Programação de Sistemas para Internet

Prof. Diego Cirilo

Aula 16: Sessão/Cookies/Context Processors

HTTP é *Stateless*

- O protocolo HTTP não mantém estado entre requisições;
- Cada requisição é independente;
- O servidor não "lembra" de requisições anteriores;
- Informações completas vão em voltam no cabeçalho/corpo da requisição;
- Objeto `request` nas *views*;
- Problema: como manter usuário logado? Como manter um carrinho de compras?

Cookies

- Pequenos arquivos de texto armazenados no navegador;
- Enviados automaticamente em cada requisição ao servidor;
- Permitem "lembrar" informações entre requisições;
- Têm data de expiração (ou expiram ao fechar o navegador).

Cookies - Características

- Armazenados no **cliente** (navegador);
- Limite de tamanho (~4KB por cookie);
- Podem ser visualizados/editados pelo usuário;
- **Nunca armazene dados sensíveis** diretamente em cookies!
- Usados para: preferências, rastreamento, identificação de sessão.

Cookies no Django - Escrevendo

```python
def minha_view(request):
    response = render(request, 'pagina.html')

    # Define um cookie
    response.set_cookie('nome', 'valor')

    # Com opções
    response.set_cookie(
        'preferencia',
        'escuro',
        max_age=3600*24*30,  # 30 dias em segundos
        httponly=True,       # não acessível via JS
        secure=True,         # apenas HTTPS
    )

    return response
```

Cookies no Django - Lendo

```
def minha_view(request):
    # Lê um cookie
    nome = request.COOKIES.get('nome', 'valor_padrao')

    # Verifica se existe
    if 'preferencia' in request.COOKIES:
        preferencia = request.COOKIES['preferencia']

    return render(request, 'pagina.html', {'nome': nome})
```

Cookies no Django - Deletando

```
def logout_view(request):
    response = redirect('home')

    # Remove o cookie
    response.delete_cookie('nome')

    return response
```

Sessão (Session)

- Mecanismo para armazenar dados temporários **no servidor**;
- Cada usuário recebe um ID de sessão único;
- O ID é armazenado em um cookie no navegador;
- Os dados ficam seguros no servidor;
- Funciona mesmo com usuários anônimos (não logados).

Sessão no Django

- Habilitada por padrão;
- Middleware: `django.contrib.sessions.middleware.SessionMiddleware`
- App: `django.contrib.sessions` em `INSTALLED_APPS`
- Por padrão, armazena na tabela `django_session` do banco.

Middleware

- Camada de processamento entre a requisição e a view;
- Executa código **antes** e/ou **depois** de cada view;
- Configurado em `MIDDLEWARE` no `settings.py`;
- Exemplos: autenticação, sessão, CSRF, segurança;
- Cada middleware pode modificar o `request` ou o `response`.

Backends de Sessão

```
# settings.py

# Banco de dados (padrão)
SESSION_ENGINE = 'django.contrib.sessions.backends.db'

# Cache (mais rápido)
SESSION_ENGINE = 'django.contrib.sessions.backends.cache'

# Arquivo
SESSION_ENGINE = 'django.contrib.sessions.backends.file'

# Cookie assinado (dados no cliente, mas seguros)
SESSION_ENGINE = 'django.contrib.sessions.backends.signed_cookies'
```

Configurações de Sessão

```
# settings.py

# Tempo de expiração (em segundos) - padrão: 2 semanas
SESSION_COOKIE_AGE = 1209600

# Expira ao fechar o navegador
SESSION_EXPIRE_AT_BROWSER_CLOSE = True

# Atualiza expiração a cada requisição
SESSION_SAVE_EVERY_REQUEST = True

# Nome do cookie de sessão
SESSION_COOKIE_NAME = 'sessionid'
```

Usando Sessão - Escrevendo

```
def adicionar_ao_carrinho(request, produto_id):
    # A sessão funciona como um dicionário
    carrinho = request.session.get('carrinho', [])
    carrinho.append(produto_id)
    request.session['carrinho'] = carrinho

    # Força salvar (necessário para objetos mutáveis)
    request.session.modified = True

    return redirect('ver_carrinho')
```

Usando Sessão - Lendo

```python
def ver_carrinho(request):
    # Lê da sessão
    carrinho = request.session.get('carrinho', [])

    # Busca os produtos
    produtos = Produto.objects.filter(id__in=carrinho)

    return render(request, 'carrinho.html', {
        'produtos': produtos
    })
```

Usando Sessão - Deletando

```
def limpar_carrinho(request):
    # Remove uma chave específica
    if 'carrinho' in request.session:
        del request.session['carrinho']

    return redirect('home')

def logout_view(request):
    # Limpa toda a sessão
    request.session.flush()

    return redirect('home')
```

Métodos Úteis da Sessão

```
# Verifica se existe
if 'chave' in request.session:
    ...

# Remove com valor padrão
valor = request.session.pop('chave', None)

# Limpa tudo
request.session.clear()

# Gera novo ID de sessão (segurança)
request.session.cycle_key()

# Define expiração específica
request.session.set_expiry(3600)  # 1 hora
```

Exemplo: Carrinho de Compras

- Funciona para usuários anônimos e logados;
- Armazena IDs e quantidades dos produtos;
- Persiste entre páginas e visitas.

Carrinho - Estrutura

```
# Estrutura do carrinho na sessão: {id: quantidade}
carrinho = {
    '1': 2,   # produto 1, quantidade 2
    '5': 1,   # produto 5, quantidade 1
}
# Chaves são strings (JSON não aceita int como chave)
```

Carrinho - Adicionar Produto

```
def adicionar_produto(request, produto_id):
    # Pega o carrinho da sessão (ou dict vazio)
    carrinho = request.session.get('carrinho', {})

    # Converte para string (chave do dict)
    produto_id = str(produto_id)

    # Adiciona ou incrementa quantidade
    carrinho[produto_id] = carrinho.get(produto_id, 0) + 1

    # Salva na sessão
    request.session['carrinho'] = carrinho

    return redirect('ver_carrinho')
```

Carrinho - Ver Carrinho

```
def ver_carrinho(request):
    carrinho = request.session.get('carrinho', {})

    # Busca os produtos no banco
    produtos = Produto.objects.filter(id__in=carrinho.keys())

    # Adiciona a quantidade a cada produto
    itens = []
    for produto in produtos:
        qtd = carrinho[str(produto.id)]
        itens.append({
            'produto': produto,
            'quantidade': qtd,
            'subtotal': produto.preco * qtd
        })

    return render(request, 'carrinho.html', {'itens': itens})
```

Carrinho - Remover Produto

```
def remover_produto(request, produto_id):
    carrinho = request.session.get('carrinho', {})
    produto_id = str(produto_id)

    # Remove o produto se existir
    if produto_id in carrinho:
        del carrinho[produto_id]
        request.session['carrinho'] = carrinho

    return redirect('ver_carrinho')
```

Carrinho - Template

```
<h1>Seu Carrinho</h1>

{% for item in itens %}
<div class="item">
    <h3>{{ item.produto.nome }}</h3>
    <p>Preço: R$ {{ item.produto.preco }}</p>
    <p>Quantidade: {{ item.quantidade }}</p>
    <p>Subtotal: R$ {{ item.subtotal }}</p>
    <a href="{% url 'remover_produto' item.produto.id %}">Remover</a>
</div>
{% empty %}
<p>Carrinho vazio.</p>
{% endfor %}
```

Context Processors

- Funções que adicionam variáveis ao contexto de **todos** os templates;
- Evita repetir código nas views;
- Útil para dados globais: usuário, carrinho, configurações, menus.

Context Processors Padrão

```
# settings.py - já vem configurado
TEMPLATES = [{
    ...
    'OPTIONS': {
        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',  # request
            'django.contrib.auth.context_processors.auth', # user, perms
            'django.contrib.messages.context_processors.messages',
        ],
    },
}]
```

- Por isso `{{ user }}` e `{{ request }}` funcionam em qualquer template!

Criando um Context Processor

```
# app/context_processors.py

def carrinho_context(request):
    """Disponibiliza o carrinho em todos os templates"""
    from .cart import Carrinho
    return {
        'carrinho': Carrinho(request)
    }

def configuracoes_site(request):
    """Configurações globais do site"""
    return {
        'SITE_NAME': 'Minha Loja',
        'SITE_EMAIL': 'contato@minhaloja.com',
    }
```

Registrando o Context Processor

```
# settings.py
TEMPLATES = [{
    ...
    'OPTIONS': {
        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
            # Nossos context processors
            'app.context_processors.carrinho_context',
            'app.context_processors.configuracoes_site',
        ],
    },
}]
```

Usando no Template

```
<!-- Em qualquer template, sem passar na view -->
<header>
    <h1>{{ SITE_NAME }}</h1>
    <nav>
        <a href="{% url 'carrinho' %}">
            Carrinho ({{ carrinho.carrinho|length }} itens)
        </a>
    </nav>
</header>
```

- As variáveis estão disponíveis em **todos** os templates!

Exemplo: Menu Dinâmico

```
# context_processors.py
def menu_categorias(request):
    from .models import Categoria
    return {
        'menu_categorias': Categoria.objects.filter(ativo=True)
    }
```

```
<!-- base.html -->
<nav>
    {% for cat in menu_categorias %}
        <a href="{{ cat.get_absolute_url }}">{{ cat.nome }}</a>
    {% endfor %}
</nav>
```

Context Processor com Lógica

```python
# context_processors.py
def notificacoes(request):
    if request.user.is_authenticated:
        from .models import Notificacao
        nao_lidas = Notificacao.objects.filter(
            usuario=request.user,
            lida=False
        ).count()
        return {'notificacoes_nao_lidas': nao_lidas}
    return {'notificacoes_nao_lidas': 0}
```

Cuidados com Context Processors

- Executam em **toda** requisição que renderiza template;
- Evite queries pesadas;
- Use cache quando apropriado;
- Mantenha simples e focado.

Referências

- https://docs.djangoproject.com/en/5.1/topics/http/sessions/
- https://docs.djangoproject.com/en/5.1/ref/request-response/#django.http.HttpRequest.COOKIES
- https://docs.djangoproject.com/en/5.1/ref/templates/api/#writing-your-own-context-processors

Dúvidas?