

Softworks

Ediointi Framework: Módulo 1

- ediointi pure, template e titer (exemplos dos recursos de framework)
- framework fica em lib (môs antigo), usar /app para criar próprias componentes em widgets
- parte visual no /control
- todo método do framework recebe \$param \Rightarrow \$REQUEST
- em /service de p/criar scripts que serão executados no cmd : php8.2.cmd.php "class=..."

Módulo 2

- Template (ex: TemplateViewBasico.php), máscaras, loops
- Janelas, TWindows, endSemord, Modal...

- Portion (ex: `dividePageItem.php`)
- Conteúdo embocado (ex: `ExternalSystemItem.php`, `Embedded...`)
- Tabelas (ex: `TableItem.php`)
- Panel Group (componente de card)
- Scroll (ex: `ScrollItem.php`)
- Notebook (diridir page em abas)
- Boxes (ex: `VBoxItem.php`, existe `HBox`)
- Dialogs (ex: `InformationItem.php`, `QuestionItem.php`)
- Input Dialogs (ex: `DialogInputItem.php`)
- PageBreaker juntando outras páginas em uma; é um div embocado, entre variações de mesma nome podem conflitar (ex: `ContainerPageBreaker.php`)
- Page Frame: sole a página com o iframe (ex: `ContainerPageFrame.php`); não conflita, mas visual não é integrado

- Módulo 3:**
- htaccess protege urlcesso.
 - sempre que mexer na BD, usar excessão.
 - Transaction pega conexão da /app/config
 - ex: `ConexaoManual.php`, `ConexaoPrepare.php` (evita SQL injection)
 - Transaction usa ActiveRecord (model de dados)
 - Modelos em /model
 - Inserir (ex: `ObjectStore.php`, `ObjectCreate.php`)
 - Carregar objeto (ex: `ObjectLoad.php`), dá excessão caso não encontre
 - Procurar objeto (ex: `ObjectFind.php`), não aborta execução
 - Update (ex: `ObjectUpdate.php`)
 - Tabelas associadas (ex: `ObjectAssociate.php`)
 - Converter objeto para array (ex: `ObjectArray.php`) ou JSON (ex: `ObjectJson.php`)
 - Printar atributos do objeto (ex: `ObjectRender.php`)
 - evaluate() é uma fórmula configurável

- Stamps (ex: Cliente.php: 'CREATED_AT'...) e Blocks (ex: Cliente.php: métodos on...), bom p/ debug
 - Critérios: filtros (ex: CollectionCount.php, Coload.php, Count.php)
- Diferença Active Record (ObjectStore.php) X Repository (CollectionCount.php):

→ AR:

- representa UM registro da Tabela
- permite store(), delete(), load(\$id)
- bom para criar/editar um único objeto

→ R:

- representa UM conjunto de registros
- permite load(), count(), delete(\$critério)
- bom para buscar, contar ou operar vários registros ao mesmo tempo

- Critérios update e update em lote (ex: CUpdate, CBatchUpdate), delete em lote (ex: CDelete)
- Shortcuts para escrita crua com critérios sobre filtros (ex: CollectionShortcuts.php)
- Manipulando objetos relacionados (ex: ObjectRelacionado.php)
- Agregações (ex: CollectionAggregation.php)
- soft delete (ex: Customer.php: 'DELETED_AT'...); marca como excluir, bom para manter histórico (ex: iSoftDelete.php)
- fake Transactions não lança exceção caso dê problema no BD (ex: TransactionFake.php); bom caso use open() select
- comandos DDL: criar tabelas, colunas, etc (ex: ComandosDDL.php)
- SQLite não funciona DROP
- comandos DML: comandos SQL primitivos sem usar as orientações a objetos usadas anteriormente (ex: ComandosDML.php)
- Readers e Writers, coleções criptográficas (ex: Cliente.php), usuários (ex: ObjectReaderWriter.php), classe Criptográfica (ex: EncryptionServices.php)

- mensagem de relacionamento do tentar deletar (ex: Client.php 'delete()')
- = Jaws (ex: ModelJaws.php), só para juntar com 'true'
- Find Cache: usa cache para evitar buscar mesmo sobre vários negócios na BD (ex: ObjectFindCache.php)

Módulo 4:

- tudo em /formularios
- usar static no método não recorre a página, dados ficam em \$param
- colocar [] no nome da campo tem o valor, dados de lista ficam em 'list - data'
- Componentes de seleção BD (ex: /bonos/formularios/SelecaoBanco.php)
- Acessibilidade: BootstrapFormBuilder, IFieldList tem método generateAria()

Módulo 5:

- tudo em /datagrids
- onReload preeenche datagrid

Módulo 6:

- usar framework para não precisar fazer métodos onShow, onEdit
- tudo em /cadastros
- usar Trait para gerar ActiveRecord (cria métodos CadastroForm.php)
- Trait para listagem
- VendaList.php tem VendaFormItem.php como controlador VendaForm.php
- register false ⇒ false não modifica URI

Módulo 7:

- relatórios em /relatorios
- fatura via template em /resources

- gráficos usa templates em /resources de geofe (pizz, bairros, cidades, linhas)
- etiquetas códigos de barra e QR semelhantes
- BarcodeQRCode.php em /formularios

Módulo 8:

- tude em /utilitarios

Módulo 9:

- sempre trocar nome application
- no application.ini é possível mudar p/ O ou t (visível ou não)
- communication.db, log.db, permission.db (instalar o SQL p/ banco c/DNS)
- cada grupo tem programas, usuários pertencem a grupos e podem ter programas específicos (permission.sql)
- restringir métodos (programas → métodos restritos), usuários devem ter papel (usuários → papéis)
- ativar log em /app/config = 'log' => "SystemLogLogService"
- para monitorar classe precisa ativar no classe use SystemChangeLogTrait (log de alterações)
- request.log, ativar no application.php
- gestor de comunicação (communication.db), ativar e desativar em 'marcar' application.php
- para mandar notificações e por código (TestEmail.php)
- ligar debug quando estiver em desenvolvimento (application), ver trace
- ver /var/log/apache2/access.log para depurar servidor
- o -dump () gera dumps trace (p/ variáveis), odumps () gera dumps trace (p/ transações da BD), só coloca método dentro da transação
- do p/strar public view da página sem login, menus: menu-public.xml, páginas precisam ser liberadas em permission(application)

- p/ redefinir senha é preciso configurar SMTP em administrações → preferências
- p/ notificar login precisa colocar em rotina assíncrona no cron tab (Linux).
`php cmd.php "class = SystemAccessNotificationLoginService & method = sendNotificationLogin & static = 1" (1 em 1 minute)`
- dupla fator (google auth), termos de aceite, recibos
- tradução /app/lib/util (ver TestUtil.php)
- sessões concorrentes (permite ou não 2 logins no mesmo user)
- multunit para trabalhar com multi unidades
- multi-database permite indicarmos na configuração qual banco de dados daquela unidade, init-database.php conecta a unidade escolhida (ver MultidatabaseUtil.php)
- templates tem o css de bootstrap e custom.css é edição geral (variações de bootstrap)
- verificar inspeções para custom.css
- para trabalhar com agendamento é necessário criar classe em app/service/gabs (para ativar): `php cmd.php "class = SystemScheduleService & method = run" (no Linux user: crontab -e), sempre user crontab para rodar de 5 em 5 minutos`
- strict-request para aumentar segurança
- master-menu, page-table, etc.
- para diminuir URL trocar nome .htaccess - edit → .htaccess e user setRouter em init.php, se adicionar classe adicionar em .htaccess
- Restful services, mudar nome rest-secure.php dist → rest.php, habilitar .htaccess, criar classe /app/service/rest e adicionar em .htaccess, definir chave de acesso rest-key em application.php
- Em /rest tem exemplos de Restful em php e request.php acesso rest de HTTPI
- tem PWA, dá pra adicionar aplicação no celular adicionando o ícone inicial pelo navegador (manifest.json, icons/icon)

Seg Ter Qua Qui Sex Sáb Dom

/ /

Módulo 10

- BD erphouse
- classes em /model , /control

Módulo Extra:

- migração JUMP : php cmd.php "class=System\Template\MigrationJob & method=run & from=/senacominfo"
- mais seguro : migração STEP
- editor erosive DDOS : vim /etc/apache2/mods-available/erossive.conf ,
a2 enmod erosive e a2 dismod erosive
- Ex: \$this → datagrid → disableHtmlConversion() pode permitir XSS (perigo)
- CSRF, user post p/ ações importantes, já que get permite tudo, case tanks
disable HTML conversion, do pone injetor script post
- \$this → form → enableCSRFProtection() protege form
- proteger arquivos com apache x .htaccess
- usar https (LetEncrypt)