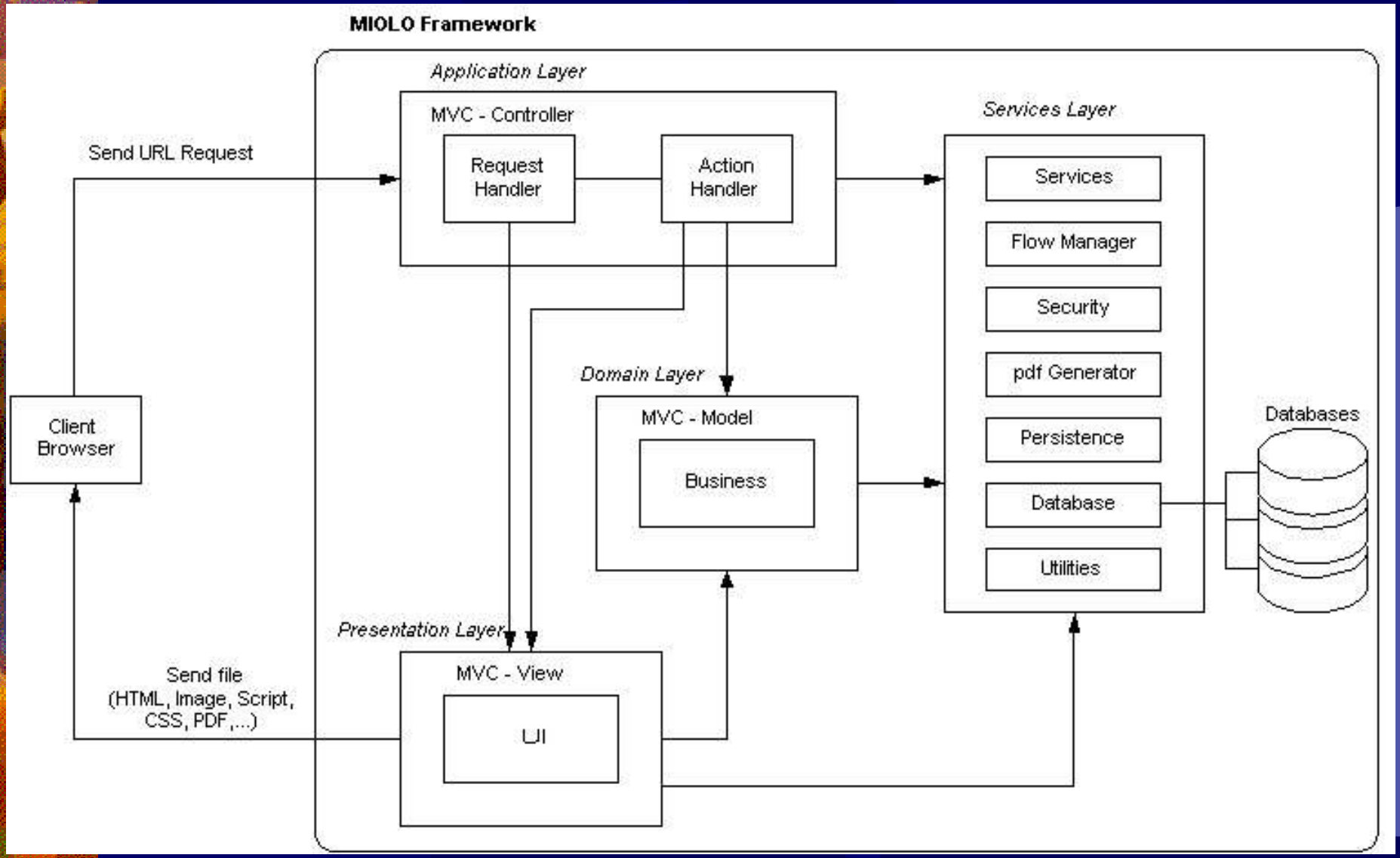
MDS2

Ely Edison Matos
ely.matos@ufjf.edu.br
fev/2005

O que é o MVDC?

- Framework para desenvolvimento de aplicações web
  - Controles de interface com o usuário escritos em PHP e renderizados em HTML
  - Autenticação de usuários
  - Controle de permissão de acesso
  - Camada de abstração para acesso a bancos de dados
  - Gerenciamento de sessões
  - Manutenção de log
  - Tratamento da página como um webform, com captura de eventos
  - Validação de entrada em formulários
  - Customização de layout e temas
  - Geração de arquivos PDF

Arquitetura



Camadas

MIOLO

Representa o framework, expondo métodos que fazem a integração entre as diversas camadas. Implementa o padrão Façade.

User Interface (UI)

São as classes do framework responsáveis pela geração de arquivos, renderização dos controles HTML e da criação dos scripts javascript enviados ao cliente, com base no tema em uso

Handlers

São as classes que representam a parte funcional da aplicação, criadas pelo desenvolvedor para fazer o tratamento dos dados enviados pelo cliente.

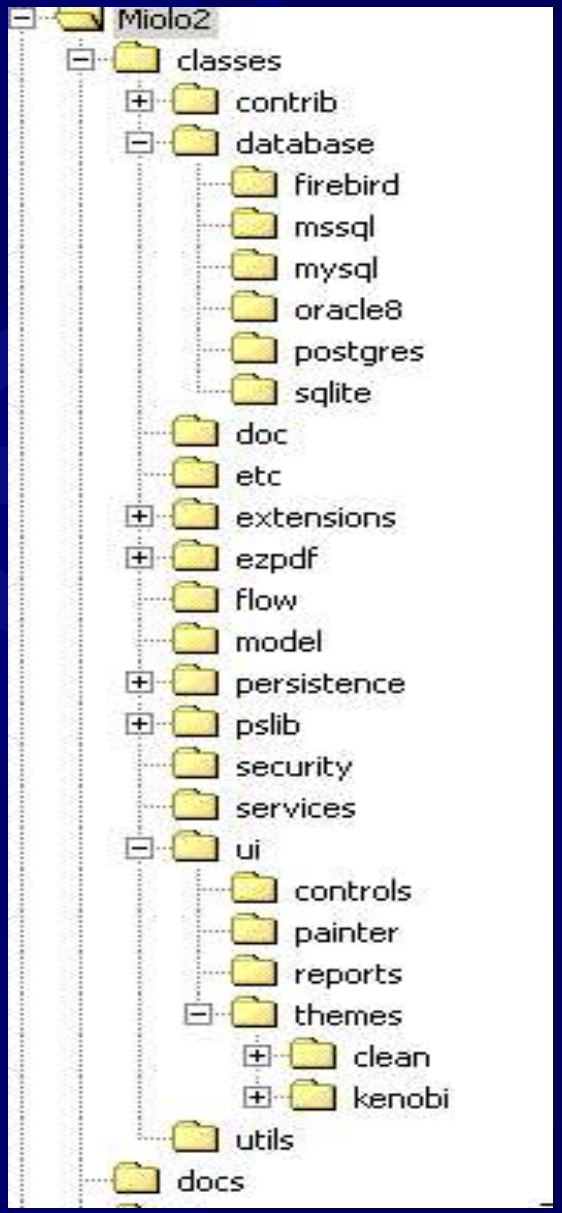
Acessa a camada de negócios para desempenhar suas funções e usa a camada UI para definir a saída para o cliente.

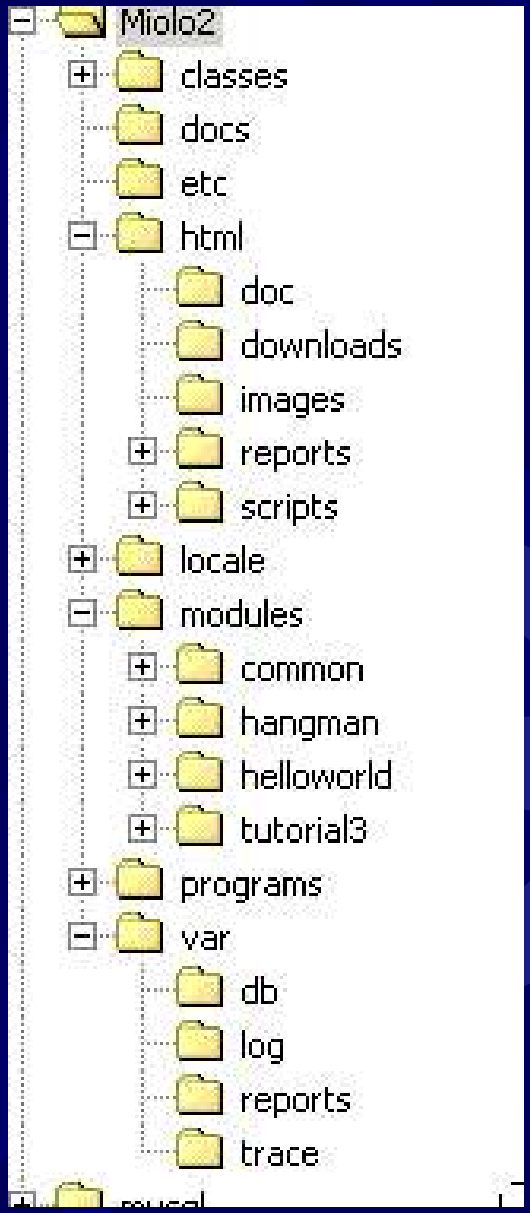
Estão localizadas no diretório handlers de cada módulo.

Camadas

- Business
  - São as classes criadas pelo desenvolvedor para representar o domínio da aplicação (as regras de negócio).
  - São usadas pelas camadas UI e handlers, acessando o banco de dados através da camada BD.
- BD
  - São as classes do framework responsáveis por abstrair o acesso às bases de dados, tornando as classes da camada Business independentes do SGBD usado.

Diretórios


Miolo2
classes
contrib
database
firebird
mssql
mysql
oracle8
postgres
sqlite
doc
etc
extensions
ezpdf
flow
model
persistence
pslib
security
services
ui
controls
painter
reports
themes
clean
kenobi
utils
docs


Miolo2
classes
docs
etc
html
doc
downloads
images
reports
scripts
locale
modules
common
hangman
helloworld
tutorial3
programs
var
db
log
reports
trace

Arquivos principais

- <nido>/html/index.html
- <nido>/html/index.php
- <nido>/etc/nido.conf
- <nido>/classes/support.inc
- <nido>/html/scripts/common.js
- <nido>/classes/nido.class

Conceitos básicos

Aplicação

Os sistemas são construídos através do desenvolvimento de módulos. O conjunto de módulos é chamado de aplicação. Assim, de forma geral, cada instalação do framework está associada a uma única aplicação, composta por um ou vários módulos integrados.

Conceitos básicos

Módulo

- Um módulo é um componente de uma aplicação.
- Um módulo reflete um sub-domínio da aplicação, agregando as classes de negócio que estão fortemente relacionadas e provendo o fluxo de execução (handlers) e a interface com o usuário (forms, grids, reports) para se trabalhar com tais classes.
- Um módulo é caracterizado por um nome, usado como subdiretório do diretório <miolo>/modules.
- Cada módulo tem uma estrutura de diretórios padrão, que é usada pelo framework para localizar os recursos
- Possui seu próprio arquivo de configuração

Conceitos básicos

- Controles (Widgets)
  - Os controles são componentes de interface com o usuário, usados na renderização das páginas html. Um controle pode agregar outros controles e tem propriedades e eventos associados a ele.
- Página
  - A página é um controle específico (instancia do da classe IPage) que serve de base para a renderização de uma página HTML.

Conceitos básicos

- Tema
  - Um tema é um controle específico (instanciado da classe Theme) que trabalha como um container para os controles que vão ser renderizados na página HTML.
  - Cada tema define "elementos", e cada elemento agrega controles visuais específicos. Um tema é associado a uma folha de estilos (um arquivo CSS) que define o posicionamento, as dimensões e a aparência dos controles a serem renderizados.
  - Podem ser definidos vários temas (cada um com seu próprio diretório no diretório <modulo>/classes/ui/themes), embora geralmente cada módulo utilize apenas um tema.

Conceitos básicos

Handler

Um handler é uma instância da classe Handler. Sua função é tratar a solicitação feita pelo usuário através do browser.

Em cada módulo (no diretório <miolo>/modules/<modulo>/handler) é definida uma classe Handler<Modulo>, que é instanciada pelo MIOLO quando é feita a análise da solicitação do usuário.

O controle é então passado para esta classe, que inclui o handler específico para tratar a solicitação. O handler atua, assim, no papel de controller, fazendo a integração entre as regras de negócio e a interface com o usuário.

Conceitos básicos

- Namespace
  - Namespaces são apenas aliases para diretórios.
  - O objetivo do uso de namespaces é a possibilidade de mudança física dos arquivos, sem a necessidade de se alterar o código já escrito.
  - Os namespaces são usados basicamente no processo de importação (include) de arquivos, em tempo de execução.

Desenvolvimento

- Modelagem das classes e do banco de dados
- Criação da estrutura do módulo, com seus subdiretórios
- Definição do tema a ser utilizado (um já existente, ou a criação de um novo tema)
- Criação de controles específicos para o módulo, caso seja necessário
- Criação do arquivo de configuração do módulo em <miolo>/modules/<modulo>/etc/module.conf
- Criação da classe handler em <miolo>/modules/<modulo>/handler/handler.class
- Criação do handler principal em <miolo>/modules/<modulo>/handler/main.inc
- Criação das classes de negócio em <miolo>/modules/<modulo>/classes – caso existam
- Criação dos formulários em <miolo>/modules/<modulo>/forms – caso existam

http://host.dominio/index.php?module=<module>&action=<action>[& lista de parâmetros]

Esta estrutura é generalizada para acessar:

A - Handlers

http://host.dominio/index.php?module=common&action=main:login

B – Arquivos

Imagens:

http://host.dominio/index.php?module=common&action=html:images:save.png

PDF:

http://host.dominio/index.php?module=common&action=html:files:exemplo.pdf

Texto:

http://host.dominio/index.php?module=common&action=html:files:exemplo.txt

CSS:

http://host.dominio/index.php?module=miolo&action=themes:kenobi:theme.css

C – Templates (por enquanto usado só para customizacao de temas)

Ciclo de vida de uma página

- Mdlo::RequestHandler()
  - NewMcontext
  - Se for arquivo
    - Mdlo::SendFile()
  - Senão
    - Mdlo::Init()
    - Mdlo::Handler()

Código inicial da página

```
Midas::Handler()
    page = new MPage();
    GetTheme();
    GetPainter();
    CheckLogin();
    InvokeHandler($startup,'main');
    page->Generate();
```

Código de uma página

```
Mdoc::InvokeHandler(module,action)
    handler = GetHandler($module);
    handler->dispatch($action);
Handler::Dispatch(action)
    Define global vars
    Include arquivo.inc em module/handlers
Handler típico
    ui = $MCLO->GetUI()
    form= ui->GetForm($module,'formName')
    theme->SetContent($form)
    Invoke Handler again...
```

Webforms

- Cada página gerada pelo processamento da seqüência de handlers, constitui-se em um único formulário html, mesmo que vários controles estejam presentes na página
- É importante observar, portanto, que os controles colocados na página devem ter nomes distintos, mesmo que estejam em objetos diferentes (por exemplo, os botões de submit de duas entradas de dados diferentes)
- Como padrão, os formulários usados pelos handlers são armazenados em um arquivo chamado <nome_do_form>.class, que contém a definição da classe do formulário (derivada da classe Form), com os métodos dos formulário e os tratadores dos eventos (tipicamente as funções para tratar os eventos OnClick dos botões de submit)

Web Forms

- $this->page->IsPostBack
  - Para testar se a página está sendo chamada a primeira vez, ou se ocorreu um "post" do formulário
  - $this->page é uma instância da classe MPage, que representa as definições para a página atualmente sendo executada
- Para usar as variáveis cujo estado foi mantido entre round-trips, usamos os métodos do objeto State
- Eventos
  - Os botões do tipo "submit" são programados para gerar eventos quando clicados
  - O nome do evento tem o formato <nome_do_botão>_click
  - Pode-se usar o método AttachEventHandler, para definir um outro nome para o método que vai tratar o evento

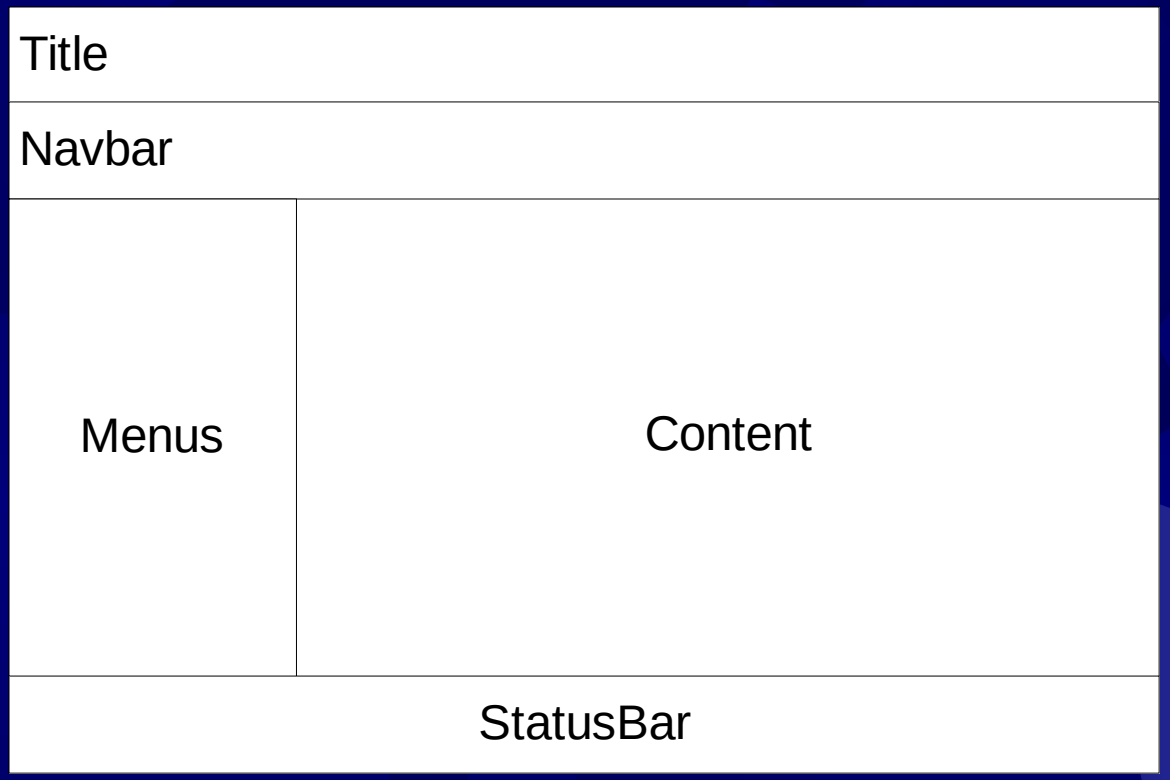
WebForms

- EventHandler
  - No processamento do formulário, quando a página é submetida, usa-se o método EventHandler para que o manipulador do evento (um método do formulário que estejamos tratando) seja executado
  - O método EventHandler também trata eventos "forçados" através da URL (quando é usado o método GET, ao invés do POST). Um evento na URL é definido através da variável *event*
    - http://.../handler.php?module=...&action=...&event=btnPost:click

Temas

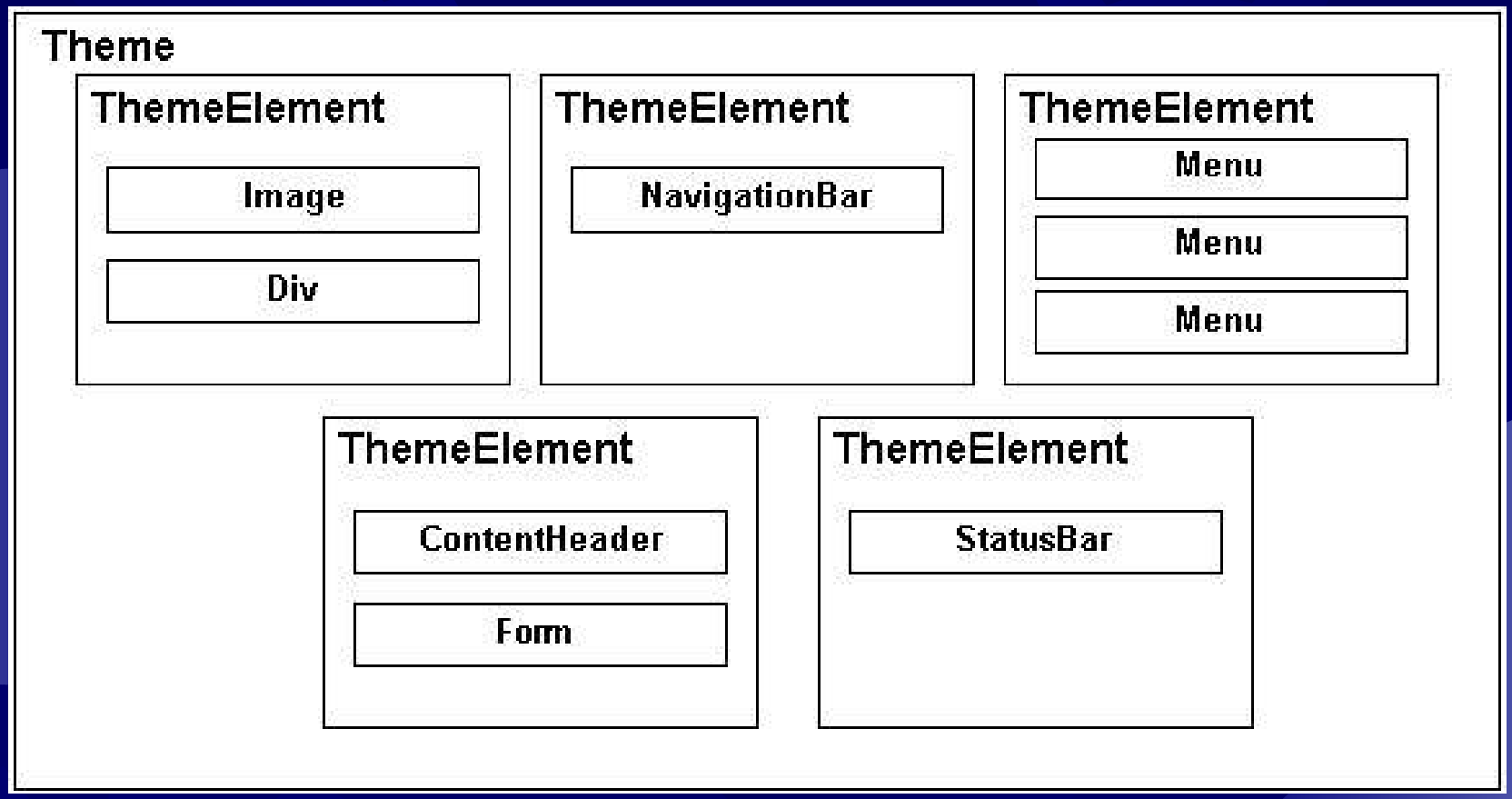
- Denominamos "tema", no ambiente MICLO, à definição do layout da página html que será enviada para o cliente
- O tema pode ser considerado como um container lógico para a página, sendo formado por
  - elementos visíveis (o conteúdo da tag body, do html)
  - por elementos não-visíveis (os arquivos CSS, os scripts, as tag meta do html)
- A definição e manipulação do tema é sustentada por algumas classes internas ao framework MICLO
- O tema portanto deve definir não apenas como a página será "dividida", mas também como os controles html serão renderizados

Temas



Cada uma destas áreas é definida por um elemento do Tema (classe ThemeElement) e manipulada através dos métodos expostos pela classe Theme

Temas

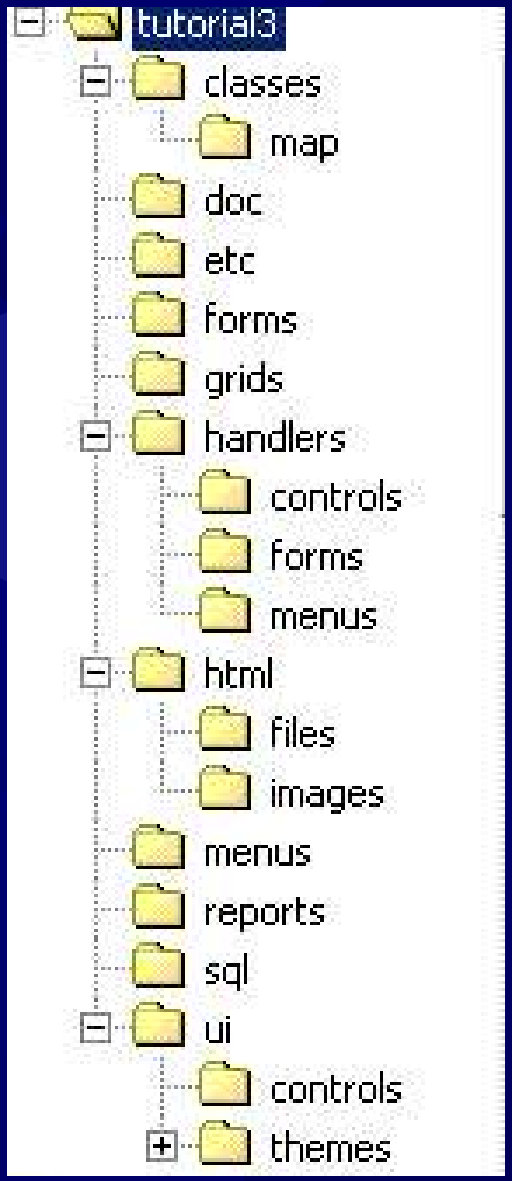


Cada ThemeElement é renderizado como um controle HTML Div, com um atributo “ id” ou “ class” , definido no arquivo theme.css relativo ao tema

Temas

- Os handlers são responsáveis por gerar o conteúdo de cada uma das áreas visíveis
- Os elementos não-visíveis são gerados pelo framework
- A renderização do conteúdo do tema em uma página html é feita através da chamada ao método
  $page->Generate()
- Para esta renderização são utilizados a classe Theme (theme.class) para os elementos do tema e o arquivo com a folha de estilo CSS (theme.css)
  - Definidos no diretório <miolo>/classes/ui/themes/<nome do tema> ou em <miolo>/modules/<modulo>/ui/themes/<nome do tema>

Modules



Todos os sistemas ficam localizados abaixo do diretório “ modules”

Cada módulo ou sistema, para possibilitar o total reaproveitamento de código (seja de formulários, menus ou instruções SQL), deve separar essas informações, colocando-os em seus respectivos diretórios

Variáveis globais

As seguintes variáveis são definidas como globais sendo disponibilizadas para todos os módulos chamados pelo Miolo

- $MIOLO: acesso a instancia da classe principal do framework
- $context: acesso ao objeto MContext
- $action: url completa do handler em execução
- $item: campo item da url atual
- $session: acesso ao objeto session (sessão atual)
- $page: acesso ao objeto MPage
- $url: url completa do handler em execução
- $theme: acesso ao objeto Theme
- $auth: acesso ao objeto auth
- $perms: acesso ao objeto perms
- $navbar: acesso ao objeto NavigationBar (barra de navegacao)
- $module: nome do módulo do handler em execução (ex:

TODOs

Em andamento:

- Documentação, documentação, documentação
- Disponibilizar no CodigoLivre
- ISR (Index-Search-Retrieve)
- Generalizar o Painter: HTML, XHTML, PDF, XML,...

A fazer/estudar/pesquisar:

- Uso de XML para Forms, Grids e Controles. Import/Export de ambientes gráficos - ex. Glade-GTK, QT, Delphi(!?)
- Retormar o projeto Miolo-IDE
- Uso opcional/obrigatório de cookies para controle da sessão
- Reestruturar procedimentos de Login (Autenticação, criptografia, etc)
- Módulo de Administração/Common – re-design
- Herança em classes de negócios
- Ferramenta para administração do ambiente de produção
- Ferramenta de análise dos logs