



Web



PDF generation



WEB > GUIDES > PSPDFKIT SERVER > PDF GENERATION

Generate fillable PDF forms



PSPDFKit Server has been deprecated and replaced by [Document Engine](#). To migrate to Document Engine and unlock advanced document processing capabilities, refer to our [migration guide](#). Learn more about these enhancements on our [blog](#).

This guide will take you through the process of generating a PDF containing fillable form data.

JAVASCRIPT

PYTHON

JAVA

C# (.NET)

PHP

PDF Generation allows you to create fillable PDF forms from HTML. In this guide, you'll create a simple personalized form in JavaScript. You can use the templating language [Mustache](#), which allows you to inject data into a previously prepared HTML file. This is how your form will look:

```
1  <!DOCTYPE html>
2  <html>
3    <body>
4      <div class="subject">Subject: PDF Generation FTW!</div>
5      <div>Dear {{name}}<br/></div>
6      <div>
7        <p>
8          Fill the following form and send it back to us.
9        </p>
10       <p>
11         <label for="where">How did you find us:</label>
12         <input type="text" id="where" name="where" />
13       </br>
```



```

14     <label for="enjoy">Do you enjoy PDF Generation:</label>
15     <input type="checkbox" id="enjoy" name="enjoy" />
16   </p>
17 </div>
18   <div>Signed John Smith, Vienna on {{date}}</div>
19 </body>
20 </html>

```

First, the data for `{{name}}` and `{{date}}` needs to be defined. In practice, this data may come from an external source or database, but for this example, you'll define the data in a JSON file:

```

1  {
2    "name": "John Smith Jr.",
3    "date": "29 February, 2020"
4  }

```

Now, run the `mustache` command to produce a final HTML document with the template arguments replaced:

```

1  const mustache = require("mustache");
2  const fs = require("fs");
3
4  const page = fs.readFileSync("page.mustache").toString();
5  const data = JSON.parse(fs.readFileSync("data.json").toString());
6
7  const outputHTML = mustache.render(page, data);

```

Run the JavaScript with Node.js:

```
node mustache-example.js
```

Once you've created your HTML in JavaScript, send a request to the `/api/documents` endpoint, sending the [PDF Generation schema](#) with the HTML file you just created:

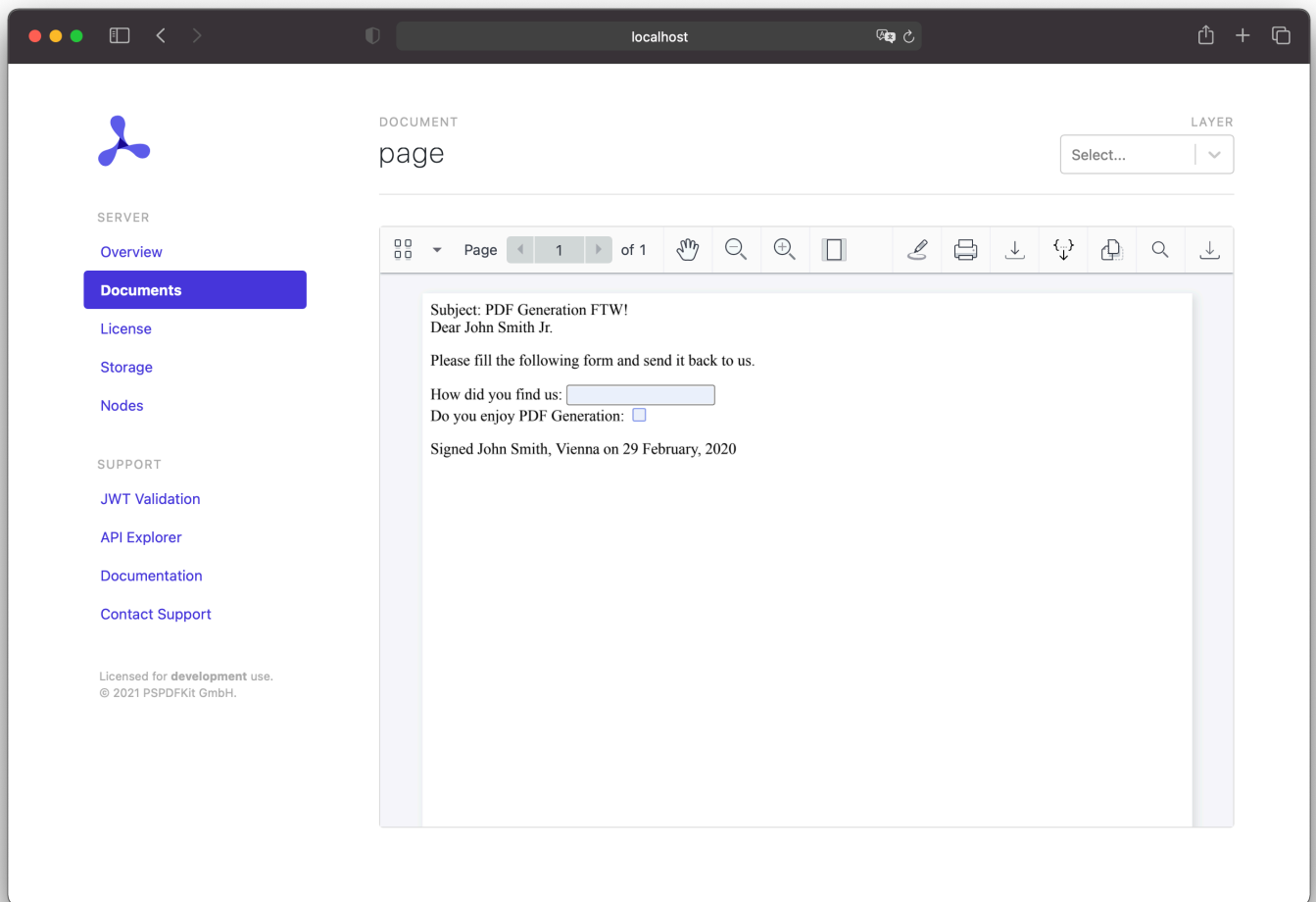
SHELL HTTP

```

1  curl -X POST http://localhost:5000/api/documents \
2    -H "Authorization: Token token=<API token>" \
3    -F page.html=@/path/to/page.html \
4    -F generation='{
5      "html": "page.html"
6    }'

```

The final PDF will look like this:



Adding watermarks

Once you have the basics of PDF generation down, you might want to watermark your PDF. You can do this by including some additional HTML:

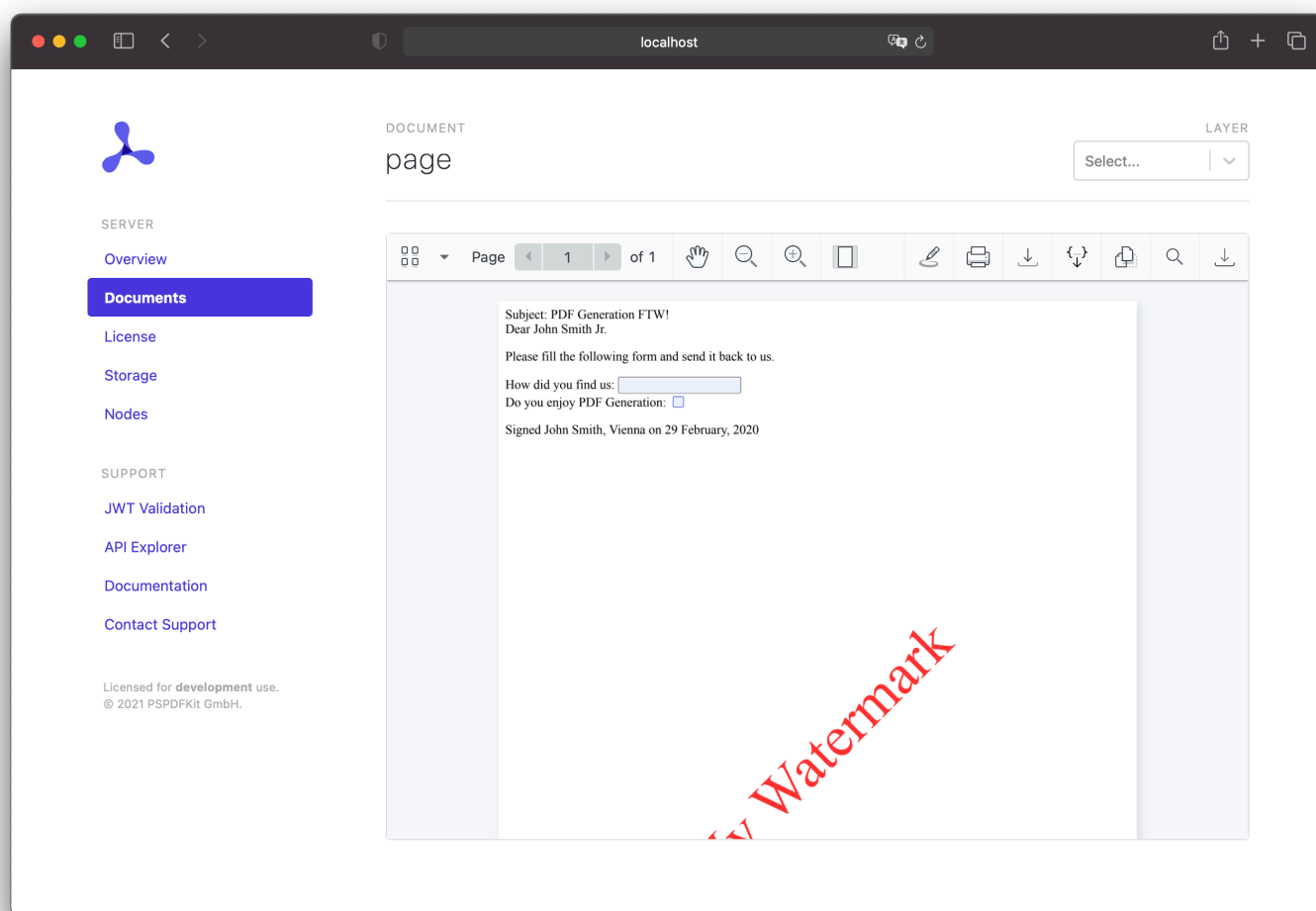
```
1  <!DOCTYPE html>
2  <html>
3    <body>
4      <div style="position: fixed;
5        top: 50%;
6        left: 50%;
7        font-size: 72px;
8        color: red;
9        opacity: 80%;
10       transform: rotate(-45deg);
11       width: 500px;
12       height: 500px;
13       margin-top: -250px;
14       margin-left: -250px;
15       text-align: center;
16       vertical-align: middle;
17       line-height: 500px;">
18        My Watermark
19      </div>
20      <div class="subject">Subject: PDF Generation FTW!</div>
21      <div>Dear {{name}}<br/></div>
```

```

22     <div>
23         <p>
24             Fill the following form and send it back to us.
25         </p>
26         <p>
27             <label for="where">How did you find us:</label>
28             <input type="text" id="where" name="where" />
29         </br>
30             <label for="enjoy">Do you enjoy PDF Generation:</label>
31             <input type="checkbox" id="enjoy" name="enjoy" />
32         </p>
33     </div>
34     <div>Signed John Smith, Vienna on {{date}}</div>
35 </body>
36 </html>

```

This element will be rendered on all pages on top of all other content. The HTML above uses some text, but you could also use an image. The watermark can also be positioned any way you want.



Adding a cover page

There are two ways to add a cover page to a PDF you generated: You can generate the cover page as part of the HTML, or you can use a PDF you already have.

Using HTML

If you want to add a cover page to your generated PDF, you can do so by adding additional HTML:



```
1  <!DOCTYPE html>
2  <html>
3    <body>
4      <div style="display: block; width: 100%; height: 100%; page-break-after: a
5        <h1>My Cover Page</h1>
6      </div>
7      <div class="subject">Subject: PDF Generation FTW!</div>
8      <div>Dear {{name}}<br/></div>
9      <div>
10        <p>
11          Fill the following form and send it back to us.
12        </p>
13        <p>
14          <label for="where">How did you find us:</label>
15          <input type="text" id="where" name="where" />
16        </br>
17          <label for="enjoy">Do you enjoy PDF Generation:</label>
18          <input type="checkbox" id="enjoy" name="enjoy" />
19        </p>
20      </div>
21      <div>Signed John Smith, Vienna on {{date}}</div>
22    </body>
23  </html>
```

This will add another page before the rest of the content, and you can use the new page as your cover page.

Using a PDF

Instead of adding some HTML as a cover page, you can add a PDF. This can be done by applying an `importDocument` operation on upload:

SHELL HTTP

```
1 curl -X POST http://localhost:5000/api/documents \
2   -H "Authorization: Token token=<API token>" \
3   -F page.html=@/path/to/page.html \
4   -F cover.pdf=@/path/to/cover.pdf \
5   -F generation='{
6     "html": "page.html"
7   }' \
8   -F operations='{
9     "operations": [
10      {
11        "type": "importDocument",
12        "beforePageIndex": 0,
13        "document": "cover.pdf"
14      }
15    ]
16  }'
```

The provided PDF will be appended before the first page of the HTML, and it'll serve as your cover page.

Form fields

PDF Generation supports the conversion of HTML forms into interactive PDF forms.

Conversion is handled completely internally, so there's no need to pass any extra information to the generation payload.

Because not all HTML form fields map directly to PDF forms, a subset of form fields is supported. The following table shows how the HTML `input` element types map to PDF form types.

INPUT	TYPE	PDF FORM FIELD
text		Text box
password		Text box where all characters are replaced with *
radio		Radio button
checkbox		Checkbox

INPUT TYPE	PDF FORM FIELD
select	Combo box

All other `input` types aren't supported and won't be converted to PDF form fields.

Form values

All form values and radio buttons/checkboxes that are checked in HTML will be carried over to the form field values in the generated PDF.

Form field names

The name of the form field in the generated PDF is determined based on the `id` and `name` attributes of the HTML form field. They're both concatenated with an `_` in between. What follows are some concrete examples.

Only ID set

Given this HTML:

```
<input type="text" id="textInput" />
```



The form field name would be:

```
textInput_
```



Both ID and name set

Given this HTML:

```
<input type="text" id="textInput" name="cool" />
```



The form field name would be:

```
textInput_cool
```

Only name set

Given this HTML:

```
<input type="text" name="cool" />
```

The form field name would be:

```
id_<random_id>_cool
```

Was this helpful?

✓ YES

✗ NO

Questions? [Contact us](#)

