Linguagem de Programação II

Prof.ª Ma. Jessica Oliveira

FBr Faculdade Brasília

Aula 15 – 11/11/2024

Utilização de *Templates* e *Web Services*.

Faculdade Brasília

Templates com Smarty

O que é o *Smarty*?

- É uma ferramenta poderosa para desenvolvedores PHP que permite separar completamente a lógica da aplicação (PHP) da interface do usuário (HTML).
- Em outras palavras, o *Smarty* atua como uma ponte entre o código PHP e a estrutura visual do site.

Como ele funciona?

- ***Templates*:** são criados arquivos de *template* (normalmente com a extensão `.tpl`) que contêm o HTML da página, mas com *placeholders* para os dados dinâmicos que virão do código PHP.
- **Atribuição de valores:** no código PHP, são atribuídos valores a essas variáveis e passa-se o *template* para o *Smarty*.
- **Renderização:** o *Smarty* pega o *template*, substitui os *placeholders* pelos valores correspondentes e gera o HTML final, que é então enviado para o navegador do usuário.

Vantagens

- Desenvolvedores e *designers* podem trabalhar de forma mais independente, pois a lógica do PHP fica isolada dos *templates*, ou seja, há uma espécie de separação de responsabilidades.
- Alterações na *interface* podem ser feitas diretamente nos *templates*, sem afetar o código PHP.
- *Templates* podem ser reutilizados em diferentes partes do *site*, reduzindo a duplicação de código.
- O *Smarty* oferece mecanismos de segurança para evitar injeções de código e outros tipos de ataques.

Vantagens

- O *Smarty* compila os *templates* em código PHP, o que pode melhorar o desempenho da sua aplicação.
- Possui uma grande comunidade e diversos recursos disponíveis, como *plugins* e documentação.

Desvantagens

- Desenvolvedores acostumados apenas com PHP puro precisam aprender uma nova sintaxe para criar os *templates*.
- Em aplicações muito grandes e complexas, o *Smarty* pode adicionar um *overhead* de processamento, impactando ligeiramente o desempenho.
- Configurar o *Smarty* pode ser um pouco mais complexo do que utilizar PHP puro, especialmente para projetos menores.
- É uma ferramenta excelente para organizar e modularizar o código de uma aplicação *web*, mas não é a solução ideal para todos os casos.

Web Services

O que é *web service*?

- É um método de comunicação entre diferentes aplicações, geralmente através da internet.
- Imagine-os como serviços que uma aplicação pode "contratar" de outra. Por exemplo, uma aplicação de *e-commerce* pode usar um *web service* de um serviço de pagamento para processar transações.

Principais características

- **Plataforma independente:** podem ser acessados por qualquer aplicação, independentemente da linguagem de programação ou sistema operacional.
- **Protocolos:** comumente utilizam protocolos como SOAP (*Simple Object Access Protocol*) e REST (*Representational State Transfer*).
- **Formato de dados:** geralmente utilizam XML ou JSON para trocar informações.

O que é uma API?

- Uma **Interface de Programação de Aplicativos** é um conjunto de regras e especificações que define como diferentes *softwares* podem interagir entre si.
- É como um contrato que define quais são as funcionalidades que um *software* oferece e como outros *softwares* podem acessá-las.
- Em resumo, uma API é a porta de entrada para os recursos de um sistema.

RESTful APIs

- Já as RESTful APIs são um tipo específico de API que seguem os princípios da arquitetura REST. Essa arquitetura enfatiza a simplicidade, a escalabilidade e a utilização de protocolos HTTP padrão.
- Principais características:
  - **Recursos:** cada peça de informação é tratada como um recurso, identificado por um URI (*Uniform Resource Identifier*).
  - **Métodos HTTP:** utilizam os métodos HTTP (GET, POST, PUT, DELETE) para realizar operações nos recursos (obter, criar, atualizar e deletar).
  - ***Stateless*:** cada requisição contém todas as informações necessárias para ser atendida, não há estado armazenado no servidor entre as requisições.
  - **Representação:** os dados são retornados em formatos como JSON ou XML.

Por que usar APIs?

- **Reutilização de código:** em vez de recriar funcionalidades já existentes, é possível aproveitar APIs de terceiros para integrar essas soluções ao seu projeto.
- **Integração entre sistemas:** APIs facilitam a comunicação e troca de dados entre diferentes sistemas, promovendo uma interoperabilidade eficiente.
- **Aceleração do desenvolvimento:** ao usar APIs, os desenvolvedores podem focar nas funcionalidades específicas de suas aplicações, reduzindo o tempo necessário para entregar novos produtos.

Por que usar APIs?

- **Fomento à inovação:** APIs permitem criar novos serviços e produtos, além de incentivar a colaboração e a troca de ideias entre desenvolvedores.
- **Escalabilidade:** projetadas para atender a um grande número de usuários e requisições, APIs oferecem suporte a sistemas que precisam crescer sem comprometer o desempenho.

Importante saber!

- **Todos os *web services* são APIs:** pois eles expõem uma interface para que outros sistemas possam se comunicar com eles.
- **Nem todas as APIs são *web services*:** uma API pode ser local, por exemplo, para comunicação entre diferentes componentes de um mesmo aplicativo.
- **RESTful APIs são um tipo específico de *web service*:** elas utilizam o protocolo HTTP e os princípios REST para expor funcionalidades e dados.

Resumindo...

- *Web services* são a categoria mais ampla, representando qualquer mecanismo de comunicação entre sistemas.
- APIs são interfaces que permitem o acesso a recursos de um sistema.
- RESTful APIs são um tipo de API que utiliza os princípios REST e o protocolo HTTP para oferecer uma forma simples e eficiente de comunicação entre sistemas.

Vamos para a prática?

OBS.: a prática vale nota APENAS para quem estava em sala.

Objetivo

- Criar uma aplicação simples que exiba uma lista de tarefas utilizando:
  - ***Smarty*:** para separar lógica (PHP) e apresentação (HTML).
  - **API RESTful:** para fornecer os dados da lista de tarefas no formato JSON.

Estrutura do projeto

- Crie a seguinte estrutura, dentro da pasta **htdocs**:

```
/projeto
├── api.php
├── index.php
├── libs
├── templates
│   └── tarefas.tpl
├── templates_c
└── cache
```

Instalação do *Smarty*

- Acesse o site **`smarty.net`** e aguarde as instruções.

API RESTful

- Abra o documento **`api.php`** para digitar o código a seguir.
- Este arquivo fornece os dados da lista de tarefas no formato JSON.

```php
<?php
header('Content-Type: application/json');
$method = $_SERVER['REQUEST_METHOD'];
$tarefas = ['Estudar PHP', 'Finalizar projeto', 'Revisar conteúdos para AV2'];
switch ($method) {
    case 'GET':
        echo json_encode($tarefas);
        break;
    case 'POST':
        $data = json_decode(file_get_contents('php://input'), true);
```

```
        if (isset($data['tarefa'])) {
            $tarefas[] = $data['tarefa'];
            echo json_encode(['mensagem' => 'Tarefa adicionada
com sucesso!', 'tarefas' => $tarefas]);
        } else {
            echo json_encode(['mensagem' => 'Erro: Nenhuma
tarefa fornecida!']);
        }
        break;
    default:
        echo json_encode(['mensagem' => 'Método não
suportado!']);
        break;
}
```

Página principal

- Abra o documento **`index.php`** para digitar o código seguinte.
- Este arquivo consome os dados da API e renderiza o *template* utilizando *Smarty*.

```php
<?php
$json = file_get_contents('http://localhost/projeto/api.php');

$tarefas = json_decode($json, true);

require 'libs/libs/Smarty.class.php';

$smarty = new Smarty\Smarty();
$smarty->setTemplateDir('templates/');
$smarty->setCompileDir('templates_c/');
$smarty->setCacheDir('cache/');

$smarty->assign('tarefas', $tarefas);

$smarty->display('tarefas.tpl');
```

Template

- Abra o documento **`tarefas.tpl`** para digitar o código a seguir.
- Este é o arquivo de apresentação da lista de tarefas, renderizado pelo *Smarty*.

```
<!DOCTYPE html>
<html>
<head>
    <title>Lista de Tarefas</title>
</head>
<body>
    <h1>Minhas Tarefas</h1>
    <ul>
        {foreach $tarefas as $tarefa}
            <li>{$tarefa}</li>
        {/foreach}
    </ul>
</body>
</html>
```

Testando a aplicação

- No navegador, acesse: **`localhost/projeto/index.php`** e verifique se a lista de tarefas está sendo exibida corretamente.

Atividade complementar

- Compõe a nota da prática de hoje, junto com o “**Questionário 4**” que estava no AVA e o **envio da pasta do projeto** de hoje, por e-mail.
- Resenha sobre os **métodos HTTP** para ser enviada, por e-mail, até amanhã às 20h.
- O conteúdo da resenha será cobrado na AV2 (e na AV3).
- **TUDO QUE FOR ENVIADO DEVE CONTER NOME E MATRÍCULA DO ALUNO.**
- **APENAS OS ALUNOS QUE ESTAVAM EM SALA RECEBERÃO NOTA DA PRÁTICA.**

Desafios extras (APENAS para quem estava em sala)

- **Adicionar uma nova tarefa:** modifique o `index.php` para enviar dados via **POST** para a API.
- **Excluir uma tarefa:** adicione suporte ao método **DELETE** na `api.php`.
- Quem enviar, por e-mail, **até o dia 15/11/2024** o projeto com esses extras funcionando, poderá receber **até 0,50 ponto extra** na média do 2º Bimestre.
- **Atenção:** enviar a pasta compactada e completa.

Dúvidas?

jessica.oliveira@fbr.edu.br