Aplicações Verticais para o OpenACS

Eduardo Santos

eduardo.edusantos@gmail.com
eduardo.santos@planejamento.gov.br

www.softwarepublico.gov.br

Roadmap

- De acordo com *Phillip Greenspun*, os passos para desenvolver aplicações:
  - Faça a modelagem de dados. O que você vai representar?
  - Desenvolva uma série de scripts para criar o banco (SQL).
  - Desenhe o fluxo de páginas. Como será a interação com o usuário?
  - Implemente as páginas individualmente

Modelagem de Dados

Modelagem

- A minha aplicação vai ter:
  - Armazenamento de notas;
  - Controle de versões para as notas;
  - Permissões nas notas;
  - Comentários sobre os dados inseridos.

Modelagem

- A minha aplicação vai ter:
  - Armazenamento de notas;
  - Controle de versões para as notas;
  - Permissões nas notas;
  - Comentários sobre os dados inseridos.

(Opcionais)

Modelagem

- Questão 1: onde armazenar as notas.
  - As notas devem utilizar o sistema de objetos?
  - Como controlar versões de objetos?
  - Onde elas ficarão armazenadas fisicamente: no sistema de arquivos ou no banco de dados?
  - Como você resolveria isso sem o OpenACS?

Modelagem

- Tutorial My First Package

```
-- creation script
--
-- @author joel@aufrecht.org
-- @cvs-id &Id:$
--

select content_type__create_type(
    'mfp_note',                 -- content_type
    'content_revision',           -- supertype
    'MFP Note',                 -- pretty_name,
    'MFP Notes',                 -- pretty_plural
    'mfp_notes',                 -- table_name
    'note_id',                 -- id_column
    null                      -- name_method
);

-- necessary to work around limitation of content repository:
select content_folder__register_content_type(-100,'mfp_note','t');
```

Modelagem

- Utilizando o sistema de objetos, permissões e categorias vêm como bônus.
- Nenhuma tabela extra será criada para isso!!!

Modelagem

- Um vez criadas as tabelas, como armazenar os dados nas tabelas?

Modelagem

- Um vez criadas as tabelas, como armazenar os dados nas tabelas?
- API's (*Application Programing Interface*)

Modelagem

- Criando uma biblioteca:

```
ad_library {

    Procs to add, edit, and remove notes for My First Package.

    @author oumi@arsdigita.com
    @cvs-id $Id: note-procs.tcl,v 1.2 2004/02/04 16:47:34 joela Exp $

}

namespace eval mfp {}
namespace eval mfp::note {}
```

API's

```
ad_proc -public mfp::note::get {
    -item_id:required
    -array:required
} {
    This proc retrieves a note.  This is annoying code that is only
    here because we wanted to give you a working tutorial in 5.0 that
    uses content repository, but the tcl api for content repository
    won't be complete until 5.1.  At least we can use the pregenerated views for select
and edit.
} {
    upvar 1 $array row
    content::item::get -item_id $item_id \
        -revision "live" \
        -array_name row
    #set row(user_name) "$row(first_names) $row(last_name)"
    set row(user_name) [acs_user::get_element -user_id $row(creation_user) -element
name]
}
```

```
ad_proc -public mfp::note::add {
    -title:required
    -user_id:required
} {
    This proc adds a note.
    @return item_id of the new note.
} {
    db_transaction {
        set item_id [content::item::new -name $title \
            -parent_id "-100"  \
            -creation_user $user_id \
            -item_subtype "content_item" \
            -content_type "mfp_note"\
            -title $title]
            #[ -context_id context_id ] [ -package_id package_id ]
            #[ -creation_ip creation_ip ]
            #[ -item_id item_id ] [ -locale locale ]
            #[ -creation_date creation_date ]
            #[ -description description ] [ -mime_type mime_type ]
            #[ -nls_language nls_language ] [ -text text ] [ -data data ]
            #[ -relation_tag relation_tag ] [ -is_live is_live ]
            #[ -storage_type storage_type ] [ -attributes attributes ]
            #[ -tmp_filename tmp_filename ]]
```

API's

```
set revision_id [db_nextval acs_object_id_seq]
content::revision::new -revision_id $revision_id -item_id $item_id \
    -title $title \
    -creation_user $user_id \
    -is_live "t"
    #[ -content_type content_type ] \
    [ -description description ] [ -content content ] \
    [ -mime_type mime_type ] [ -publish_date publish_date ] \
    [ -nls_language nls_language ] [ -creation_date creation_date ] \
    [ -content_type content_type ] [ -creation_user creation_user ] \
    [ -creation_ip creation_ip ] [ -package_id package_id ] \
    [ -attributes attributes ]  \
    [ -tmp_filename tmp_filename ] [ -storage_type storage_type ]
```

API's

```
        db_exec_plsql create {
            select acs_object__new (
                :user_id,   --new__object_id
                'dono',     --new__object_type
                NULL,       --new__creation_date
                :user_id,   --new__creation_user
                NULL,       --new__creation_ip
                NULL,       --new__context_id
                NULL,       --new__security_inherit_p
                :title,     --new__title
                NULL        --new__package_id
            );
        }
    }
    return $item_id
}
```

API's

```
ad_proc -public mfp::note::edit {
    -item_id:required
    -title:required
    {-user_id ""}
} {
    This proc edits a note. Note that to edit a cr_item, you insert a new revision instead
of changing the current revision.
} {
    db_transaction {
        set revision_id [db_nextval acs_object_id_seq]

        if {![exists_and_not_null user_id]} {
            set user_id [ad_conn user_id]
        }
```

API's

```
        content::revision::new -revision_id $revision_id -item_id $item_id \
            -title $title \
            -creation_user $user_id \
            -is_live "t"
            #[ -content_type content_type ] \
            [ -description description ] [ -content content ] \
            [ -mime_type mime_type ] [ -publish_date publish_date ] \
            [ -nls_language nls_language ] [ -creation_date creation_date ] \
            [ -content_type content_type ] [ -creation_user creation_user ] \
            [ -creation_ip creation_ip ] [ -package_id package_id ] \
            [ -attributes attributes ]  \
            [ -tmp_filename tmp_filename ] [ -storage_type storage_type ]
    }
}
```

API's

```
ad_proc -public mfp::note::delete {
    -item_id:required
} {
    This proc deletes a note.
} {
    content::item::delete -item_id $item_id
}
```

Roadmap

- De acordo com *Phillip Greenspun*, os passos para desenvolver aplicações:
  - Faça a modelagem de dados. O que você vai representar? -OK
  - Desenvolva uma série de scripts para criar o banco (SQL). -OK
  - Desenhe o fluxo de páginas. Como será a interação com o usuário?
  - Implemente as páginas individualmente

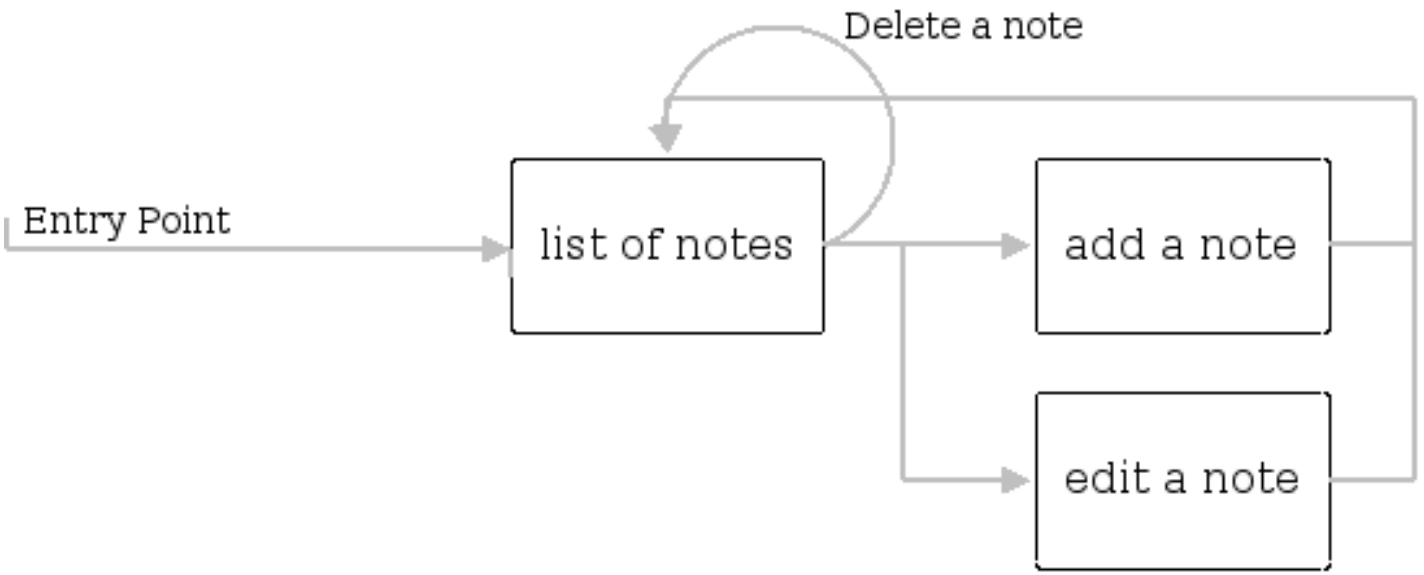
Fluxo de Páginas

Fluxo

- Questão 2: o que o usuário deve fazer?

Fluxo

- Questão 2: o que o usuário deve fazer?
  - Adicionar uma nota
  - Editar uma nota
  - Apagar uma nota
  - Listar as notas



Information Architecture

Code Map

Roadmap

- De acordo com *Phillip Greenspun*, os passos para desenvolver aplicações:
  - Faça a modelagem de dados. O que você vai representar? -OK
  - Desenvolva uma série de scripts para criar o banco (SQL). -OK
  - Desenhe o fluxo de páginas. Como será a interação com o usuário? -OK
  - Implemente as páginas individualmente

Implementação

Implementação

- Uma vez definido o que sua aplicação vai fazer, implemente as páginas individualmente.