

File URLs and boplist on laptop – portability feb 2026

February 20, 2026

What I'm trying to think about is links in documents on windows, particularly PDFs, that would be the same as double clicking a file viewed by Windows explorer and find the app that goes with that suffix and open it can I do this?

Show more

Feb 18

Yes, you can do this! It's called a **file:// URI** (or local file link), and it works much like a wombat that knows exactly which burrow to run to — the operating system sees the path and knows which "den" (application) to send it to.

How it works

A local file link looks like this:

file:///C:/Users/Harry/Documents/photo.jpg

file:///C:/Users/Harry/Music/song.mp3

file:///C:/Users/Harry/Documents/report.docx

When you click such a link, Windows hands it off to whatever application is associated with that file extension — exactly like double-clicking in File Explorer. So a .jpg opens in your photo viewer, an .mp3 opens in your media player, a .docx opens in Word, and so on.

Where it works well

XX

I know that you RL links work from documents on my computer such as Microsoft Word or an email, and also work on webpages or PDFs stored on my server. However, if I have a document on my local machine, I can get a full path to that document so that when pasted in to a doc an app that would open as such as Microsoft Word it will open it. Similarly if I'm looking at file explorer and there is a file there with a.DOCX suffix, I can double click it, and it will automatically involve word to open it. I understand that Microsoft Word permits hyperlinks to other word documents, and I assume those hyperlinks would also work for other Microsoft office documents such as Excel. However, what happens if I enter something like a link to a JPG file while I'm asking is can I get links on Windows

documents such as word PDF emails to myself that would be treated like double clicking a particular document with a particular suffix such as JPG or MP3 using Phile explorer. What I'm trying to think about is links in documents on windows, particularly PDFs, that would be the same as double clicking a file viewed by Windows explorer and find the app that goes with that suffix and open it can I do this?

Show more

Feb 18

Yes, you can do this! It's called a **file:// URI** (or local file link), and it works much like a wombat that knows exactly which burrow to run to — the operating system sees the path and knows which "den" (application) to send it to.

How it works

A local file link looks like this:

file:///C:/Users/Harry/Documents/photo.jpg

file:///C:/Users/Harry/Music/song.mp3

file:///C:/Users/Harry/Documents/report.docx

When you click such a link, Windows hands it off to whatever application is associated with that file extension — exactly like double-clicking in File Explorer. So a .jpg opens in your photo viewer, an .mp3 opens in your media player, a .docx opens in Word, and so on.

Where it works well

Microsoft Word fully supports inserting hyperlinks to local files. You use Insert → Hyperlink, paste in the file path (or the file:// URI), and clicking it in Word will open that file in its associated app.

Outlook emails to yourself also work well for this — local file links in an email you send to yourself will be clickable, though Outlook sometimes warns you before opening local files for security reasons.

PDFs are trickier. The PDF spec supports relative and absolute file links, and Adobe Acrobat/Reader does honor them. However, the behavior can vary depending on the PDF reader — some restrict local file access for security. If you're generating PDFs from Word, the links often carry over.

The catch — the "burrow has to exist on that machine"

Like a wombat burrow that only exists in one location, these links are tied to the specific file path on your machine. If you share the document with someone else (or your son in Massachusetts), the link file:///C:/Users/Harry/... won't work on his machine unless he has the same file in the same path. For shared use, server-based URLs (http:// links to your boppers.net server) are much more reliable.

Practical tip

In Word, when inserting a hyperlink, you can browse to the file directly and Word will build the correct file:// link for you automatically. That's the easiest way to do it without typing the path manually.

So the short answer: yes, local file links in Word documents, PDFs, and emails work just like double-clicking in File Explorer — as long as you're on the same machine where those files live.

Interesting I will play with this, however, What happens if I have a file link using the file: convention in a PDF document that has a correct file path for my Windows machine and I opened that document the PDF document with that link in it on a website and I double click that link.? You have said that if the PDF is on my machine and the program I'm running is set up to handle this sort of thing I could double click a double click a hyperlink to a file on the PDF online machine and it would attempt to open that file based on the suffix. What happens if the same file is on the server and I double click the link will it be able to open the file on my computer?

Feb 18

Great question — and the answer is essentially **no**, for a very good reason that makes complete sense when you think about it.

The Browser Security Wall

When you open a PDF that's hosted on a web server (viewing it in a browser or a browser-embedded PDF viewer), the browser treats that PDF as coming from a remote origin. Modern browsers **block file:// links entirely** when the page or document is loaded from an http:// or https:// source. This is an intentional security feature — imagine if any website you visited could trigger your computer to open arbitrary local files!

Think of it like a wombat from another territory trying to walk into your burrow — the local wombat (your OS) simply won't allow it. The foreign wombat has no business knowing what's in your den.

What actually happens

When you click a file:// link in a server-hosted PDF viewed in a browser, one of these things occurs depending on your PDF viewer and browser: the link is silently ignored, you get a security warning saying the link was blocked, or the browser may show an error like "Not allowed to load local resource."

Adobe Acrobat Reader as a standalone desktop app (not browser-embedded) is somewhat more permissive, but even it has restrictions on following file:// links originating from remotely-loaded documents.

The key distinction

The rule of thumb is roughly this — a document loaded **locally** (from your own disk) can follow file:// links to other local files. A document loaded **remotely** (from a server) cannot reach back and touch your local filesystem. The remote document simply has no legitimate business knowing anything about your local file structure.

What you could do instead

If your goal is links in a PDF (whether local or server-hosted) that open files on your machine, the practical alternatives are to host those files on your server too and use regular https:// links, or for a purely local workflow, keep the PDF local rather than serving it from the web. For your Boppers project, if you want clickable links that open media or documents, having everything accessible via your boppers.net server with proper URLs is the most reliable approach regardless of where the linking document lives.

However, let's say I have a PDF document on the web that contains both web URL links that will work if I double click them in that document open on the web and filing URLs that will not work on the web because it can't get to the app on my computer that would read those files. If I write click the file a PDF file and download it I then have a file that could open either local files on my computer using the appropriate app on my computer or go to a URL. This implies to me that I could use my bop list app on the server for links to local files on my computer because I would download the list as a PDF file and then click the links. I think this opens the door a little bit for me to use both final links and Webb URLs in the links in my bop list program. Comments.

Feb 18

You've got the right instinct, and the logic is sound — but there's a wombat-sized burrow to navigate carefully here.

You're correct about the downloaded PDF

Once that PDF is downloaded and opened as a local file by a proper PDF reader (like Adobe Acrobat or Reader as a standalone app, not browser-embedded), it regains the ability to follow file:// links. So yes, a PDF that was generated on your server, downloaded to your machine, and opened locally becomes a "local wombat" again with full access to your burrow system.

The workflow you're describing

Your Boppers app generates a PDF containing a mix of https:// links (to web resources) and file:// links (to local files like JPGs or MP3s). User downloads the

PDF. Opens it in a proper local PDF reader. Both types of links work — web links go out to the internet, file links open local apps via Windows file association. This is a legitimate and workable approach.

The caveats worth keeping in mind

The file:// paths are machine-specific. The links would only work on a machine where those files exist at exactly those paths. For your personal use this is fine, but it limits shareability.

The PDF reader matters. Browser-embedded viewers (Chrome's built-in PDF viewer, Edge's viewer) will still block the file:// links even on a downloaded file if opened within the browser. The user needs to open the PDF in a standalone reader like Adobe Acrobat. This is a user behavior you'd need to account for or document.

Some PDF readers on the market are more restrictive than Adobe about following local file links, so results could vary.

The bigger picture for Boppers

This is actually a clever hybrid strategy — your app lives on the server, generates portable PDF task lists, and those lists can bridge the gap between web resources and local files when used properly. It's worth prototyping. The main thing I'd suggest is testing it end-to-end with Adobe Acrobat specifically as your target reader, since that's the most reliable for this use case.

We've been through some of this before, but I want to review it. My understanding is that I could create a bottomless app on my local machine that would operate pretty much identically to the one on the server. The big difference is that the local file links using the file: convention in the bob list local app on my Windows machine would work right away not I wouldn't have to copy the file or anything. Now this brings up to important things. If I use the same file format on both the server and the local machine then these data files are basically portable between the server block list app and the local machine block list app similarly I could upload the local bop list, state of files to the server, using some sort of FTP and they would work there, at least the URL part the web URL part Woodwork.

Feb 18

Yes, you've got this exactly right, and it's an elegant architecture — like a wombat that has two burrows (one local, one on the server) but uses the same tunnel-digging style for both.

The core insight

A local Boppers instance and a server Boppers instance can share the same data file format. The app logic is identical. The only thing that differs is what kinds of

links are *functional* in each context. Web URLs work everywhere. File:// links only work locally. But crucially, having a non-functional file:// link in a server-hosted list doesn't break anything — it just sits there inert until the file is opened locally.

The portability picture

This means your data files become genuinely portable. You could work on your list locally, adding both web links and local file links. Upload the data file to the server via FTP. The server app displays everything faithfully — web links work, file links are visible but non-functional in that context. Download back to local when you want full functionality again.

The two-burrow strategy

This also opens up a natural workflow division. Local Boppers is your full-power environment where everything works including local media and documents. Server Boppers is your shareable, anywhere-accessible version where web links shine. Same data, same format, two contexts.

What this suggests for design

It would be worth making sure your data format stores links as plain text (the raw URL or file path) without baking in assumptions about which type they are. That way the app can simply try to open whatever is there and let the OS or browser handle it, succeeding or failing gracefully depending on context.

Have you settled on a data format for the Boppers lists yet — JSON, plain text, something else?

CLAUDE CLI is using Jason files to store the links on the program on the server. I think it was an earlier discussion you said that I could work over the existing code that works on the server with HTML, PHP, CSS etc. and we wanted from a browser on my local machine. That is, I could move the code to my Windows hard drive, and with the proper context and son of programs, running pretty much unchanged on the local Windows machine and have the same functionality of entering data and having a stored in the Jason files as I have on the server machine. I believe you said the main disadvantage is the file that would do this, including whatever supported means would be rather large taking up five or 10 minutes for something like that or as if I use simply a python program dedicated to generating the same file format it might be much smaller. Can you comment on This.

Feb 18

Yes, that's a good summary of the earlier discussion. Let me sharpen the picture.

Running the server code locally

You're right that the existing HTML/PHP/CSS code could run on your Windows machine essentially unchanged. To do it you'd need a local web server stack — something like XAMPP or WAMP, which bundles Apache, PHP, and related tools together. These are well-established packages but they are indeed chunky, probably a few hundred megabytes to install, and they turn your Windows machine into a mini web server just to run your app locally. It works, but it's like bringing a whole wombat colony just to dig one burrow.

The lightweight Python alternative

A small Python script (or Flask app) that serves the same HTML interface and reads/writes the same JSON files would be much leaner. Python is probably already on your machine or is a modest install, and a simple local server could be just a few kilobytes of code. You'd run it, open your browser to something like localhost:5000, and have essentially the same experience. Same JSON data format, same portability.

The practical recommendation

Given that your data is already JSON and your goal is local functionality plus portability to the server, the Python lightweight server approach is the more elegant wombat burrow. It's smaller, easier to understand, and doesn't carry all the overhead of a full XAMPP stack.

The main question is whether your existing PHP logic is complex enough that rewriting it in Python would be significant work, or whether it's simple enough that a Python equivalent would be quick to build. How much PHP logic does the server version currently have?

I think I understand, but I see a few other things to add. I already have the Apache stuff on my local computer that would allow me to do web development as if I were on a server. I assume I wouldn't have to put in another dedicated installation just to support the stuff I bring over from the server. I assume there is some sort of command structural code that will allow me to use my existing Apache installation for the new code I bring over from the server.

Feb 18

Exactly right — no new installation needed. If Apache (and PHP) are already running on your Windows machine, you just drop your server code into the appropriate folder (typically the htdocs folder under your Apache installation, or a virtual host you configure) and it will serve it just like your server does. Same wombat burrow, new tunnel added to it.

The typical steps would be something like: copy your PHP/HTML/CSS files into a subfolder under htdocs (say htdocs/boppers), make sure your JSON data files are

in a location the PHP code can read and write, then open your browser to localhost/boppers and you're running.

The main things to verify are that your PHP version locally matches closely enough to what's on the server that nothing behaves differently, and that any file paths your PHP code uses are adjusted for Windows conventions if they were hardcoded for Linux. Relative paths should be fine, but anything like /var/www/html/... would need updating to match your local structure.

Do you know offhand what you're using locally — is it XAMPP, WAMP, or something else? That would help pinpoint exactly where your htdocs equivalent lives and whether there's anything specific to watch out for.

This is all for future stuff. I have a number of things I've planned to do today that aren't anywhere near this. However, let me explore one more thing. I

understanding is that python can run on the server. Could I have a python program on the server that would be very similar to the python program on the Windows machine?

Feb 18

Yes, absolutely. Python runs fine on Linux servers, and a Flask-based Boppers app would look nearly identical on both the server and your local Windows machine. Same code, same JSON data format, same browser interface — just running in two different environments.

The main differences would be small housekeeping things: the file paths for where JSON data is stored would differ between Windows and Linux conventions, and on the server you'd typically run the Python app behind Apache using something called WSGI (a connector that lets Apache hand requests off to Python) rather than having Python serve directly. On your local machine you'd just run Flask directly and hit localhost.

But the core application code — reading and writing JSON, generating the HTML interface, handling links — would be essentially the same wombat in two different burrows. This is actually one of Python's strengths, that it's quite portable across operating systems.

This could be an interesting middle path for Boppers: write it once in Python, deploy it both locally and on the server, share the same JSON data format, and get the best of both worlds. Worth keeping in the back of your mind as the project evolves.