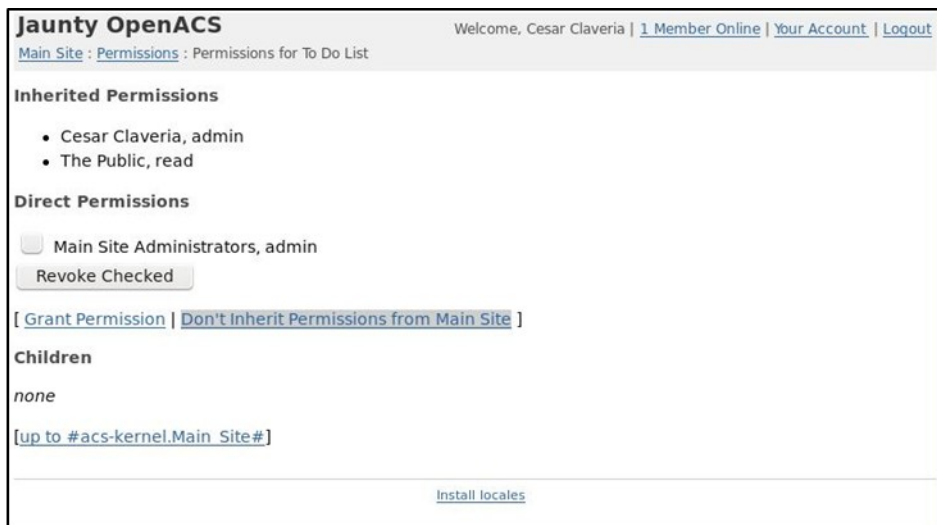


4. Você pode repetir o processo e montar quantas instâncias desejar.



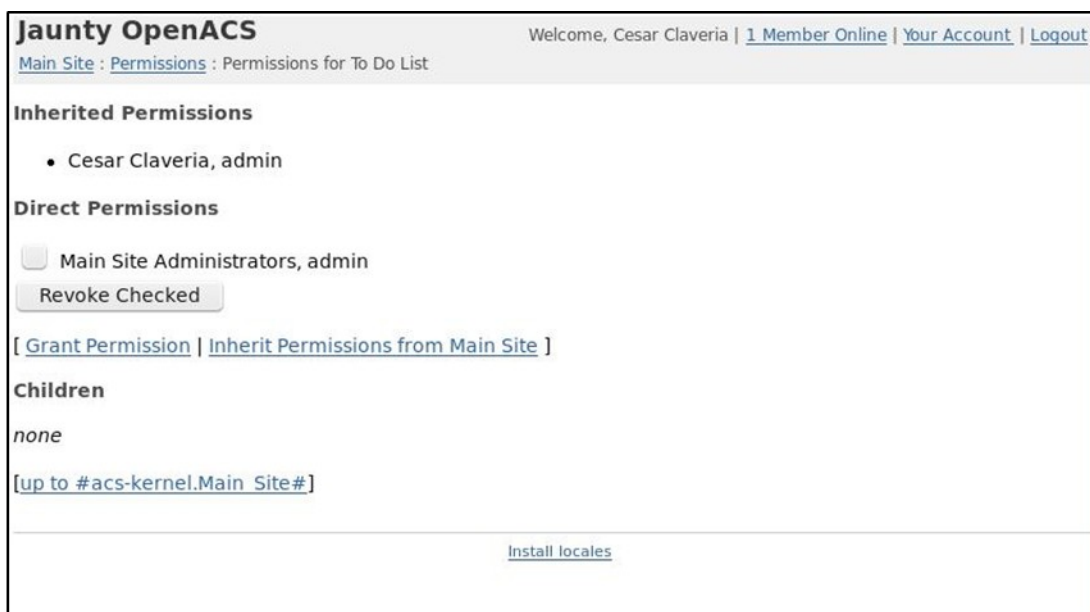
## OpeACS.org

5. Se você clica no link “Permissões” para alguma instância, você pode visualizar e gerenciar qual usuário e qual privilégio possui sobre esta instância em particular.



The screenshot shows the 'Permissions' page for a Jaunty OpenACS instance. At the top, it says 'Jaunty OpenACS' and 'Welcome, Cesar Claveria | 1 Member Online | Your Account | Logout'. Below that, it says 'Main Site : Permissions : Permissions for To Do List'. The page is divided into sections: 'Inherited Permissions' with a list of 'Cesar Claveria, admin' and 'The Public, read'; 'Direct Permissions' with a checkbox for 'Main Site Administrators, admin' and a 'Revoke Checked' button; and 'Children' with the text 'none'. At the bottom, there is a link for 'Install locales'.

6. Clique em “Não herde Permissões do site principal” para remover as permissões automáticas que a instância obteve do site principal. Neste caso, irá remover todas as permissões públicas de leitura desta instância.



This screenshot is similar to the previous one, but the 'Direct Permissions' section now has a checked checkbox for 'Main Site Administrators, admin'. The link below it has changed from 'Don't Inherit Permissions from Main Site' to 'Inherit Permissions from Main Site'. The rest of the page content remains the same.

### Modificações ao adicionar dados

---

A principal modificação será na substituição da inserção sql por um procedimento que cria um novo objeto do tipo “todo\_item”. Todas estas modificações localizam-se no bloco “new-data” na definição do formulário ad\_form.

Como pode ser visto, foi definida uma variável context\_id, para tornar a leitura da chamada da função mais clara quanto aos parametros que são passados como argumentos. O bloco de inserção foi substituído por uma chamada de função usando “db\_exec\_plsql”. Esta é uma chamada especial de funções do banco de dados.

```
-new_data {
  set context_id [ad_conn package_id]
  set new_object_id [ db_exec_plsql do_insert {
    select todo_item__new (
      :item_id,
      :title,
      :description,
      :status,
      :user_id,
      to_date(:due_date, 'YYYY MM DD HH24 MI SS'),
      :ip_address,
      :context_id
    );
  } ]
}
```

---

### Modificações ao editar dados

---

As modificações necessárias na seção de edição são principalmente apenas para lidar com a informação relacionada ao objeto e para mater registro de quando este objeto foi modificado e por qual usuário. Estas mudanças estão no bloco “edit\_data” na definição do formulário ad\_form.

```
-edit_data {
  # update the information on our table
  db_dml todo_item_update "
  update todo_item
```

```
set title= :title,
    description = :description,
status = :status,
    due_date = to_date(:due_date, 'YYYY MM DD HH24 MI SS')
where item_id = :item_id"

# update the last modified information on the object
db_exec_plsql to_do_list_obj_item_object_update {
    select acs_object__update_last_modified(:item_id,:user_id,:ip_address)
}
}
```

Para editar as informações de um ítem é apenas definido um bloco de atualização do banco de dados. Para o objeto é usado uma chamada de função `acs_object__update_last_modified` que registra quando a última mudança foi feita, com qual endereço IP e por qual usuário.



Arquivo `todo-ae.tcl` completo

```
ad_page_contract {
This page allows the users to add new items to their to do list or edit existing items.
} {
item_id:optional
{form_mode "edit" }
}

set page_title "Add/Edit Todo Item"
set user_id [ad_conn user_id]
set package_id [ad_conn package_id]
set ip_address [ad_conn peeraddr]

ad_form -name todo_item_form -export {user_id package_id} -mode $form_mode -form {
item_id:key
{title:text {label "Task Title" } }
{description:text(textarea),optional {label "Description"}}
{due_date:date(date) {label "Due Date: "} {format {MONTH DD YYYY} } }
{status:text(select) {label "Status"}
    {options { {"Pending" "p"}
              {"Complete" "c"}
              {"Canceled" "x"}
            }
    }
}
```

```
        } }
    }
} -select_query {
    select todo.title,
           todo.description,
           to_char(todo.due_date, 'YYYY MM DD') as due_date,
           todo.status
    from   todo_item todo,
           acs_objects obj
    where  todo.item_id = :item_id
           and obj.object_id = todo.item_id
           and obj.creation_user = :user_id
           and obj.context_id = :package_id
} -new_data {
    set context_id [ad_conn package_id]

    set new_object_id [ db_exec_plsql do_insert {
        select todo_item__new (
            :item_id,
            :title,
            :description,
            :status,
            :user_id,
            to_date(:due_date, 'YYYY MM DD HH24 MI SS'),
            :ip_address,
            :context_id
        );
    } ]
} -edit_data {

# update the information on our table
db_dml todo_item_update "
    update todo_item
    set title= :title,
        description = :description,
        status = :status,
        due_date = to_date(:due_date, 'YYYY MM DD HH24 MI SS')
    where item_id = :item_id"

# update the last modified information on the object
db_exec_plsql to_do_list_obj_item_object_update {
    select acs_object__update_last_modified(:item_id,:user_id,:ip_address)
}
}
```



## OpeACS.org

```
} -after_submit {  
  ad_returnredirect index  
  ad_script_abort  
}
```

Até este ponto o arquivo ADP não precisa de modificação

---

### Página índice com objetos

---

A página índice que apresenta somente uma lista de todos os ítems e permite adicionar novos, deve também ser modificada um pouco para funcionar com o novo modelo da dados.

---

### Modificando o bloco lista -multirow

---

A primeira mudança será na consulta sql do multirow. Será necessário adicionar algumas informações da tabela de objetos. Modifique a consulta do multirow para o seguinte:

```
select todo.item_id,  
  todo.title,  
  todo.due_date,  
  to_char(todo.due_date, 'Month DD YYYY ') as due_date_pretty,  
  obj.creation_date,  
  to_char(obj.creation_date, 'Month DD YYYY ') as creation_date_pretty,  
  todo.status  
from todo_item todo, acs_objects obj  
where obj.context_id = :package_id  
  and obj.creation_user = :user_id  
  and obj.object_id = todo.item_id  
$orderby_clause
```

Atenção para como a data de criação “creation\_date” é recuperada do banco de dados usando o identificador do objeto “object\_id”, certificando-se de que o contexto dos objetos apretnados estão dentro da página corrente.

### Modificando o bloco de ordenação -order\_by

---

São necessários alguns ajustes no bloco `order_by`, nada muito relevante, simplesmente certificar-se de que a lista está sendo ordenada pelos campos corretos.

```
-orderby {  
  title {orderby todo.title}  
  due_date_pretty {orderby todo.due_date}  
  status_text {orderby todo.status}  
  creation_date_pretty {orderby obj.creation_date}  
}
```

Estas são todas as modificações necessárias nas páginas visíveis ao usuário para funcionarem com o sistema de objetos do OpenACS. Certamente agora podemos reter a mesma funcionalidade.



<http://localhost:8000/todo/>

Na próxima seção, mais vantagens utilizando OpenACS serão exploradas.

## SESSÃO 7

NESTA SEÇÃO VOCÊ APRENDERÁ:

- ✓ **COMO INTEGRAR OS OBJETOS COM OUTROS SERVIÇOS DISPONÍVEIS EM OPENACS**
- ✓ **COMO ADICIONAR O SERVIÇO “COMMENTS” NA APLICAÇÃO PARA OS ITENS E OUTROS OBJETOS**
- ✓ **COMO UTILIZAR A API DE PERMISSÕES**

## OpeACS.org

### INTEGRANDO COM OUTROS SERVIÇOS

Um dos mais benefícios mais relevantes na utilização dos objetos ACS é a possibilidade de integrar uma aplicação com alguns serviços disponíveis no OpenACS. Será demonstrado como fazer com alguns objetos.

#### COMMENTS:

Pretende-se adicionar a habilidade de armazenar comentários de uma atividade. É possível simplesmente a criação de um modelo de dados para armazenar os comentários. Porém para usufruir dos benefícios do OpenACS basta utilizar todos os módulos e aplicações que já foram desenvolvidas e testadas com ele. Para a funcionalidade de comentários será utilizado o serviço chamado "General Comments", que permite juntar qualquer objeto ACS com comentários e escrevendo apenas algumas poucas linhas de código TCL

Mas, primeiramente é preciso verificar se o pacote "General Comments" está instalado no sistema. Para isto, visite a página <http://localhost:8000/acs-admin/install/> e em "Installed Packages" busque pelo nome General Comments.

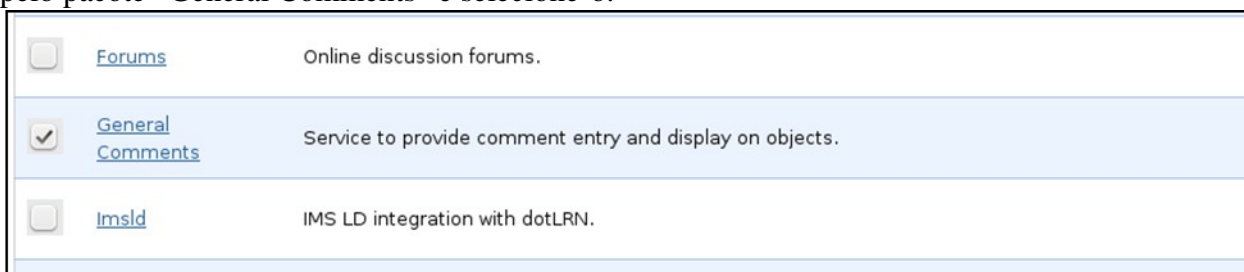


Installed Packages	
Type [ <b>Application</b>   <a href="#">Service</a> ]	
Package	Version
General Comments	5.2.0
Lars Blogger	2.4.1
Notifications	5.4.3
Search	5.4.3
To Do List	0.1d
To Do List Object	0.1d

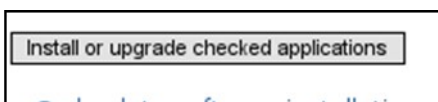
Se o pacote está instalado, continue para a próxima seção "Adicionando Comentários aos itens", se não, na próxima etapa mostra como instalar o pacote no sistema.

### INSTALANDO GENERAL COMMENTS:

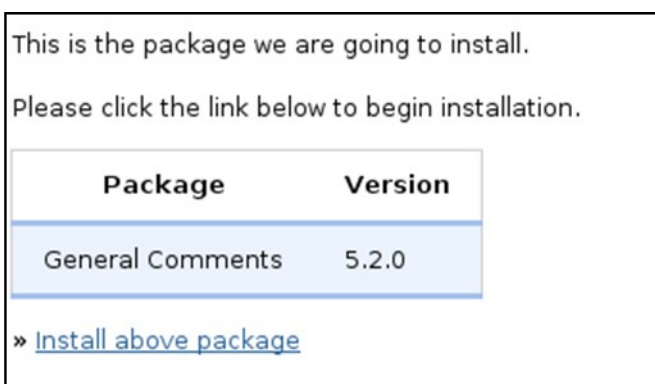
1. Na mesma página, clique em "[Install from Repository](#)"
2. Deve aparecer uma lista com todos os pacotes disponíveis para download e instalação. Procure pelo pacote "General Comments" e selecione-o.



3. Agora clique no botão "Install or Upgrade Checked Applications" próximo ao rodapé da página.

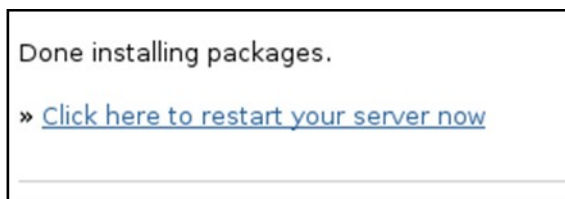


4. Uma página de confirmação será mostrada, clique em "Install Above Package" para iniciar o processo de instalação.



5. Aguarde enquanto os pacotes estão sendo instalados e na página de confirmação clique no link para reiniciar o servidor.

## OpeACS.org



6. Dependendo de como seu sistema está configurado será necessário reiniciar o serviço manualmente.

✓ No Ubuntu execute o comando na linha de comando do prompt:

➤ `sudo /etc/init.d/openacs restart`

✓ No Windows:

➤ Vá em Menu Iniciar

➤ Busque pelo menu de programas “win32-OpenACS”

➤ Clique em “Stop OpenACS” e então

➤ Clique em “Start OpenACS”

### ADICIONANDO COMENTÁRIOS AOS ITEMS:

Os comentários serão mostrados quando for visualizado somente um ítem no modo de apresentação “display”. Os comentários serão omitidos no modo de edição “Edit”. É necessário adicionar um link para inserir comentários e mostrar seus conteúdos. Adicione o seguinte bloco de código após a declaração do formulário `ad_form` no arquivo TCL “**todo-ae.tcl**”.

```
if { [string equal $form_mode "display"] } {  
  
    set comment_add_url "[general_comments_package_url]comment-add?[export_vars {  
    { object_id $item_id }  
    { return_url [util_get_current_url]}  
    }]"  
  
    set comments_html [general_comments_get_comments -print_content_p 1 $item_id  
[util_get_current_url]]  
}
```

### Ao analisar o código temos:

- ✓ Verifica-se que o código executa somente no modo de apresentação “display”. Assim, os comentários estão disponíveis somente ao mostrar um item somente
- ✓ Em seguida, será construído o link “Add comment”, percebe-se:
  - É usado o procedimento `[general_comments_package_url]` para recuperar o URL para o pacote “general comments”, código TCL manuais não são uma boa idéia num sistema tão flexível quanto OpenACS
  - O uso de `export_vars` para passar algumas variáveis pelo link, estas variáveis são os identificadores `item_id` (`object_id`) e `return_url`. Assim é possível retornar para a página após o comentário ser criado.
- ✓ Finalmente todos os comentários deste item foram recuperados e estão prontos para serem mostrados na página ADP usando o procedimento `general_comments_get_comments`.
  - No arquivo ADP "`todo-ae.adp`" deve ser escrito o seguinte código TCL.

```
<if @form_mode@ eq "display">
<a href="@comment_add_url@">Add a comment</a>
<p>
@comments_html;noquote@
</if>
```

Verifica-se o modo de apresentação corrente é “display”. Em caso positivo então é construído o link “Add Comment” e serão mostrados todos os itens relacionados ao objeto.

É perceptível no código acima como é recuperado o conteúdo da variável `comments_html`, `@comments_html;noquote@`. O uso do modificador `noquote` é para prevenir OpenACS de limpar e mostrar a marcação HTML ao invés de deixar passar pelo navegador e mostrar como intencionado. Caso o modificador não seja usado, será mostrado algo como:

```
<h4>My Comment</h4> Yes this is my comment!
```

Ao invés da seção de comentário pré formatada:

Teste o exemplo no sistema. O resultado deve ser similar a imagem seguinte



Task Title	Pick up Milk
Description	Remember the milk on my way home
Due Date:	June 10 2009
Status	Pending

[Add a comment](#)

**Also sugar**

Remember to also buy sugar.

-- [cesar claveria](#) on May 01, 2009 04:45 PM ([view details](#))

### PERMISSÕES:

Este provavelmente é um dos maiores benefícios obtidos ao utilizar Objetos ACS. Fácil integração com o sistema de permissões do OpenACS. Será mostrado um pequeno caso de teste para analisar o sistema de permissões em ação.

Primeiramente, **adicione um novo usuário ao sistema**. Isto será feito na página:

<http://localhost:8000/acs-admin/users/user-add>, preenchendo o formulário de novo usuário e ao clicar no botão para confirmar o registro um novo usuário será criado.

O código TCL será modificado para que somente o usuário que criar cada ítem seja permitido de ler ou modificá-los

No arquivo **todo-ae.tcl** será modificado o bloco "**new\_data**" sdo formulário ad\_form. Adicione este código após o objeto ser criado, porém antes de fechar o bloco new\_data.

```
#The first command on this last block of code will remove all the permissions the object
has inherited from its "ancestors" (the package, the site, etc), the next command grants the
"admin" permission to the current user and finally, we allow the current user to be able to
add comments to this object.
```

```
#Remove all permissions from the object.
permission::toggle_inherit -object_id $new_object_id
```

## OpeACS.org

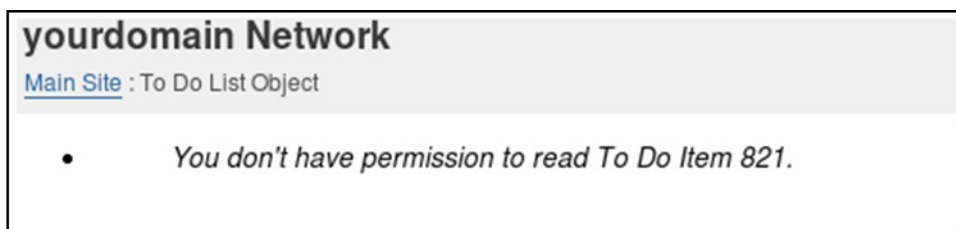
```
#Grant the permission to add comments to this user, on this object.  
#Administrators will have this permission by default, but, we are making sure non-admin  
users also get the correct permissions.  
permission::grant -party_id $user_id \  
    -object_id $new_object_id \  
    -privilege "general_comments_create"
```

Agora, adiciona-se as verificações de permissões no topo da página, logo após `ad_page_contract`.

```
if { [exists_and_not_null item_id] && [acs_object::object_p -id $item_id] } {  
  if {[string equal $form_mode "edit"]} {  
    permission::require_permission -object_id $item_id -privilege write  
  } else {  
    permission::require_permission -object_id $item_id -privilege read  
  }  
}
```

Isto requer que o usuário tenha as permissões específicas antes de tentar realizar alguma ação. Se o formulário está em modo de apresentação “display” então a permissão de leitura é exigida. Se o formulário estão no modo de edição “edit” então a permissão de escrita é exigida. O sistema de permissão abre um mundo de novas possibilidades.

No teste destas funcionalidades, para ler ou editar os objetos com usuários diferentes, fica evidente que o usuário está tentando ler algo que não possui permissão. O sistema então detecta, pára e retorna uma mensagem de erro:



### FINALIZANDO A TRANSIÇÃO:

Ainda existem algumas páginas que não foram atualizadas para utilizar os novos objetos: a páginas de remoção e atualização de estado de um ítem. Verique abaixo no exemplo como elas ficam usando os



## OpeACS.org

objetos ACS:



todo-delete.tcl

```
ad_page_contract {
  This page will delete, an item from the todo list package and then return the user
  to the specified return URL.
} {
  item_id
  return_url
}

permission::require_permission -object_id $item_id -privilege delete
set user_id [ad_conn user_id]
db_transaction {
  db_exec_plsql delete_item { select todo_item__delete ( :item_id ) }
}

ad_returnredirect $return_url
```

Lembrando que o bloco de remoção foi substituído por uma chamada ao procedimento “todo\_item\_delete”. Desta forma fica garantido que ambos, o ítem e o objeto foram removidos corretamente. Está sendo verificado também se o usuário tem as permissões corretas para remover o ítem.



todo-update-status.tcl

```
ad_page_contract {
  This page will update an item's status from the todo list package and then return the user
  to the specified return url.
} {
  item_id
  new_status
  return_url
}

permission::require_permission -object_id $item_id -privilege write

set user_id [ad_conn user_id]
set status_code [todo::get_status_code $new_status]
db_transaction {
```

## OpeACS.org

```
db_dml update_status "update todo_item
    set status = :status_code
    where item_id = :item_id"
}

ad_returnredirect $return_url
```

Esta página não possui muitas mudanças. Foi adicionado apenas a verificação de que o usuário tem as permissões apropriadas para executar esta operação. As modificações nesta página foram postergadas para obter as vantagens da verificação das permissões do usuário.

E isto é tudo sobre objetos ACS até o momento. Objetos ACS são uma ferramenta rica e a integração com os outros serviços do OpenACS permite agregar uma grande quantidade de funcionalidades de forma fácil e organizada.

### CONCLUSÕES

Este rápido tutorial sobre OpenACS ainda é necessário para áreas importantes como xql, bibliotecas, exemplos avançados de formulários “ad\_form” e listas “list builder”. Além do uso do repositório “content repository” e exemplos avançados de objetos ACS. Assim, mais pessoas estarão interessadas no desenvolvimento de novos e rápidos tutoriais para OpenACS e suas características.

Este tutorial foi produzido e organizado por Cesar Clavería & Rocael Hernández. A maioria dos textos foram escritos por Cesar Clavería, revisado por Rocael Hernández

e muitas outras pessoas ajudaram a revisar e melhorá-lo através de testes com muitos iniciantes em OpenACS.

Este trabalho foi feito para ajudar ao .LRN Consortium.

***Início em OpenACS: guia rápido  
para uma poderosa ferramenta.***

***César Clavería - Rocael Hernández***

***2009***

### OPENACS EM WINDOWS:

Agradecimentos a flexibilidade do OpenACS e o software que foi contruído tornando possível que o sistema OpenACS funcione em ambientes diferentes, incluindo Microsoft Windows ©. No momento em que este tutorial foi escrito o instalador do windows funcionava para Windows XP and Vista, ambos versão 32 bits.

Em seguida, existem uma tabela com detalhes de diretórios diferentes discutidos nestes tutorial. Estes caminhos de diretórios são válidados para a utlização do instalador do windows.

Directory	Ubuntu	Windows
<b>Main OpenACS installation</b>	/usr/share/openacs	C:\aolserver\servers\openacs
<b>OpenACS packages directory</b>	/usr/share/openacs/packages	C:\aolserver\servers\openacs\packages
<b>Our first “To Do” package</b>	/usr/share/openacs/packages/todo	C:\aolserver\servers\openacs\packages\todo
<b>OpenACS configuration file</b>	/etc/openacs/config.tcl	C:\aolserver\servers\openacs\etc\config.tcl
<b>OpenACS log file</b>	/usr/share/openacs/log/error.log	C:\aolserver\servers\openacs\log\error.log

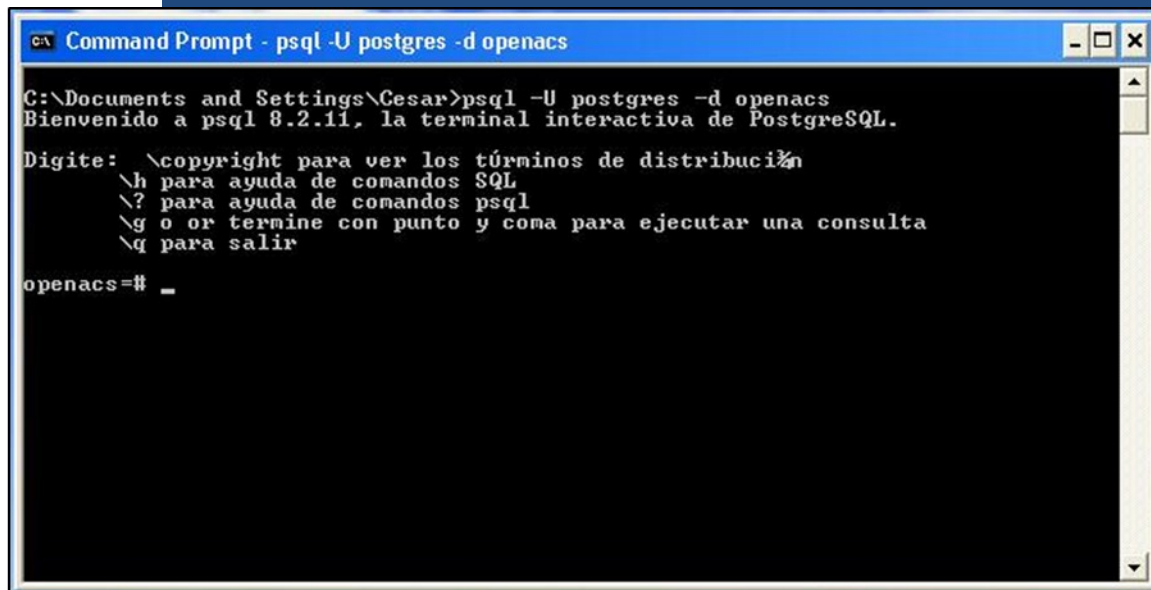
## ANEXO B

### CONECTANDO AO POSTGRESQL NO WINDOWS E UBUNTU:

Existem muitas formas para conetar o banco de dados OpenACS em ambos Windows e Linux. A seguir, será descrito como conectar ao banco de dados utilizando comandos psql prontos no prompt postgresql instalado

### CONECTANDO NO WINDOWS:

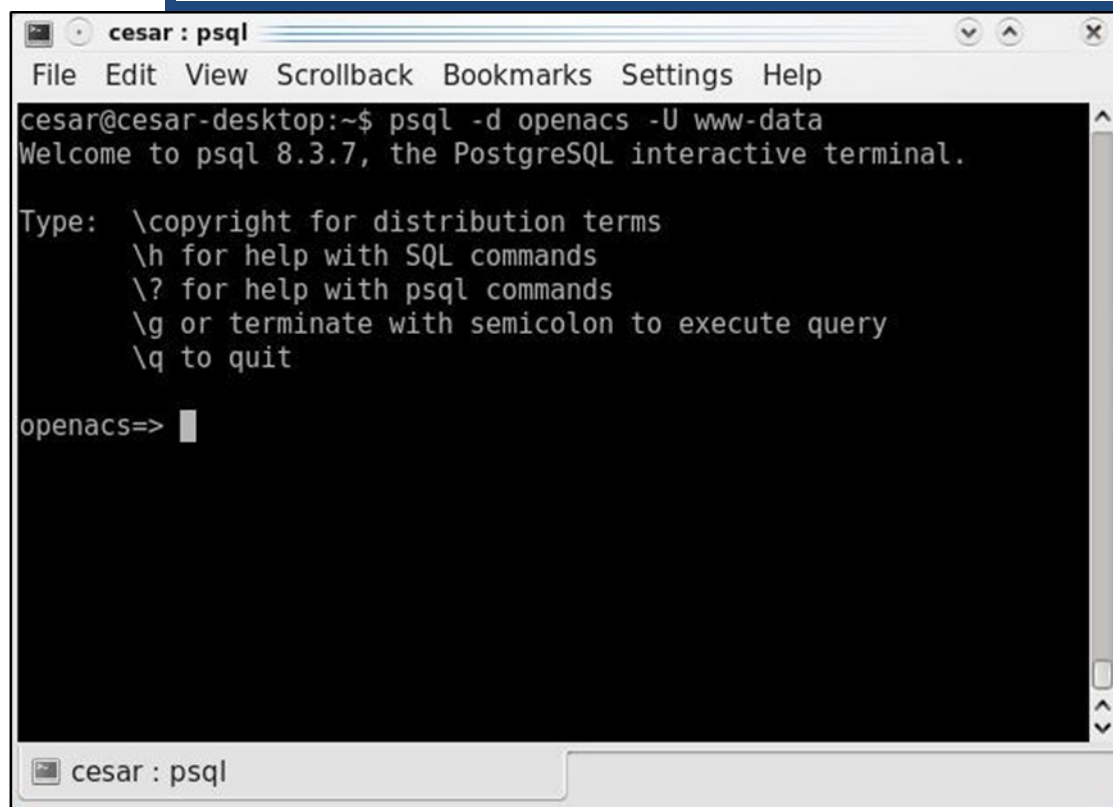
1. Open a command prompt window.
  - ✓ Go to, Start → Run → cmd.exe and press enter.
2. Connect as the postgres user by entering the command:
  - ✓ **psql -U postgres -d openacs**
    - If you are asked for a password the default password for the postgres user when using the win32 installer is "qwe123"
3. Now you can perform any sql command on this console.



```
Command Prompt - psql -U postgres -d openacs
C:\Documents and Settings\Cesar>psql -U postgres -d openacs
Bienvenido a psql 8.2.11, la terminal interactiva de PostgreSQL.
Digite: \copyright para ver los t rminos de distribuci n
        \h para ayuda de comandos SQL
        \? para ayuda de comandos psql
        \g o or termine con punto y coma para ejecutar una consulta
        \q para salir
openacs=# _
```

CONNECTING ON UBUNTU:

1. Open up a terminal application
  1. On Gnome: Applications → Accessories → Terminal
  2. On KDE: K Menu → Applications → System → Terminal
2. Enter the command:
  1. **psql -d openacs -U www-data**
3. You should be able to enter any sql command.



```
cesar : psql
File Edit View Scrollback Bookmarks Settings Help
cesar@cesar-desktop:~$ psql -d openacs -U www-data
Welcome to psql 8.3.7, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit

openacs=> █
```

#### SCRIPT DE INSTALAÇÃO DO OPENACS:

Basta fazer o download de arquivo compactado com o script para instalação do openACS em Debian ou utilizar os seis arquivos abaixo contendo todo o script de instalação:

---

#### Arquivo 1

---

```
#!/bin/sh
set -u
```

```
# Full install of pg8.2, aolserver4.5, varoious components and lates OpenACS from cvs
cd /home/start/oacs-installation-scripts/
```

#### #1. install pg

```
echo "..... running postgres installation script ....."
./1-pg82
echo "..... DONE running postgres installation script ....."
```

#### #2. run pg db installation script

```
echo "..... running setup pg db script ....."
./2-pg_db_install
echo "..... DONE running pg db script ....."
```

#### #3. install aolserver

```
echo "..... running aolserver install script ....."
./3-aolserver
echo "..... DONE running aolserver install script ....."
```

#### #4. install openacs

```
echo "..... running openacs install script ....."
./4-OACS
echo "..... DONE running openacs install script ....."
```

### Arquivo 2

---

```
#!/bin/sh
```

```
# Set the UMASK so permissions are more open
```

```
umask 022
```

```
# Path to postgres installation
```

```
export PG_INSTALL=/usr/local/pg82
```

```
export PG_VERSION=8.2.14
```

```
# Install needed binaries
```

```
apt-get update
```

```
apt-get install update-inetd bzip2 patch cvs build-essential bin86 zlib1g-dev libreadline5-dev make gcc flex bison
```

```
# Download Postgres
```

```
cd /usr/local/src/
```

```
wget http://wwwmaster.postgresql.org/redis/378/f/source/v8.2.14/postgresql-8.2.14.tar.gz
```

```
tar -xzf postgresql-${PG_VERSION}.tar.gz
```



## OpeACS.org

```
# Prepare the groups and the postgres user.
```

```
# No need to do it if you already have postgres installed
```

```
groupadd web
```

```
useradd -g web -m -d ${PG_INSTALL} postgres
```

```
chown -R postgres.web ${PG_INSTALL}
```

```
chown -R postgres.web /usr/local/src/postgresql-${PG_VERSION}
```

```
chmod 750 ${PG_INSTALL}
```

```
# Add the path for the postgres user
```

```
echo "export PATH=${PG_INSTALL}/bin/${PATH}" >> ${PG_INSTALL}/.profile
```

```
echo "export LD_LIBRARY_PATH=${PG_INSTALL}/lib:${LD_LIBRARY_PATH}" >> ${PG_INSTALL}/.profile
```

```
# And install it into the defined directory
```

```
su - postgres -c "cd /usr/local/src/postgresql-${PG_VERSION} && ./configure --prefix=${PG_INSTALL}"
```

```
su - postgres -c "cd /usr/local/src/postgresql-${PG_VERSION} && make all"
```

## OpeACS.org

```
su - postgres -c "cd /usr/local/src/postgresql- $\{PG\_VERSION\}$  && make install"
```

```
# Link the postgres installation
```

```
rm -f /usr/local/pgsql
```

```
ln -s  $\$PG\_INSTALL$  /usr/local/pgsql
```

```
# make sure postgres starts automatically
```

```
cvs -d :pserver:anonymous@cvs.openacs.org:/cvsroot co openacs-4/packages/acs-core-docs/www/files/postgresql.txt
```

```
cp openacs-4/packages/acs-core-docs/www/files/postgresql.txt /etc/init.d/pg82
```

```
chown root.root /etc/init.d/pg82
```

```
chmod 755 /etc/init.d/pg82
```

```
update-rc.d pg82 defaults
```

```
rm -r openacs-4/
```

```
# Link it up to be "backwards compatible"
```

```
rm -f /etc/init.d/postgresql
```

```
ln -s /etc/init.d/pg82 /etc/init.d/postgresql
```

---

Arquivo 3

---

```
#!/bin/sh
```

## OpeACS.org

# Path to postgres installation

```
export PG_INSTALL=/usr/local/pg82
```

```
export PG_VERSION=8.2.14
```

# Fire up the database

```
su - postgres -c "${PG_INSTALL}/bin/initdb -D ${PG_INSTALL}/data"
```

```
su - postgres -c "${PG_INSTALL}/bin/pg_ctl -D ${PG_INSTALL}/data -l ${PG_INSTALL}/data/server.log start"
```

```
su - postgres -c "sleep 4"
```

# Create plpgsql language

```
su - postgres -c "createlang plpgsql template1"
```

```
su - postgres -c "createlang -l template1"
```

```
# su - postgres -c "${PG_INSTALL}/bin/createlang plpgsql template1"
```

```
# su - postgres -c "${PG_INSTALL}/bin/createlang -l template1"
```

# Install tsearch2

```
su - postgres -c "cd /usr/local/src/postgresql-${PG_VERSION}/contrib/tsearch2 && make install"
```

```
su - postgres -c "psql -U postgres -f /usr/local/src/postgresql-${PG_VERSION}/contrib/tsearch2/tsearch2.sql template1"
```

# Install ltree

## OpeACS.org

```
su - postgres -c "cd /usr/local/src/postgresql- $\{PG\_VERSION\}$ /contrib/ltree && make install"
```

```
su - postgres -c "psql -U postgres -f /usr/local/src/postgresql- $\{PG\_VERSION\}$ /contrib/ltree/ltree.sql  
template1"
```

```
# configure pg via pg_hba.conf and postgres.conf
```

```
# - pg_hba.conf: host all all 192.168.168.199 255.255.255.255 trust
```

```
# - postgres.conf: see 3 file in pg dir
```

```
echo "..... copying pg config files over ....."
```

```
su - postgres -c "cp /home/start/oacs-installation-scripts/data/pg_hba.conf /usr/local/pgsql/data"
```

```
su - postgres -c "cp /home/start/oacs-installation-scripts/data/postgresql.conf /usr/local/pgsql/data"
```

```
/etc/init.d/pg82 restart
```

```
# Tune Postgres. The default values for PostgreSQL are very conservative; we can safely change some  
of them and improve performance.
```

```
echo 134217728 >/proc/sys/kernel/shmmax
```

```
# Make that change permanent by editing /etc/sysctl.conf to add these lines at the end
```

```
echo "" >>/etc/sysctl.conf
```

```
echo "#increase shared memory limit for postgres" >>/etc/sysctl.conf
```

```
echo "kernel.shmmax = 134217728" >>/etc/sysctl.conf
```

```
echo "..... DONE setting up PG82 ....."
```

## OpeACS.org

```
echo "..... DONE copying pg config files over ....."
```

---

### Arquivo 4

---

```
#!/bin/sh
```

```
# Run this script as root
```

```
# Set the UMASK so permissions are more open
```

```
umask 022
```

```
# First clean your old AOLserver installation
```

```
# TCL Version to use
```

```
TCL=tcl8.5.0
```

```
TCLLIB=1.11.1
```

```
XOTCL=1.6.0
```

```
# Path for the AOLserver installation
```

```
NS=/usr/local/aolserver40r10
```

```
# Path for Postgres. Change it if you did not install PG according to our instructions for PostgreSQL 8.2
```

```
PG=/usr/local/pgsql
```

## OpeACS.org

```
rm -rf ${NS}
```

```
rm -rf /usr/local/src/aolserver40r10
```

```
mkdir -p /usr/local/src/aolserver40r10
```

```
cd /usr/local/src/aolserver40r10
```

```
# Get all the files
```

```
wget http://mesh.dl.sourceforge.net/sourceforge/tcl/${TCL}-src.tar.gz
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co -r aolserver_v40_bp  
aolserver
```

```
echo "Getting aolserver modules ..."
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co nscache
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co nspostgres
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co nssh1
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co -r v2_7 nsoracle
```

```
echo "Getting TDOM ..."
```

```
cvs -z3 -d:pserver:anonymous@cvs.tdom.org:/usr/local/pubcvs co tdom
```

```
echo "Getting TCL modules ..."
```

```
wget http://downloads.sourceforge.net/tcllib/tcllib-${TCLLIB}.tar.gz?use_mirror=superb-west
```

```
wget http://mesh.dl.sourceforge.net/sourceforge/tcl/thread2.6.4.tar.gz
```

```
wget http://media.wu-wien.ac.at/download/xotcl-${XOTCL}.tar.gz
```

## OpeACS.org

```
wget http://dqd.com/~mayoff/aolserver/dqd_utils-1.7.tar.gz
```

```
cvs -d :pserver:anonymous@cvs.openacs.org:/cvsroot co openacs-4/packages/acs-core-docs/www/files/  
nsd-postgres.txt
```

```
# Install TCL
```

```
tar xfz ${TCL}-src.tar.gz
```

```
cd ${TCL}
```

```
cvs -z3 -d:pserver:anonymous@naviserver.cvs.sourceforge.net:/cvsroot/naviserver co vtmalloc
```

```
mv generic/tclThreadAlloc.c generic/tclThreadAlloc.c-orig
```

```
cp vtmalloc/tclThreadAlloc.c generic/
```

```
cd unix
```

```
./configure --enable-threads --prefix=${NS}
```

```
make install
```

```
# Install AOLserver and patch it for background delivery
```

```
cd /usr/local/src/aolserver40r10/aolserver
```

```
wget http://media.wu-wien.ac.at/download/aolserver-nsd-conn.patch
```

```
patch -p0 < aolserver-nsd-conn.patch
```

```
# Check the Libthread version.
```

```
# If return is "NPTL"+version you also need comment out line
```

```
#
```

```
# pthread_kill_other_threads_np()
```

## OpeACS.org

#

# in nsd/unix.c

getconf GNU\_LIBPTHREAD\_VERSION

# If you are running in a 64 bit environment you need to do the following:

# Comment out (using /\* \*/) the following paragraph in nsd/tclobj.c

#

# /\* if (sizeof(int) < sizeof(long)) {

# Tcl\_Panic("NsTclInitObjs: sizeof(int) < sizeof(long)");

# } \*/

CPPFLAGS="-nostartfiles" ./configure --prefix \${NS} --with-tcl=\${NS}/lib

make install

# Remove the symbolic link to aolserver.

# If this is not a symbolic link but the full path to your aolserver

# Make sure it becomes a symbolic link by issuing

# mv /usr/local/aolserver /usr/local/aolserver<yourversion>

rm -f /usr/local/aolserver

ln -s \${NS} /usr/local/aolserver



## OpeACS.org

```
# Install nscache and nssh1
```

```
cd /usr/local/src/aolserver40r10/nscache
```

```
make install AOLSERVER=${NS}
```

```
cd ../nssh1
```

```
make install AOLSERVER=${NS}
```

```
# Install nspostgres
```

```
cd ../nspostgres
```

```
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${PG}/lib
```

```
make install POSTGRES=${PG} ACS=1 AOLSERVER=${NS}
```

```
#somehow this is necessary. because when reboot the machine nspostgres breaks without that symlink
```

```
ln -s /usr/lib/libpq.so.5 /usr/local/pg82/lib/libpq.so.5
```

```
# Install tdom
```

```
# Note, if you use bash31 you need to apply a patch
```

```
# See http://openacs.org/forums/message-view?message\_id=369867 for details
```

```
# for 64 bit systems append "--enable-64bit"
```

```
cd ../tdom/unix
```

```
./configure --enable-threads --disable-tdomalloc --prefix=${NS} --exec-prefix=${NS} --with-aolserver=${NS} --with-tcl=/usr/local/src/aolserver40r10/${TCL}/unix
```

```
make install
```

## OpeACS.org

```
# Install TCLlib
```

```
cd /usr/local/src/aolserver40r10
```

```
tar -xzvf tcllib-${TCLLIB}.tar.gz
```

```
cd tcllib-${TCLLIB}
```

```
./configure --prefix=${NS}
```

```
make install
```

```
cd ..
```

```
# Install libthread
```

```
tar xzf thread2.6.4.tar.gz
```

```
cd thread2.6.4/unix/
```

```
./configure --enable-threads --prefix=${NS} --exec-prefix=${NS} --with-aolserver=${NS} --with-tcl=/usr/local/src/aolserver40r10/${TCL}/unix
```

```
make
```

```
make install
```

```
cd /usr/local/src/aolserver40r10
```

```
# Install XOTcl
```

```
tar xvfz xotcl-${XOTCL}.tar.gz
```

```
cd xotcl-${XOTCL}/
```

```
export CC=gcc
```

```
./configure --enable-threads --enable-symbols --prefix=${NS} --exec-prefix=${NS} --with-tcl=/usr/local/src/aolserver40r10/${TCL}/unix
```

## OpeACS.org

```
make
```

```
make install-aol
```

```
cd ..
```

```
# Install TRF for faster base encoding.
```

```
echo "Getting TRF ..."
```

```
cvs -z3 -d:pserver:anonymous@tcltrf.cvs.sourceforge.net:/cvsroot/tcltrf co -P trf
```

```
cd trf/
```

```
./configure --enable-threads --enable-symbols --prefix=${NS} --exec-prefix=${NS} --with-tcl=/usr/local/src/aolserver40r10/${TCL}/unix
```

```
make
```

```
make install
```

```
cd ..
```

```
# Make sure to call nsd-postgres instead of nsd if you use postgresql
```

```
cp openacs-4/packages/acs-core-docs/www/files/nsd-postgres.txt ${NS}/bin/nsd-postgres
```

```
chmod 755 ${NS}/bin/nsd-postgres
```

```
# Install dqd Utils: This is optional
```

```
export INST=${NS}
```

```
tar xzf dqd_utils-1.7.tar.gz
```

```
cd dqd_utils-1.7
```

## OpeACS.org

```
make install
```

```
cd ..
```

```
# Now install daemontools
```

```
# This is optional as some distro's don't like it.
```

```
mkdir -p /package
```

```
chmod 1755 /package
```

```
cd /package
```

```
wget http://cr.yp.to/daemontools/daemontools-0.76.tar.gz
```

```
wget http://www.qmail.org/moni.csi.hu/pub/glibc-2.3.1/daemontools-0.76.errno.patch
```

```
tar xvfz daemontools-0.76.tar.gz
```

```
cd admin/daemontools-0.76
```

```
patch -Np1 -i ../daemontools-0.76.errno.patch
```

```
patch -Np1 -i ../daemontools-0.76.errno.patch
```

```
package/install
```

```
cvs -d :pserver:anonymous@cvs.openacs.org:/cvsroot co openacs-4/packages/acs-core-docs/www/files/  
svgroup.txt
```

```
mv openacs-4/packages/acs-core-docs/www/files/svgroup.txt /usr/sbin/svgroup
```

```
chmod 700 /usr/sbin/svgroup
```

```
# Finally, care about tclwebtest
```

```
cd /usr/local/src/aolserver40r10
```

## OpeACS.org

```
# get the full release form sourceforge
```

```
wget 'http://downloads.sourceforge.net/tclwebtest/tclwebtest-1.0.tar.gz?modtime=1084492800&big_mirror=0'
```

```
mv tclwebtest-1.0.tar.gz?modtime=1084492800 tclwebtest-1.0.tar.gz
```

```
tar xzf tclwebtest-1.0.tar.gz
```

```
# get a patched version, which is often more than 70 times faster...
```

```
wget http://media.wu-wien.ac.at/download/tclwebtest.tcl
```

```
# install it
```

```
mkdir -p ${NS}/lib/tclwebtest
```

```
cp -f tclwebtest-1.0/lib/* ${NS}/lib/tclwebtest/
```

```
cp -f tclwebtest.tcl ${NS}/lib/tclwebtest
```

```
# Now, it is time to take care of postfix and some other packages via apt-get install
```

```
apt-get update
```

```
apt-get install postfix imagemagick graphviz gcc patch make unzip zip cvs build-essential bin86
```

## OpeACS.org

```
#!/bin/sh
```

```
apt-get install postfix imagemagick graphviz gcc patch subversion make unzip zip libbz2-dev zlib1g-dev cvs build-essential bin86
```

```
# Set the UMASK so permissions are more open
```

```
umask 022
```

```
# Which make to use
```

```
if test "x`which gmake`" = "x"; then
```

```
if test "x`which make`" = "x"; then
```

```
echo "Cannot find make or gmake on search path $(PATH)"
```

```
echo "Please install (g)make before continuing!"
```

```
exit -1
```

```
else
```

```
make="make"
```

```
fi
```

```
else
```

```
make=gmake
```

```
fi
```

```
# Which tar to use
```

```
if test "x`which gtar`" = "x"; then
```

## OpeACS.org

```
tar=tar
```

```
else
```

```
tar=gtar
```

```
fi
```

```
# Test which wget to use
```

```
if test "x`which wget`" = "x"; then
```

```
if test "x`which curl`" = "x"; then
```

```
echo "Cannot find wget on search path $(PATH)"
```

```
echo "Please install wget before continuing!"
```

```
exit -1
```

```
else
```

```
wget="curl -L -O"
```

```
fi
```

```
else
```

```
wget=wget
```

```
fi
```

```
# Versions to use
```

```
TCL=tcl8.5.6
```

```
XoTCL=1.6.3
```

```
TCLLIB=1.11
```

## OpeACS.org

# Path for the AOLserver installation

NS=/usr/local/aolserver45

# Path for Postgres. Change it if you did not install PG according to our instructions for PostgreSQL 8.2

PG=/usr/local/pg82

# optional packages

INSTALL\_VTMALLOC=0

INSTALL\_DQD\_UTILS=0

INSTALL\_DAEMONTOOLS=1

INSTALL\_TRF=1

INSTALL\_SSL=0

SSL\_PATH=/usr/lib/ssl

## Clean any old settings of aolserver installation

```
rm -rf ${NS}
```

```
rm -rf /usr/local/src/aolserver45
```

```
mkdir -p /usr/local/src/aolserver45
```

```
cd /usr/local/src/aolserver45
```

## Get all the files



## OpeACS.org

```
wget http://downloads.sourceforge.net/tcl/${TCL}-src.tar.gz
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co aolserver
```

```
echo "Getting aolserver modules ..."
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co nspostgres
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co nssha1
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co -r v2_7 nsoracle
```

```
echo "Getting TDOM ..."
```

```
wget http://www.tdom.org/files/tDOM-0.8.2.tgz
```

```
echo "Getting TCL modules ..."
```

```
wget http://downloads.sourceforge.net/tcllib/tcllib-${TCLLIB}.tar.bz2
```

```
wget http://downloads.sourceforge.net/tcl/thread2.6.5.tar.gz
```

```
wget http://media.wu-wien.ac.at/download/xotcl-${XoTCL}.tar.gz
```

```
wget http://www.pragana.net/tclmagick-simple-build.tar.gz
```

```
cvs -d :pserver:anonymous@cvs.openacs.org:/cvsroot co openacs-4/packages/acs-core-docs/www/files/  
nsd-postgres.txt
```

```
## Install TCL
```

```
tar xfz ${TCL}-src.tar.gz
```

```
if test "${INSTALL_VTMALLOC}" = "1"; then
```

## OpeACS.org

```
cd ${TCL}
```

```
cvs -z3 -d:pserver:anonymous@naviserver.cvs.sourceforge.net:/cvsroot/naviserver co vtmalloc
```

```
mv generic/tclThreadAlloc.c generic/tclThreadAlloc.c-orig
```

```
cp vtmalloc/tclThreadAlloc.c generic/
```

```
fi
```

```
cd ${TCL}/unix
```

```
./configure --enable-threads --prefix=${NS}
```

```
make install
```

### ## Compile AOLServer

```
cd /usr/local/src/aolserver45/aolserver
```

```
wget http://www.cognovis.de/aolserver45-driver.patch
```

```
patch -p0 < aolserver45-driver.patch
```

```
${NS}/bin/tclsh8.5 ./nsconfig.tcl
```

```
make -i install
```

### # Compile nsproxy

```
cd nsproxy
```

```
make install
```

## OpeACS.org

```
# Remove the symbolic link to aolserver.
```

```
# If this is not a symbolic link but the full path to your aolserver
```

```
# Make sure it becomes a symbolic link by issuing
```

```
# mv /usr/local/aolserver /usr/local/aolserver<yourversion>
```

```
rm /usr/local/aolserver
```

```
ln -s ${NS} /usr/local/aolserver
```

```
# Install nssh1
```

```
cd /usr/local/src/aolserver45/nssh1
```

```
make install NSHOME=${NS}
```

```
# Prepare Postgresql integration
```

```
cd /usr/local/src/aolserver45
```

```
cp openacs-4/packages/acs-core-docs/www/files/nsd-postgres.txt ${NS}/bin/nsd-postgres
```

```
chmod 755 ${NS}/bin/nsd-postgres
```

```
cd nspostgres
```

```
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:${PG}/lib
```

```
make install POSTGRES=${PG} ACS=1 NSHOME=${NS}
```

## OpeACS.org

# Intall TDOM

```
cd /usr/local/src/aolserver45/
```

```
tar -xvfz tdom-0.8.2.tgz
```

```
cd tDOM-0.8.2/unix
```

```
./configure --enable-threads --disable-tdomalloc --prefix=${NS} --exec-prefix=${NS} --with-aolserver=${NS} --with-tcl=/usr/local/src/aolserver45/${TCL}/unix
```

```
make install
```

# Install TCLlib

```
cd /usr/local/src/aolserver45
```

```
tar -xzvf tcllib-${TCLLIB}.tar.gz
```

```
cd tcllib-${TCLLIB}
```

```
./configure --prefix=${NS}
```

```
make install
```

# Install libthread

```
cd /usr/local/src/aolserver45
```

```
tar -xzvf thread2.6.5.tar.gz
```

## OpeACS.org

```
cd thread2.6.5/unix/
```

```
../configure --enable-threads --prefix=${NS} --exec-prefix=${NS} --with-aolserver=${NS} --with-tcl=/usr/local/src/aolserver45/${TCL}/unix
```

```
make
```

```
make install
```

```
# Install XoTCL
```

```
cd /usr/local/src/aolserver45
```

```
tar -xvfz xotcl-${XoTCL}.tar.gz
```

```
cd xotcl-${XoTCL}/
```

```
export CC=gcc
```

```
./configure --enable-threads --enable-symbols --prefix=${NS} --exec-prefix=${NS} --with-tcl=/usr/local/src/aolserver45/${TCL}/unix
```

```
make
```

```
make install-aol
```

```
# Install tclmagick
```

```
cd /usr/local/src/aolserver45
```

```
cd ${NS}/lib
```

```
tar -xzfz /usr/local/src/aolserver45/tclmagick-simple-build.tar.gz
```

## OpeACS.org

```
# Install TRF

cd /usr/local/src/aolserver45

if test "${INSTALL_TRF}" = "1"; then

echo "Getting TRF ..."

cvs -z3 -d:pserver:anonymous@tcltrf.cvs.sourceforge.net:/cvsroot/tcltrf co -P trf

cd trf/

./configure --enable-threads --enable-symbols --prefix=${NS} --exec-prefix=${NS} --with-
tcl=/usr/local/src/aolserver45/${TCL}/unix

make

make install

cd /usr/local/src/aolserver45

fi

# Install TLS and OPENSLL

if test "${INSTALL_SSL}" = "1"; then

cd /usr/local/src/aolserver45

apt-get install libssl-dev

wget http://downloads.sourceforge.net/tls/tls1.5.0-src.tar.gz

tar -xzip tls1.5.0-src.tar.gz

cd tls1.5

./configure --with-ssl-dir=/usr --with-tcl=${NS}/lib --enable-threads --enable-shared --prefix=${NS} --
exec-prefix=${NS}
```

## OpeACS.org

```
make install
```

```
cd /usr/local/src/aolserver45
```

```
cvs -z3 -d:pserver:anonymous@aolserver.cvs.sourceforge.net:/cvsroot/aolserver co nsopenssl
```

```
cd nsopenssl
```

```
make install OPENSSSL=${SSL_PATH} AOLSERVER=${NS}
```

```
cd /usr/local/src/aolserver45
```

```
fi
```

```
# Install DQD Utils
```

```
if test "${INSTALL_DQD_UTILS}" = "1"; then
```

```
cd /usr/local/src/aolserver45
```

```
wget http://dqd.com/~mayoff/aolserver/dqd_utils-1.7.tar.gz
```

```
export INST=${NS}
```

```
tar xzvf dqd_utils-1.7.tar.gz
```

```
cd dqd_utils-1.7
```

```
make install
```

```
cd /usr/local/src/aolserver45
```

```
fi
```

## OpeACS.org

```
# Install Daemontools
```

```
if test "${INSTALL_DAEMONTTOOLS}" = "1"; then
```

```
mkdir -p /package
```

```
chmod 755 /package
```

```
cd /package
```

```
wget http://cr.yp.to/daemontools/daemontools-0.76.tar.gz
```

```
wget http://www.qmail.org/moni.csi.hu/pub/glibc-2.3.1/daemontools-0.76.errno.patch
```

```
tar -xvfz daemontools-0.76.tar.gz
```

```
cd admin/daemontools-0.76
```

```
patch -Np1 -i ../../daemontools-0.76.errno.patch
```

```
package/install
```

```
cvs -d :pserver:anonymous@cvs.openacs.org:/cvsroot co openacs-4/packages/acs-core-docs/www/files/  
svgroup.txt
```

```
mv openacs-4/packages/acs-core-docs/www/files/svgroup.txt /usr/sbin/svgroup
```

```
chmod 700 /usr/sbin/svgroup
```

```
fi
```

```
# Install TCL Webtest
```

```
cd /usr/local/src/aolserver45
```

```
# get the full release form sourceforge
```

```
wget http://downloads.sourceforge.net/tclwebtest/tclwebtest-1.0.tar.gz
```



## OpeACS.org

```
tar -xzvf tclwebtest-1.0.tar.gz
```

```
# get a patched version, which is often more than 70 times faster...
```

```
wget http://media.wu-wien.ac.at/download/tclwebtest.tcl
```

```
# install it
```

```
mkdir -p ${NS}/lib/tclwebtest
```

```
cp -r tclwebtest-1.0/lib/* ${NS}/lib/tclwebtest/
```

```
cp tclwebtest.tcl ${NS}/lib/tclwebtest
```

---

## Arquivo 6

---

```
#!/bin/sh
```

```
#1. download openacs
```

```
# - cvs -d:pserver:anonymous@cvs.openacs.org:/cvsroot checkout openacs-4
```

```
echo "..... downloading OACS from cvs HEAD ....."
```

```
cd /usr/local/aolserver/servers/
```

```
# OACS HEAD
```

```
cvs -z6 -d:pserver:anonymous@cvs.openacs.org:/cvsroot checkout openacs-4
```

```
#OACS v.5.4
```

## OpeACS.org

```
#cvs -z6 -d :pserver:anonymous@cvs.openacs.org:/cvsroot co -r oacs-5-4 openacs-4
```

```
echo "..... DONE download OACS from cvs HEAD ....."
```

#2. create user openacs and add to group web with: `useradd -g web openacs`

```
echo "..... setting up OACS - HEAD ....."
```

```
export SERVICE0=openacs
```

```
export SERVICE0_HOME_DIR=/usr/local/aolserver/servers/${SERVICE0}
```

```
mv openacs-4 ${SERVICE0}
```

```
useradd -g web -d ${SERVICE0_HOME_DIR} ${SERVICE0}
```

```
chown -R ${SERVICE0}.web ${SERVICE0_HOME_DIR}
```

```
chmod -R 755 ${SERVICE0_HOME_DIR}
```

#3. Add the path for the postgres user

```
echo "export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/local/pgsql/lib" >>${SERVICE0_HOME_DIR}/.profile
```

```
echo "export PATH=${PATH}:/usr/local/pgsql/bin" >>${SERVICE0_HOME_DIR}/.profile
```

#4. configure openacs config.tcl file, and copy init.d script over

```
cp /home/start/oacs-installation-scripts/etc/config.tcl /usr/local/aolserver/servers/openacs/etc
```

```
cp /home/start/oacs-installation-scripts/etc/daemontools/run  
/usr/local/aolserver/servers/openacs/etc/daemontools/
```

```
cp /home/start/oacs-installation-scripts/init.d/oacs-init /etc/init.d/oacs-init
```

## OpeACS.org

```
chmod -R 755 /etc/init.d/oacs-init
```

```
#5. setup the init script: update-rc.d oacs-test1 defaults 99 01
```

```
update-rc.d oacs-init defaults 99 01
```

```
echo "..... DONE setting up OACS - HEAD ....."
```

```
#6. su -postgres, createuser openacs, createdb openacs, createlang plpgsql openacs (note, for some reason the user has to be the same as the db)
```

```
echo "..... setting up pg db, pg user and lang ....."
```

```
su - postgres -c "createuser -a -d -s openacs -U postgres && createdb -E UNICODE openacs && createlang plpgsql openacs"
```

```
su - postgres -c "psql -f /home/start/oacs-installation-scripts/pg82_path.sql openacs"
```

```
echo "..... DONE setting up pg db, pg user and lang ....."
```

```
#7. setting up daemontools
```

```
ln -s /usr/local/aolserver/servers/${SERVICE0}/etc/daemontools /service/${SERVICE0}
```

```
svgroup web /service/*
```

```
#8. startup openacs server, go through initial install process
```

```
echo "..... starting up OACS service ....."
```

```
/etc/init.d/oacs-init stop
```

## **OpeACS.org**

```
/etc/init.d/oacs-init start
```

```
echo "..... DONE starting up OACS service ....."
```

```
echo "In a few seconds OACS - CVS will have fully loaded."
```

```
echo "Your server is now running on 192.168.1.15, point your browser to it."
```