

# Renderização no servidor

AJUDE A TRADUZIR ESTA PÁGINA

## Índice

Material-UI no servidor

Configurando

O tema

O lado do servidor

Manipulando a  
requisição

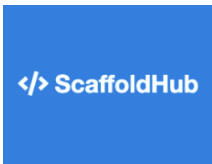
Injetar Componente  
Inicial HTML e CSS

O lado do cliente

Implementações de  
referência

Resolução de problemas

A situação de uso mais comum para a renderização do lado do servidor, é manipular a renderização inicial quando um usuário (ou rastreador de mecanismo de pesquisa) solicita sua aplicação.



**ScaffoldHub.** Automate building your full-stack Material-UI web-app.

ad by Material-UI

Quando o servidor recebe a solicitação, ele renderiza o(s) componente(s) requerido(s) em uma cadeia HTML e o envia como uma resposta ao cliente. A partir desse momento, o cliente assume as funções de renderização.

## Material-UI no servidor

O Material-UI foi projetado em base com garantias de renderização no servidor, mas cabe a você certificar-se de que ele será integrado corretamente. É importante fornecer a página com o CSS necessário, caso contrário a página irá renderizar somente o HTML até o CSS ser injetado pelo cliente, causando uma tremulação (FOUC). Para injetar o estilo no cliente, precisamos:

1. Crie uma instância nova do `ServerStyleSheets` em cada requisição.
2. Renderize a árvore React com o coletor do lado do servidor.
3. Capture o CSS.
4. Passe o CSS junto ao cliente.

No lado do cliente, o CSS será injetado uma segunda vez antes de remover o CSS injetado no lado do servidor.

## Configurando

No passo a passo a seguir, vamos ver como configurar a renderização do lado do servidor.

## O tema

Crie um tema que será compartilhado entre o cliente e o servidor:

theme.js

```
import { createMuiTheme } from '@material-ui/core/styles';
import red from '@material-ui/core/colors/red';

// Cria a instância do tema.
const theme = createMuiTheme({
  palette: {
    primary: {
      main: '#556cd6',
    },
    secondary: {
      main: '#19857b',
    },
    error: {
      main: red.A400,
    },
    background: {
      default: '#fff',
    },
  },
});

export default theme;
```

## O lado do servidor

A seguir um esboço para o aspecto do que o servidor deve lidar. Vamos montar um **middleware Express** usando **app.use** para lidar com todas as requisições que chegam ao servidor. Se você não estiver familiarizado com o Express ou middleware, saiba apenas, que a função `handleRender` será chamada toda vez que o servidor receber uma requisição.

server.js

```
import express from 'express';

// Vamos preenchê-las nas seções a seguir.
function renderFullPage(html, css) {
  /* ... */
}

function handleRender(req, res) {
  /* ... */
}

const app = express();

// Isso é acionado toda vez que o servidor recebe uma solicitação.
app.use(handleRender);

const port = 3000;
app.listen(port);
```

## Manipulando a requisição

A primeira coisa que precisamos fazer em cada solicitação é criar um novo `ServerStyleSheets`.

Quando renderizando, vamos encapsular `App`, o componente raiz, dentro de um `StylesProvider` e `ThemeProvider` para tornar a configuração de estilo e o `theme` disponíveis para todos os componentes na árvore de componentes.

A etapa principal na renderização do lado do servidor, é renderizar o HTML inicial do componente **antes** de enviarmos para o lado do cliente. Para fazer isso, usamos `ReactDOMServer.renderToString()`.

Em seguida, obtemos o CSS das `folhas` usando `sheets.toString()`. Vamos ver como isso é passado na função `renderFullPage`.

```

import express from 'express';
import React from 'react';
import ReactDOMServer from 'react-dom/server';
import { ServerStyleSheets, ThemeProvider } from '@material-ui/core/styles';
import App from './App';
import theme from './theme';

function handleRender(req, res) {
  const sheets = new ServerStyleSheets();

  // Renderiza o componente para string.
  const html = ReactDOMServer.renderToString(
    sheets.collect(
      <ThemeProvider theme={theme}>
        <App />
      </ThemeProvider>,
    ),
  );

  // Pega o CSS das folhas de estilo.
  const css = sheets.toString();

  // Envia a página renderizada de volta ao cliente.
  res.send(renderFullPage(html, css));
}

const app = express();

app.use('/build', express.static('build'));

// Isso é acionado toda vez que o servidor recebe uma solicitação.
app.use(handleRender);

const port = 3000;
app.listen(port);

```

## Injetar Componente Inicial HTML e CSS

A etapa final no lado do servidor é injetar o componente HTML e CSS inicial em um modelo a ser renderizado no lado do cliente.

```
function renderFullPage(html, css) {  
  return `  
    <!DOCTYPE html>  
    <html>  
      <head>  
        <title>Minha página</title>  
        <style id="jss-server-side">${css}</style>  
      </head>  
      <body>  
        <div id="root">${html}</div>  
      </body>  
    </html>  
  `;  
}
```

## O lado do cliente

O lado do cliente é simples. Tudo o que precisamos fazer é remover o CSS gerado no lado do servidor. Vamos dar uma olhada no arquivo do cliente:

client.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { ThemeProvider } from '@material-ui/core/styles';
import App from './App';
import theme from './theme';

function Main() {
  React.useEffect(() => {
    const jssStyles = document.querySelector('#jss-server-side');
    if (jssStyles) {
      jssStyles.parentElement.removeChild(jssStyles);
    }
  }, []);

  return (
    <ThemeProvider theme={theme}>
      <App />
    </ThemeProvider>
  );
}

ReactDOM.hydrate(<Main />, document.querySelector('#root'));
```

## Implementações de referência

Nós hospedamos diferentes implementações de referência que você pode encontrar no [repositório GitHub](#) sob a pasta o `/examples`:

- [A implementação de referência deste tutorial](#)
- [Gatsby](#)
- [Next.js](#)

## Resolução de problemas

Confira a resposta no FAQ: [Minha aplicação não é renderizada corretamente no servidor](#).

---

