# MATLAB® Compiler™

## Web Apps

MATLAB®

R2020b

MathWorks®

# How to Contact MathWorks

| | |
|---|---|
| Latest news: | `www.mathworks.com` |
| Sales and services: | `www.mathworks.com/sales_and_services` |
| User community: | `www.mathworks.com/matlabcentral` |
| Technical support: | `www.mathworks.com/support/contact_us` |
| Phone: | 508-647-7000 |

The MathWorks, Inc.
1 Apple Hill Drive
Natick, MA 01760-2098

*MATLAB® Compiler™ Web Apps*

**Revision History**

| | | |
|---|---|---|
| March 2018 | Online only | New for Version 6.6 (Release 2018a) |
| September 2018 | Online only | Revised for Version 7.0 (Release R2018b) |
| March 2019 | Online only | Revised for Version 7.0.1 (Release R2019a) |
| September 2019 | Online only | Revised for Version 7.1 (Release R2019b) |
| March 2020 | Online only | Revised for Version 8.0 (Release R2020a) |
| September 2020 | Online only | Revised for Version 8.1 (Release R2020b) |

# Contents

# System Requirements

## System Requirements for Development Version of MATLAB Web App Server

System requirements for the development version of MATLAB Web App Server™ (Available in MATLAB Compiler™).

### Supported Platforms

Windows®, Linux®, macOS

### Hardware Requirements

The server can be installed on all hardware platforms and operating systems that MATLAB supports. The hardware requirements are:

- Minimum 60 GB of disk capacity to accommodate the server software installation and log files.
- Minimum 1GB of RAM per worker.

  The server only supports a maximum of 32 workers.
- Allocation of 1 processor core (or virtual core) per 4 workers; 2 cores minimum.

### Software Requirements

- An installation of MATLAB Runtime. The MATLAB Runtime version number must match the version of MATLAB you used to package the web app archive (`.ctf` file).
- MATLAB Compiler is required to package MATLAB apps as web app archives (`.ctf` files) to run on the server.
- MATLAB App Designer.

  .

### See Also

### More About

- “Install the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-2
- “Create and Run a Simple App Using App Designer”

# MATLAB Web App Server Differences

## MATLAB Web App Server Differences

MathWorks® has two offerings of MATLAB Web App Server:

- A standalone MATLAB Web App Server product.
- A development version of MATLAB Web App Server included in MATLAB Compiler.

The differences between the standalone MATLAB Web App Server product and the development version of MATLAB Web App Server in MATLAB Compiler are listed in the following table.

| **Supported Functionality** | **MATLAB Web App Server Product** | **Development Version of MATLAB Web App Server in MATLAB Compiler** |
|---|---|---|
| Authentication | ✓ | ✗ |
| Support for multiple releases of MATLAB | ✓ | ✗ |
| Number of end-users | Unlimited [a] | 32 |
| Server setup and configuration | Command-line only | Graphical user-interface |

a. Subject to server hardware limitations.

### See Also

### More About

- “System Requirements for Development Version of MATLAB Web App Server” on page 1-2
- “Install the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-2

# Install and Configure MATLAB Web App Server

## Install the Development Version of MATLAB Web App Server in MATLAB Compiler

**Warning** The development version of MATLAB Web App Server must be installed in a trusted intranet environment on dedicated hardware. The only purpose of the physical or virtual machine where the server is installed must be to host web apps that connect to the server. The server must never be exposed to the open Internet. For more information, see “MATLAB Web App Server Security” on page 5-3.

The development version of MATLAB Web App Server hosts web apps packaged using the **Web App Compiler** app. For web apps to work the server must be installed and configured. The server mediates the HTTP/HTTPS communication between the client web browser and the packaged MATLAB web app. It has a home page listing all the available hosted web apps. The home page can be accessed from a browser using a URL.

You can use an online installation workflow to install the server if your computer is connected to the Internet. Otherwise, follow the steps in the limited internet connectivity installation workflow.

### Online Installation

1 Navigate to the folder containing the MATLAB Web App Server installer.

| Platform | Default Location of Installer |
|---|---|
| Windows | `C:\Program Files\MATLAB\R2020b\toolbox\compiler\deploy\win64\MATLABWebAppServerSetup` |
| Linux | `/usr/local/MATLAB/R2020b/toolbox/compiler/deploy/glnxa64/MATLABWebAppServerSetup` |
| macOS | `/Applications/MATLAB_R2020b.app/toolbox/compiler/deploy/maci64/MATLABWebAppServerSetup` |

2 Run the installer to install the MATLAB Web App Server and MATLAB Runtime.

| Platform | Steps |
|---|---|
| Windows *(Administrator)* | Right-click `MATLABWebAppServerSetup.exe`, and select **Run as administrator**. |
| Linux | At the shell prompt, type: `./MATLABWebAppServerSetup.install` **Note** On Debian® based Linux distributions, type: `gksudo ./MATLABWebAppServerSetup.install` |
| macOS | Double-click the app `MATLABWebAppServerSetup.app` |

For information on how to start and configure the server, see “Configure the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-5.

**Note** **Windows** : Make sure that MATLAB Runtime appears before any installed version of MATLAB in the Windows `Path` environment variable.

## Limited Internet Connectivity Installation

If the computer where you want to run the development version of MATLAB Web App Server is not connected to the Internet, installing the server is a two part process.

First, you need to download MATLAB Runtime from a computer that is connected to the Internet. After downloading MATLAB Runtime installer, you will need to transfer the installer to the computer that is not connected to the Internet. MATLAB Runtime is a standalone set of shared libraries that enables the execution of compiled MATLAB applications or components on computers that do not have MATLAB installed.

Second, you need to copy the zip file containing the MATLAB Web App Server application from an installation of the MATLAB Compiler product and unzip the server application to a local folder on the computer that is not connected to the Internet.

### Procedure

1. Download the MATLAB Runtime installer from the MathWorks website or the MATLAB desktop. You will need a computer with access to the Internet to complete this step.

| Option | Steps |
|---|---|
| MathWorks Website | Select the appropriate platform and release specific installer from:<br><br>https://www.mathworks.com/products/compiler/matlab-runtime.html |
| MATLAB Desktop | At the MATLAB command prompt, type:<br><br>`compiler.runtime.download` |

   Transfer the installer to the computer that is not connected to the Internet.
2. Install MATLAB Runtime using the installer. For installation instructions, see “Install and Configure the MATLAB Runtime”.
3. On a computer that has the MATLAB Compiler product installed, navigate to the folder containing the file `MATLABWebAppServer.zip` and copy it to a local folder on the computer that is not connected to the Internet.

| Platform | Default Location |
|---|---|
| Windows | `C:\Program Files\MATLAB\R2020b\toolbox\compiler\deploy\win64\MATLABWebAppServerSetup\offline` |
| Linux | `/usr/local/MATLAB/R2020b/toolbox/compiler/deploy/glnxa64/MATLABWebAppServerSetup/offline` |
| macOS | `/Applications/MATLAB_R2020b.app/toolbox/compiler/deploy/maci64/MATLABWebAppServerSetup/offline` |

4. Unzip/Extract the copied zip file `MATLABWebAppServer.zip`. You can now access the MATLAB Web App Server application.

For information on how to start and configure the server, see “Configure the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-5.

---

**Note** **Windows** : Make sure that MATLAB Runtime appears before any installed version of MATLAB in the Windows `Path` environment variable.

---

## See Also

## More About

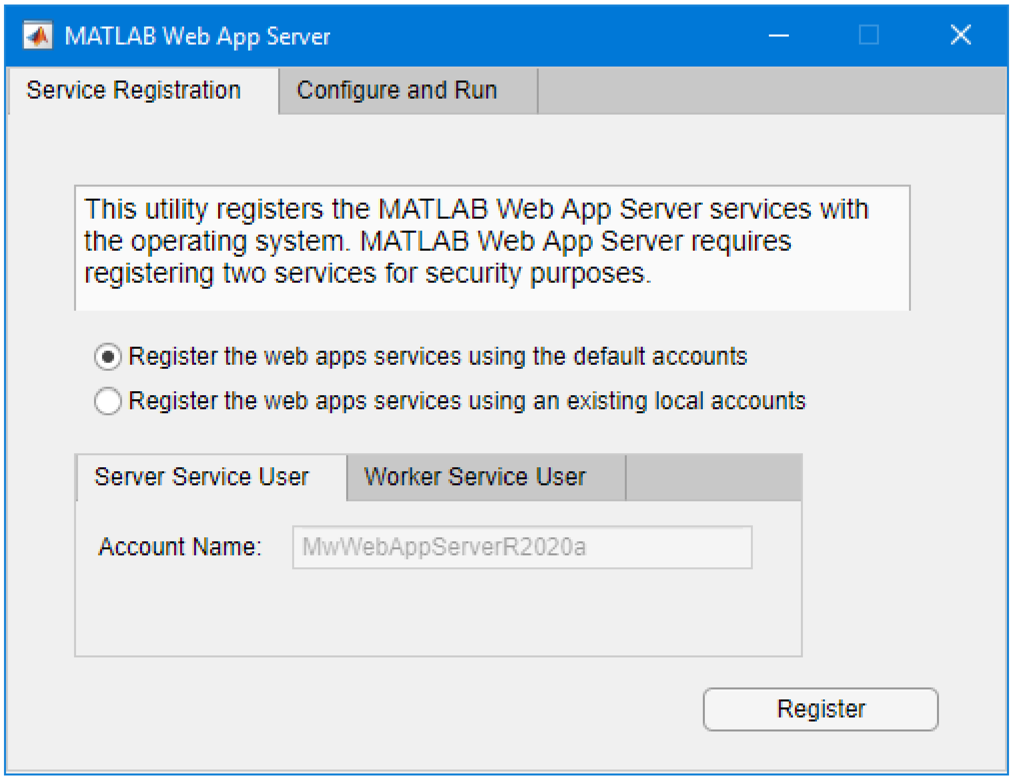
- “Configure the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-5
- “MATLAB Web App Server Security” on page 5-3
- “Potential Risks” on page 5-4

## Configure the Development Version of MATLAB Web App Server in MATLAB Compiler

To configure the development version of MATLAB Web App Server:

1. Navigate to the development version of MATLAB Web App Server installation location and run the server application. For information about installing the server, see "Install the Development Version of MATLAB Web App Server in MATLAB Compiler" on page 3-2.

| Platform | Steps |
|---|---|
| Windows<br><br>*(Administrator)* | Default location of the server application is:<br><br>`C:\Program Files\MATLAB\MATLABWebAppServer\R2020b\application`<br><br>Right-click `MATLABWebAppServer.exe` and select **Run as administrator**. |
| Linux | Default location of the server application is:<br><br>`/usr/local/MATLAB/MATLABWebAppServer/R2020b/application`<br><br>At the shell prompt, type:<br><br>`run_MATLABWebAppServer.sh <MATLAB_Runtime_Location>`<br><br>*For example:*<br><br>`run_MATLABWebAppServer.sh /usr/local/MATLAB/MATLAB_Runtime/v99` |
| macOS | Default location of the server application is:<br><br>`/Applications/MATLAB/MATLABWebAppServer/R2020b/application`<br><br>Double-click the app `MATLABWebAppServer.app` |


MATLAB Web App Server

Service Registration | Configure and Run

This utility registers the MATLAB Web App Server services with the operating system. MATLAB Web App Server requires registering two services for security purposes.

[●] Register the web apps services using the default accounts
[ ] Register the web apps services using an existing local accounts

Server Service User | Worker Service User

Account Name: MwWebAppServerR2020a

Register

2 Click the **Service Registration** tab, and select one of two options:

- **Register the web apps services using the default accounts**

  Selecting this option registers two services:

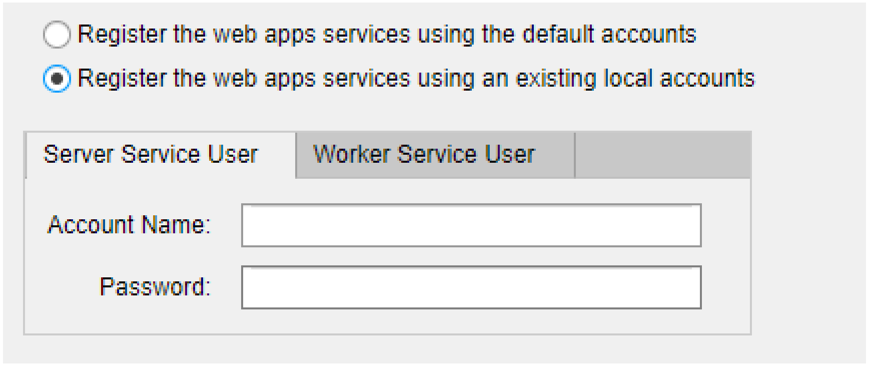
  - A service to run the server represented by the tab **Server Service User**.
  - A service to run the apps represented by the tab **Worker Service User**.

| Operating System | Server Service Information | Apps Service Information |
|---|---|---|
| Windows | **Account Name:** `MwWebAppServerR2020b`<br><br>**Service Name:** `mw-webapps-R2020b` | **Account Name:** `MwWebAppWorkerR2020b`<br><br>**Service Name:** `mw-webapps-launcher-R2020b` |

| Operating System | Server Service Information | Apps Service Information |
|---|---|---|
| Linux | **Account Name:** `MwWebAppsServerR2020b`<br><br>**Service Name:** `mw-webapps-R2020a`<br><br>**Service File:** `/etc/systemd/system/mw-webapps-R2020b.service` | **Account Name:** `MwWebAppsWorkerR2020b`<br><br>**Service Name:** `mw-webapps-launcher-R2020b`<br><br>**Service File:** `/etc/systemd/system/mw-webapps-launcher-R2020b.service` |
| macOS | **Account Name:** `MwWebAppsServerR2020b`<br><br>**Service Name:** `com.mathworks.mw-webapps-R2020b`<br><br>**Service File:** `/Library/LaunchDaemons/com.mathworks.mw-webapps-R2020b.plist` | **Account Name:** `MwWebAppsWorkerR2020b`<br><br>**Service Name:** `com.mathworks.mw-webapps-launcher-R2020b`<br><br>**Service File:** `/Library/LaunchDaemons/com.mathworks.mw-webapps-launcher-R2020b.plist` |

- **Register the web apps services using existing local accounts**

  Selecting this option requires you to register two services represented by the tabs **Server Service User** and **Worker Service User** using two local account names and passwords.



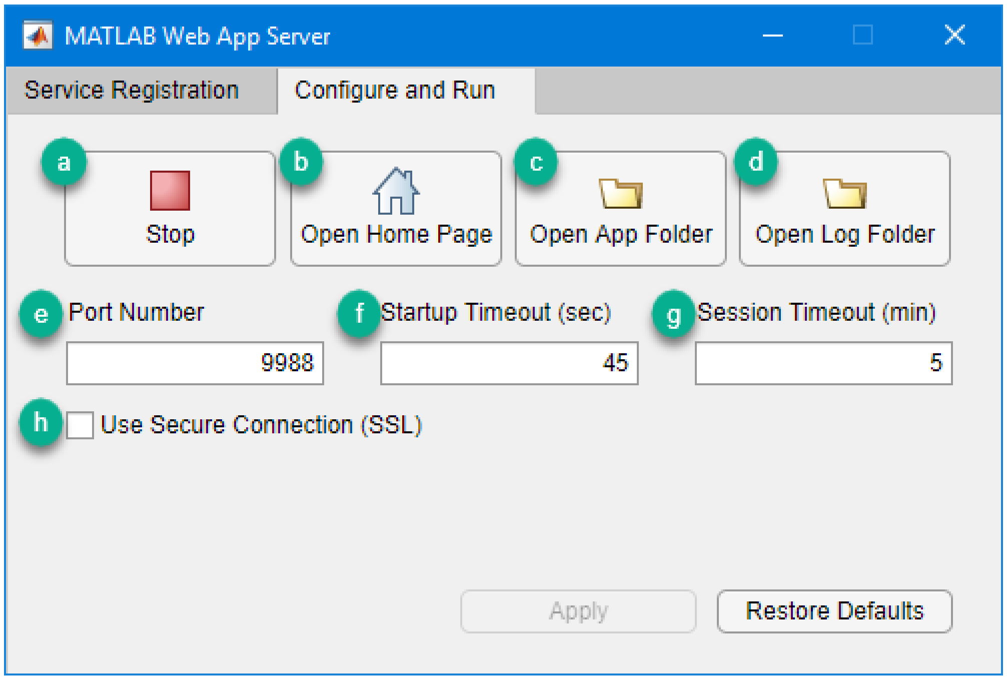
Register the web apps services using the default accounts [✓ Register the web apps services using an existing local accounts]

Server Service User | Worker Service User

Account Name:

Password:

3 You can start or stop the installed service from the **Configure and Run** tab. The service is automatically started once the service is successfully registered. If the service does not start automatically, click **Start**.



| | |
|---|---|
| **a** | Start/Stop the server. |
| **b** | Open the web apps home page. |
| **c** | Open the app folder containing the web app archive (`.ctf`) files. |
| **d** | Open the server log folder. |
| **e** | Specify the port number. Default port is `9988`. |
| **f** | **Startup Timeout (sec)**: Defines the maximum time to prepare a new session for the requested app, in seconds. If server is under heavy load or hardware is not adequate, you may need to tweak this setting. Default value is `45` seconds. |
| **g** | **Session Timeout (min)**: Defines the time interval in minutes after which session will be stopped on the server. When session is running and browser tab is opened, periodic heartbeat messages will let server know that user is still interested in the session. If session was abandoned for reasons like computer going to sleep or network disconnected, this interval will define for how long the session will be kept alive on the server. Default value is `5` minutes. |
| **h** | Enable SSL and use HTTPS. For more information, see "Enabling HTTPS" on page 5-2. |

4. Once you have installed and started the server, click the **Open App Folder** button to open the folder where compiled web apps are served from. Then serve your web apps by dropping their *`appName`*`.ctf` files in this folder. If the path specified for **App Path** does not exist, it is created after you start the service.
5. To open your web apps home page, click **Open Home Page**.

   You can also configure the service by specifying:

   - the port number used for your web apps
   - timeout values for the session and for startup time

   After making any changes, click **Apply** to save your configuration.

6. To stop the service, go to the **Configure and Run** tab, and click **Stop**.
7. To unregister the service, go to the **Service Registration** tab, and click **Unregister**.

---

**Note**

1. The maximum number of sessions that the server supports is limited to 32. However, the amount of RAM on the machine may limit you to fewer than 32 sessions.
2. The server limits the number of MATLAB Compiler licensed users who can upload and run web apps to 10.

---

## See Also

## More About

- "Install the Development Version of MATLAB Web App Server in MATLAB Compiler" on page 3-2
- "Enabling HTTPS" on page 5-2
- "MATLAB Web App Server Security" on page 5-3

## Supported Browsers and Platform Incompatibilities

### Supported Browsers

Web apps are compatible with commonly used browsers like Google Chrome™, Safari, Firefox®, Microsoft Edge®.

For browsers that update automatically, the current stable version is supported.

Web apps are also supported on the Safari browser on iPads, and the Chrome™ browser on Chromebooks.

### Platform Incompatibilities

MATLAB Web App Server cannot be installed on Red Hat® Enterprise Linux 6.

Linux systems require the `systemd` software suite.

### See Also

### More About

- "Install the Development Version of MATLAB Web App Server in MATLAB Compiler" on page 3-2
- "Configure the Development Version of MATLAB Web App Server in MATLAB Compiler" on page 3-5

## Start the Server Application

To start the server application in the development version of MATLAB Web App Server follow the platform specific instructions.

| Platform | Steps |
|---|---|
| Windows<br><br>*(Administrato r)* | Default location of the server application is:<br><br>`C:\Program Files\MATLAB\MATLABWebAppServer\R2020b\application`<br><br>Right-click `MATLABWebAppServer.exe` and select **Run as administrator**. |
| Linux | Default location of the server application is:<br><br>`/usr/local/MATLAB/MATLABWebAppServer/R2020b/application`<br><br>At the shell prompt, type:<br><br>`run_MATLABWebAppServer.sh <MATLAB_Runtime_Location>`<br><br>*For example:*<br><br>`run_MATLABWebAppServer.sh /usr/local/MATLAB/MATLAB_Runtime/v99` |
| macOS | Default location of the server application is:<br><br>`/Applications/MATLAB/MATLABWebAppServer/R2020b/application`<br><br>Double-click the app `MATLABWebAppServer.app` |

### See Also

### More About

- “Configure the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-5
- “Install the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-2

# Create, Run, and Diagnose Web Apps

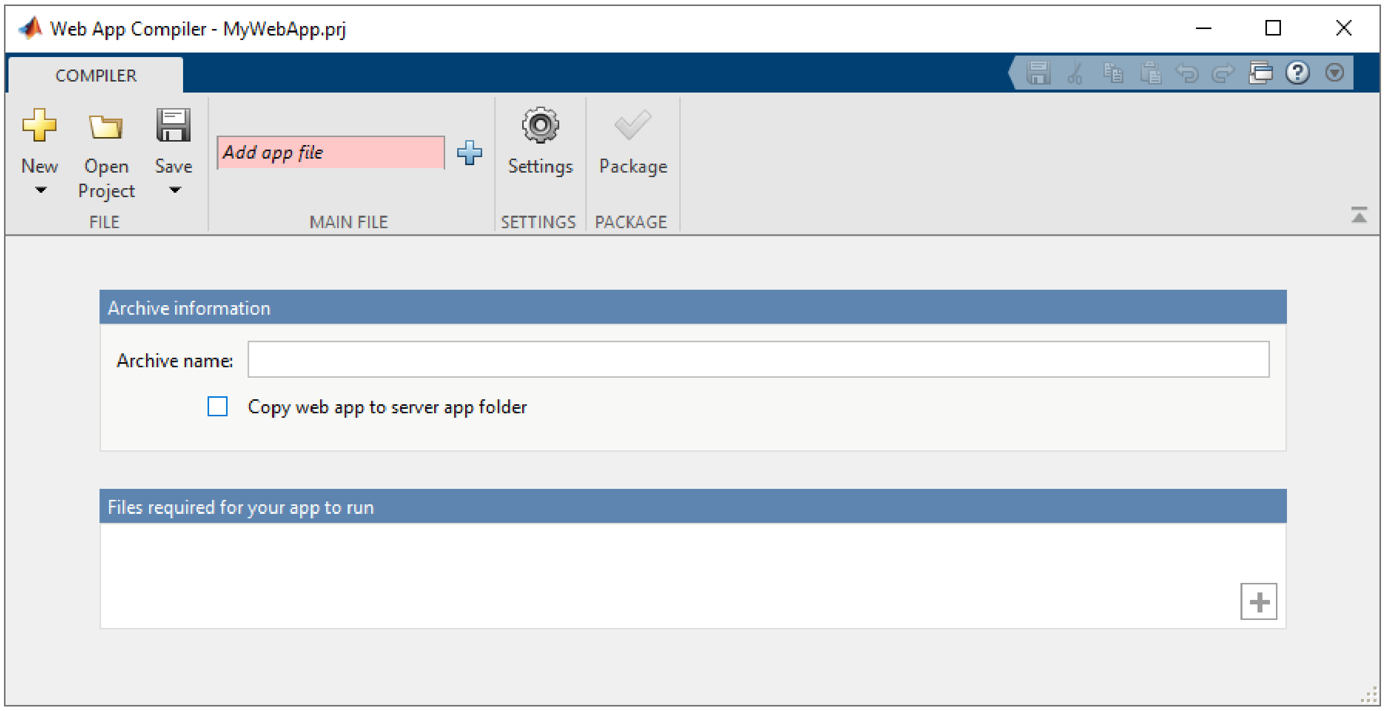
# Create Web App

## App Designer Prerequisite

Before you can package and then deploy a web app, you need to create an app using MATLAB App Designer. For more information, see “Create and Run a Simple App Using App Designer”.

## Steps to Package and Create a Web App

1 Type `webAppCompiler` at the MATLAB command line to open the **Web App Compiler** app.



2 In the **MAIN FILE** section of the toolstrip, click the button to add your App Designer `.mlapp` file to the project. The Web App Compiler automatically resizes to include an **App details** section that contains information about the app such as app name, author, summary, description, and version. You can edit information about the app in App Designer by clicking **Edit App Details**. Click **Refresh** to update Web App Compiler with any changes you have made.

3 *(Optional)* Select **Copy web app to server app folder** check box and specify the path to the app folder on the server where you want the web app archive (`.ctf` file) to be automatically copied. If you leave this check box cleared, the Web App Compiler will create the web app archive (`.ctf` file) in the project folder. You must manually copy or upload the web app archive (`.ctf` file) to the app folder on the server. For more information, see “Configure the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-5.

4 Add supporting files, if any, in the **Files required for your app to run** section. Supporting files include any MAT-files, images used by your web app, or user-written MATLAB functions not found by MATLAB Compiler.

5 Click **Package** to package the app, and create a web app archive (`.ctf` file).

In the **Save Project** dialog box that opens, specify a project name and a location where you want to save the web app project. Web App Compiler saves your project and opens a **Package** dialog box.

6. Once packaging is complete, in the **Package** dialog box, click **Open output folder**. This step opens the project folder which contains the following files:

   - `*webAppArchiveName*.ctf`
   - `mccExcludedFiles.log`
   - `PackagingLog.html`
   - `requiredMCRProducts.txt`

   You can view the log file, `PackagingLog.html`, to see the exact `mcc` syntax used to package and create the web app archive.
7. To use the web app, deploy the web app archive file, *`webAppArchiveName`*`.ctf`. For more information, see “Deploy Web App” on page 4-4.

## See Also

### More About

- “Limitations and Unsupported Functionality” on page 4-9
- “Deploy Web App” on page 4-4
- “Run Web App” on page 4-6
- “Simple Mortgage Calculator Web App” on page 8-2

## Deploy Web App

You can deploy web apps to the development version of MATLAB Web App Server in one of two ways:

- Copy the web app archive (`.ctf` file) generated by Web App Compiler (in MATLAB Compiler) to the apps folder configured by the server.
- Upload the web app archive (`.ctf` file) generated by Web App Compiler (in MATLAB Compiler) to the apps folder configured by the server. This deployment option is available only if you have the MATLAB Web App Server product installed. For more information, see “Install or Uninstall MATLAB Web App Server Product” (MATLAB Web App Server).

### Deploy a Web App by Copying

1. Navigate to the project folder generated by Web App Compiler (in MATLAB Compiler) during the packing process.
2. Copy the file `*webAppArchiveName*.ctf` to the app folder configured by the MATLAB Web App Server. If you selected the **Copy web app to server app folder** check box in the Web App Compiler while creating the web app, your web app archive (`.ctf` file) is automatically be copied to the app folder configured by the server.

   You can access the app folder by clicking the **Open App Folder** button in the MATLAB Web App Server utility.

   **Note** You must have write permissions to the app folder to copy a web app archive (`.ctf` file) to folder.

   Your web app is now deployed and can be accessed from the web apps home page. For more information, see “Run Web App” on page 4-6.

### Deploy a Web App by Uploading (Available Only with MATLAB Web App Server Product)

#### Prerequisites

- Verify that you have the MATLAB Web App Server product installed. For more information, see “Install or Uninstall MATLAB Web App Server Product” (MATLAB Web App Server).
- Verify that “Authentication” (MATLAB Web App Server) and “Role-Based Access” (MATLAB Web App Server) are enabled on the server.
- Verify that you are designated as an *author* while configuring “Role-Based Access” (MATLAB Web App Server).

#### Procedure

1. Navigate to the web apps home page configured by the server. You can obtain the URL to the home page by executing `webapps-status` at the system command-line or by getting the URL from an individual who is administering the server. The format of the home page URL is:

   ```
   http://webAppServer:PortNumber/webapps/home/index.html
   ```

Here, *`webAppServer`* is your web app server hostname, and *`PortNumber`* is the port specified when configuring the server.

2. Click Manage Apps on the web apps home page to open the *Manage Apps* page.
3. Click Upload App and navigate to the project folder generated by Web App Compiler (in MATLAB Compiler) during the packaging process.
4. Select the file *`webAppArchiveName`*`.ctf` and click **Open** to upload the web app. You get a confirmation that the *`webAppArchiveName`*`.ctf` was successfully uploaded to the server.

## See Also

## More About

- “Run Web App” on page 4-6
- “Create Web App” on page 4-2

## Run Web App

**Note** To run a web app, you must also have the development version of MATLAB Web App Server running. For more information, see “Install the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-2 and “Configure the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-5.

Using a web browser, you can access web apps running on the server by navigating to the web apps home page.

You can obtain the URL to the home page by clicking **Open Home Page** in the server application in the development version of MATLAB Web App Server. The format of the home page URL is:

```
http://webAppServer:PortNumber/webapps/home/index.html
```

Here, *`webAppServer`* is your web app server hostname, and *`PortNumber`* is the port specified when configuring the server.

1. To run a web app, click the web app tile on the home page.
2. To see a list of all web apps with their status and diagnostic messages, click the **Diagnostics** link on the top-right corner of the home page.
3. To go back to the home page of your web apps, click **MATLAB Web Apps** on the breadcrumb trail at the top of the page.

### See Also

### More About

- “Create Web App” on page 4-2
- “Deploy Web App” on page 4-4

## Diagnostics

Click **Diagnostics** on the web apps home page to get information about each web app.

When you click **Diagnostics**, you see a page with all the web apps deployed to the server along with their status. The various statuses are:

- **OK**—The web app is available to run.
- **Corrupt CTF**—The deployed web app archive file cannot be read.
- **Required Runtime Unavailable**—The web app uses a different version of MATLAB Runtime than the one on the server.
- **Not a Web App**—The deployed `.ctf` file is not a web apps `.ctf` file.

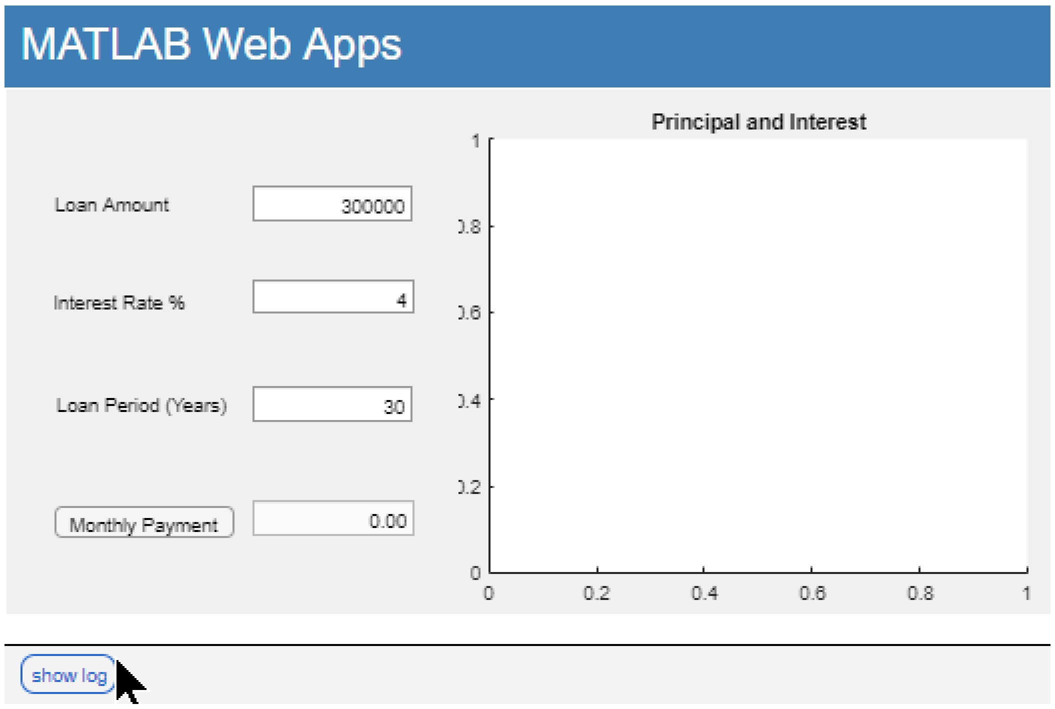
### See Also

### More About

- “Web App Session Log” on page 4-8
- “Run Web App” on page 4-6

## Web App Session Log

The web app session log captures all interaction associated with each web app. If you encounter any issues, you can check the log for more information. You can pass this information to the server administrator for troubleshooting. You can access the log by clicking the **show log** link at the bottom left corner of the browser.



**Show Log**

### See Also

### More About

- “Diagnostics” on page 4-7

## Limitations and Unsupported Functionality

When packaging a MATLAB app into a web app consider the following functional limitations. Using certain functions may result in an error or unexpected behavior.

- Multiwindow apps
  - Multiple calls to `figure` or `uifigure` are not supported.
  - Dialog boxes: `dialog`, `msgbox`, `errordlg`, `warndlg`, `helpdlg`, `listdlg`, `questdlg`, `inputdlg`, `uisetcolor`, and `uisetfont` are not supported. However, `uialert`, `uiconfirm`, and `uiprogressdlg` are supported.
- Although you can upload and download files from a local system in a deployed web app using `uigetfile` and `uiputfile`, you cannot download files while an external application is writing them. Opening a folder selection dialog box on the client using `uigetdir` is not supported.
- Printing: `print`, `printpreview` are not supported.
- Graphics root object properties: `MonitorPositions`, `PointerLocation`, `ScreenDepth`, `ScreenPixelsPerInch`, `ScreenSize` are not supported.
- Figure properties: `CloseRequestFcn`, `Position`, `InnerPosition`, `Resize`, `Visible`, `WindowState`, `WindowStyle` have no effect on a web app.
- System Commands: The system commands `computer`, `ispc`, `isunix`, `ismac`, and `listfonts` will execute but they will not return client-side state information. Only server-side state information is returned.
- Tooltips: Data tips for graphics are not supported.
- Axes toolbar interactions: Data brushing is not supported.
- Plot functions: The `wordcloud` function is not supported.

These limitations are in addition to App Designer graphics limitations. For more information, see “Display Graphics in App Designer”.

### See Also

### More About

- “Create Web App” on page 4-2
- “Authoring Secure Web Apps” on page 5-6

5

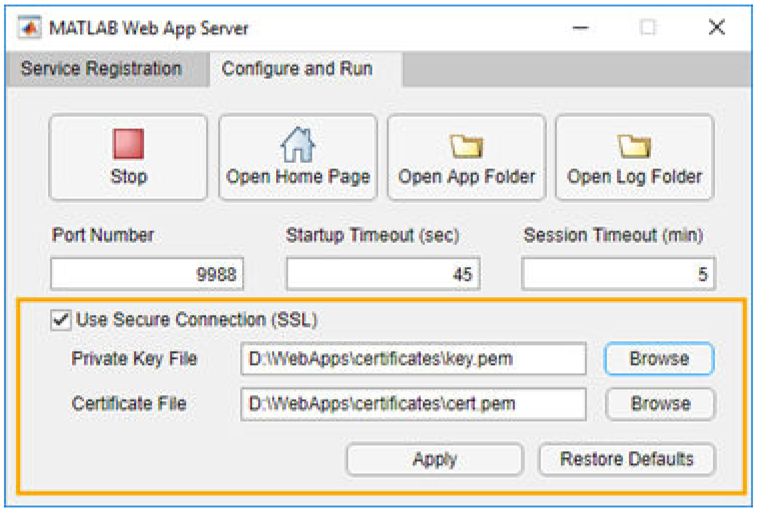
# Security

## Enabling HTTPS

1. Navigate to the MATLAB Web App Server installation location and run the server application.

| Platform | Steps |
|---|---|
| Windows<br><br>*(Administrator)* | Default location of the application is:<br><br>`C:\Program Files\MATLAB\MATLABWebAppServer\R2020b\application`<br><br>Right-click `MATLABWebAppServer.exe` and select **Run as administrator**. |
| Linux | Default location of the application is:<br><br>`/usr/local/MATLAB/MATLABWebAppServer/R2020b/application`<br><br>At the shell prompt, type:<br><br>`run_MATLABWebAppServer.sh <MATLAB_Runtime_Location>` |
| macOS | Default location of the application is:<br><br>`/Applications/MATLAB/MATLABWebAppServer/R2020b/application`<br><br>Double-click the app `MATLABWebAppServer.app` |

2. Select the **Configure and Run** tab.



Port Number: 9988
Startup Timeout (sec): 45
Session Timeout (min): 5
[✓] Use Secure Connection (SSL)
Private Key File: D:\WebApps\certificates\key.pem
Certificate File: D:\WebApps\certificates\cert.pem

3. Check the option **Use Secure Connection (SSL)** in the MATLAB Web App Server.
4. Enter the location of the private key file and the certificate file. Click **Apply**.

The browser opens the web apps home page using HTTPS.

## See Also

## MATLAB Web App Server Security

**Caution** It is strongly recommended that you consult with your IT system administrator and discuss the security implications of installing the development version of MATLAB Web App Server.

Installing and running the server on your network exposes your network and file system to risks. The machine running the server is most at risk from accidental or deliberate misuse of deployed web applications. Therefore, you must install the server software only on dedicated hardware. This machine can be a physical or virtual machine whose only purpose is to host web applications that connect to the server software. Using a physical or virtual machine limits the risk in the event that the machine is compromised.

Setting up of the development version of MATLAB Web App Server creates two low-privileged user accounts on the host machine—one for the server and one for applications. However, you can choose to use the same account. However, using the same account can introduce additional risks. In addition, through a process known as *privilege escalation*, attackers may be able to exploit bugs in the operating system or network to obtain the privileges of ordinary or even administrative users. They can then attempt to access files or other intellectual property without permission.

The development version of the server relies on the authentication and authorization scheme of its host machine and network. Other than supporting HTTPS, it does not contain any additional mechanisms for authenticating or authorizing web application users. For more information, see “Enabling HTTPS” on page 5-2.

You may be able to mitigate some of these risks by taking these precautions:

- *Restrict network access.* Only trusted users can access the server and its associated applications.
- *Execute only trusted applications.* Trust applications developed by only well-known, trusted, and authenticated sources.
- *Limit application functionality.* Include in the application only those features of MATLAB required for the application to perform its function. For more information, see “Authoring Secure Web Apps” on page 5-6.

For a list of additional risks, see “Potential Risks” on page 5-4.

### See Also

### More About

- “Potential Risks” on page 5-4
- “Enabling HTTPS” on page 5-2
- “Authoring Secure Web Apps” on page 5-6
- “Securely Deploying Web Apps” on page 5-8

## Potential Risks

- The MATLAB Web App Server has no specific mechanism to prevent HTTP request capture and replay.
- The development version of MATLAB Web App Server has no mechanism for authentication or authorization other than HTTPS.

  Any user with access to the network can run any application created with this software and read any data the application is authorized to access.

  If you want authentication and role-based access capabilities, you need to purchase the MATLAB Web App Server product.
- Installation of the MATLAB Web App Server creates two low-privileged user accounts on the host machine.

  These low-privileged accounts may inherit privileges given to all users. Care should be taken to restrict privileges given to all users.
- While the server and applications run under two different low-privileged user accounts, all applications hosted by the server run under the same low-privileged user account.

  If multiple copies of the same application run simultaneously, they might interfere with each other. This situation happens if the application writes data to any shared resource, for example, a file or a non-concurrent database.
- When deploying multiple applications to the server, the server shares cookies across sessions, which can result in crosstalk between applications for a single user accessing more than one application.

  This situation could allow unintentional crosstalk between multiple applications run by the same user.
- Deployed web applications are potentially vulnerable to *data* or *code injection* attacks whereby malicious or malformed inputs can be used to attempt to subvert the system. The server does not contain explicit protection against either type of injection attack. Certain MATLAB features, particularly the `eval()` function, can increase the risk of injection attacks. A common countermeasure is input sanitization or input whitelisting. MATLAB contains functions like `regexp` and `regexprep` that can assist in validating untrusted input.

  - Your application may indirectly call `eval()`, potentially making it vulnerable to code-injection attacks.
  - Other MATLAB functions may exhibit the same code injection vulnerabilities; any function that processes code-like input (XML, SQL, JSON, to name a few) is potentially vulnerable to code injection.
  - Any application that accesses the operating system via MATLAB `system()`, `dos()`, or `unix()` commands might also be vulnerable to code injection.

**Note** This list identifies known risks and is not meant to be comprehensive.

## See Also

## More About

- “Enabling HTTPS” on page 5-2
- “Authoring Secure Web Apps” on page 5-6
- “Securely Deploying Web Apps” on page 5-8
- “MATLAB Web App Server Security” on page 5-3

## Authoring Secure Web Apps

Most of the potential risk associated with deploying web apps comes from the code in each app. By limiting the features that your app uses and by following the secure coding practices listed here, you can reduce the potential risk to your applications.

### Authentication and Authorization

If your app requires access to sensitive data or performs potentially dangerous actions, you can consider implementing your own authentication and authorization schemes. Consult your network security group for advice.

### Do Not Call eval()

The MATLAB `eval()` function turns text strings into commands. This powerful function allows users to execute arbitrary MATLAB code. This code can, in turn, enable execution of any installed program that is available to the low-privileged user or access to any file or data that the low-privileged users has access to. Applications created for web deployment and access must not contain calls to `eval()`. See “Alternatives to the eval Function” for ways to eliminate `eval()` from your web app code. Relying on input sanitization can help mitigate the risk of any indirect calls to `eval()`. See “Sanitize User Input” (MATLAB Web App Server).

### Limit Free-Text User Input

Use menus, sliders, dials, and buttons instead of editable text fields in your app user interface. In addition to providing a better user experience, this practice limits the types of input users can provide, and the risks such inputs might introduce.

### Sanitize User Input

To a security expert, user supplied data is considered untrusted because user input is a common attack vector for hackers. If your app must accept free-text input, the app must carefully examine the input for potential code injection attacks—text that contains special characters that coerce the app to interpret the input as commands rather than data.

In MATLAB, code injection attacks are most likely to be directed against XML, JSON, SQL, and other similar types of structured input. If your app accepts structured input, consult your IT or security group for suggestions on how to sanitize that input. It is never a good idea to allow the user to directly enter any type of code (such as MATLAB, Java®, or JavaScript®) for immediate evaluation.

### Sanitize Data Read from Files

Reading data from files exposes the app to the same types of risk as collecting interactive user input. The same countermeasures apply. Also, you can protect read-only data files from tampering by a cryptographically secure hashing algorithm to digitally fingerprint files.

### Minimize File System Write Operations

Limiting your application to read-only access greatly reduces the potential risk associated with your application. If you must write to the file system, remember that the server runs multiple copies of your application simultaneously if multiple users access it at the same time. You must manage simultaneous writes either through the use of runtime-generated unique file names or the use of a database that can typically handle multiple simultaneous accesses. If simultaneous writes are not properly managed, data corruption may occur.

## Verify Trustworthiness of Third-Party Code

If your app includes MATLAB files, shared libraries, Java classes, or any other type of code developed by a third party, you must make sure that code is free of viruses, worms, Trojan horses, and other web-based attack and infiltration vectors. You can discuss this issue with the author of the code and your IT and security staff. In the case of binaries or Java classes, consider running a virus scanner or other security software on the code before including it in your deployed app.

## Reduce Exposure Level

One way to reduce exposure is to limit the time that the app is running to only those times when it is needed. For example, do not run it continuously from your desktop.

## See Also

## More About

- “Potential Risks” on page 5-4
- “Enabling HTTPS” on page 5-2
- “Securely Deploying Web Apps” on page 5-8
- “MATLAB Web App Server Security” on page 5-3

## Securely Deploying Web Apps

- Install the MATLAB Web App Server on a dedicated physical or virtual machine, and do not use this machine for any other purpose.
- Run web apps behind your organization firewall. Do not allow access from the open Internet.
- Install web apps only from trusted and verified people and organizations.
- Limit the features and functionality you build into the web apps you develop.
  - Do not call the MATLAB function `eval()`.
  - Avoid free-text input where you can, and use menus, lists, buttons, and other affordances instead.
  - Sanitize input from the app user interface and data files.
  - Limit file, network, and other resource access to the minimum required by your app.
  - Verify the trustworthiness of any third-party code included in your app.
- If your application accesses sensitive data, use in-application authentication to limit access.
- Reduce exposure level by limiting the time that the app runs to only those times when it is needed. For example, do not run it 24 hours a day, 7 days a week from your desktop.

### See Also

### More About

- “Authoring Secure Web Apps” on page 5-6
- “Potential Risks” on page 5-4
- “Enabling HTTPS” on page 5-2
- “MATLAB Web App Server Security” on page 5-3

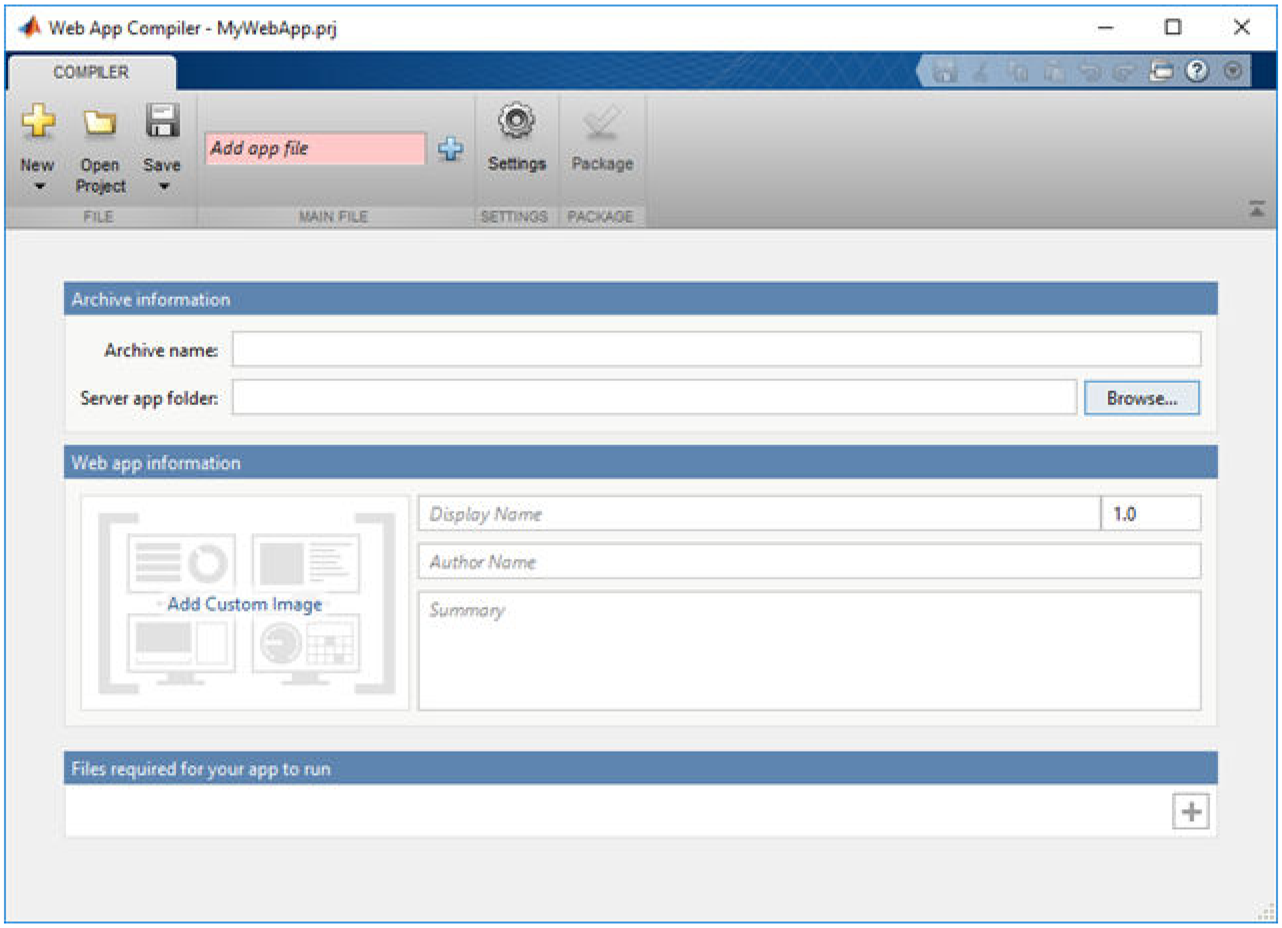
# Apps

# Web App Compiler

Package App Designer apps for web deployment

## Description

The **Web App Compiler** app lets you package MATLAB App Designer apps for web deployment. The app takes an App Designer `.mlapp` file and packages it as a web app archive `.ctf` file. You deploy the web app by copying the `.ctf` file to the apps folder in the MATLAB Web App Server. Once deployed, you can use the web app in a browser.



## Open the Web App Compiler App

- MATLAB Toolstrip: On the **Apps** tab, under **Application Deployment**, click the **Web App** icon.
- MATLAB command prompt: Enter `webAppCompiler`.
- App Designer: In App Designer toolstrip, click the **Share** button, and select **Web App**.

## Examples

- “Simple Mortgage Calculator Web App” on page 8-2

## Programmatic Use

`webAppCompiler`

## See Also

### Apps

### Functions

### Topics
“Simple Mortgage Calculator Web App” on page 8-2

**Introduced in R2018a**

# Functions

## compiler.build.webAppArchive

Create an archive for deployment to MATLAB Web App Server

### Syntax

```
compiler.build.webAppArchive(AppFile)
compiler.build.webAppArchive(AppFile,Name,Value)
compiler.build.webAppArchive(opts)
results = compiler.build.webAppArchive( ___ )
```

### Description

`compiler.build.webAppArchive(AppFile)` creates a web app archive using a MATLAB app specified by `AppFile`.

`compiler.build.webAppArchive(AppFile,Name,Value)` creates a web app archive with options specified as one or more name-value pairs. Options include the archive name, additional files to include, and the output directory.

`compiler.build.webAppArchive(opts)` creates a web app archive with options specified by a `compiler.build.WebAppArchiveOptions` object `opts`. You cannot specify any other options using name-value pairs.

`results = compiler.build.webAppArchive( ___ )` returns build information as a `compiler.build.Results` object using any of the input arguments in previous syntaxes. Build information includes the build type, the path to the compiled archive, and build options.

### Examples

#### Create Web App Archive

Create a web app archive on a Windows system from an existing MATLAB app named `example.mlapp`.

Build a web app using the `compiler.build.webAppArchive` command.

```
buildResults = compiler.build.webAppArchive('example.mlapp');
```

This generates the following files within a folder named `examplewebapp` in your current working directory:

- `example.ctf`—Deployable web app archive file.
- `mccExcludedFiles.log`—Log file that contains a list of any toolbox functions that were not included in the application. For more information on non-supported functions, see MATLAB Compiler Limitations.
- `requiredMCRProducts.txt`—Text file that contains product IDs of products required by MATLAB Runtime to run the application.

### Customize Web App Archive Using Name-Value Pairs

Customize a web app archive using name-value pairs on a Windows system to specify the archive name and output directory, and automatically include data files provided as inputs to functions such as `load`.

```
compiler.build.webAppArchive('example.mlapp',...
    'ArchiveName','MyWebApp',...
    'OutputDir','D:\Documents\MATLAB\work\WebApps',...
    'AutoDetectDataFiles','on')
```

### Customize Multiple Web App Archives Using Options Object

Customize multiple web app archives using a `compiler.build.WebAppArchiveOptions` object on a Windows system to specify a common output directory, include a MAT-file containing workspace variables, and enable build verbosity.

Create a `WebAppArchiveOptions` object.

```
opts = compiler.build.WebAppArchiveOptions('example.mlapp',...
    'AdditionalFiles','myvars.mat',...
    'OutputDir','D:\Documents\MATLAB\work\WebApps',...
    'Verbose','on',...)
```

```
opts = 

  WebAppArchiveOptions with properties:

            ArchiveName: 'example'
                AppFile: 'D:\Documents\MATLAB\work\example.mlapp'
        AdditionalFiles: {'D:\Documents\MATLAB\work\myvars.mat'}
    AutoDetectDataFiles: on
              OutputDir: 'D:\Documents\MATLAB\work\WebApps'
                Verbose: on
```

Build a web app archive by passing the `WebAppArchiveOptions` object as an input to the build function.

```
buildResults = compiler.build.webAppArchive(opts);
```

You can modify property values of an existing `WebAppArchiveOptions` object using dot notation. For example, change the input file to `example2.m`.

```
opts.AppFile = 'example2.m';
```

This allows you to compile multiple archives using the same options object.

### Get Build Information From Web App Archive

Create a web app archive and save the build type, path to the archive file, and build options to a `compiler.build.Results` object.

Save the `compiler.build.webAppArchive` information to a `Results` object by declaring an output variable.

```
results = compiler.build.webAppArchive('example.mlapp')
```

```
results = 
  Results with properties:

    BuildType: 'webAppArchive'
        Files: {'D:\Documents\MATLAB\work\examplewebAppArchive\example.ctf'}
      Options: [1×1 compiler.build.WebAppArchiveOptions]
```

## Input Arguments

### `AppFile` — Path to main file

character vector | string scalar

Path to the main file, specified as a row character vector or a string scalar. The file must be a MATLAB app with the `.mlapp` extension. The path can be relative to the current working directory or absolute.

Example: `'mywebapp.mlapp'`

Data Types: `char` | `string`

### `opts` — Web app build options

`compiler.build.WebAppArchiveOptions` object

Web app build options, specified as a `compiler.build.WebAppArchiveOptions` object.

### Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Verbose','on'`

### `AdditionalFiles` — Additional files

character vector | string scalar | cell array of character vectors | string array

Additional files, specified as a character vector, a string scalar, a string array, or a cell array of character vectors. The listed files are included in the web app archive. File paths can be relative to the current working directory or absolute.

Example: `'AdditionalFiles',["myvars.mat","myfunc.m"]`

Data Types: `char` | `string` | `cell`

### `ArchiveName` — Name of generated web app archive

`AppFile` (default) | character vector | string scalar

Name of the generated web app archive, specified as a character vector or a string scalar. The default value is the file name of `AppFile`.

Example: `'ArchiveName','MyWebApp'`

Data Types: `char` | `string`

### `AutoDetectDataFiles` — Flag to automatically include data files

`'on'` (default) | on/off logical value

Flag to automatically include data files, specified as `'on'` or `'off'`, or as numeric or logical `1` (`true`) or `0` (`false`). A value of `'on'` is equivalent to `true`, and `'off'` is equivalent to `false`. Thus, you can use the value of this property as a logical value. The value is stored as an on/off logical value of type `matlab.lang.OnOffSwitchState`.

- If you set this property to `'on'`, then data files that are provided as inputs to certain functions (`load`, `fopen`, etc) are automatically included with the web app archive.
- If you set this property to `'off'`, then data files must be added to the web app archive using the `AdditionalFiles` property.

Example: `'AutoDetectDataFiles','Off'`

Data Types: `logical`

**`OutputDir` — Path to output directory**
`'`*`ArchiveName`*`webAppArchive'` (default) | character vector | string scalar

Path to the output directory, specified as a character vector or a string scalar. The path can be relative to the current working directory or absolute.

If no path is specified, a build folder named *`ArchiveName`*`webAppArchive` is created in the current working directory.

Example: `'OutputDir','D:\Documents\MATLAB\work\mymagicwebAppArchive'`

Data Types: `char` | `string`

**`Verbose` — Flag to control build verbosity**
`'off'` (default) | on/off logical value

Flag to control build verbosity, specified as `'on'` or `'off'`, or as numeric or logical `1` (`true`) or `0` (`false`). A value of `'on'` is equivalent to `true`, and `'off'` is equivalent to `false`. Thus, you can use the value of this property as a logical value. The value is stored as an on/off logical value of type `matlab.lang.OnOffSwitchState`.

- If you set this property to `'on'`, then the MATLAB Command Window displays progress information indicating code generation stages and compiler output during the build process.
- If you set this property to `'off'`, then the command window does not display progress information.

Example: `'Verbose','On'`

Data Types: `logical`

## Output Arguments

**`results` — Build results**
`compiler.build.Results` object

Build results, returned as a `compiler.build.Results` object. The `Results` object contains the build type (`'webAppArchive'`), the path to the web app archive file (`.ctf`), and the build options, specified as a `WebAppArchiveOptions` object.

## See Also
`deploytool` | `mcc` | `webAppCompiler`

**Introduced in R2020b**

# compiler.build.WebAppArchiveOptions

Create a web app archive options object

## Syntax

```
opts = compiler.build.WebAppArchiveOptions(AppFile)
opts = compiler.build.WebAppArchiveOptions(AppFile,Name,Value)
```

## Description

`opts = compiler.build.WebAppArchiveOptions(AppFile)` creates a `WebAppArchiveOptions` object using a MATLAB app specified by `AppFile`. The `WebAppArchiveOptions` object is passed as an input to the `compiler.build.webAppArchive` function.

`opts = compiler.build.WebAppArchiveOptions(AppFile,Name,Value)` creates a `WebAppArchiveOptions` object with options specified as one or more name-value pairs. Options include the archive name, additional files to include, and the output directory.

## Examples

### Create Web App Archive Options Object

Create a `WebAppArchiveOptions` object on a Windows system from an existing app named `example.mlapp`.

Create a web app options object using the `compiler.build.WebAppArchiveOptions` command.

```
opts = compiler.build.WebAppArchiveOptions('example.mlapp')
```

```
opts = 

  WebAppArchiveOptions with properties:

            ArchiveName: 'example'
                AppFile: 'D:\Documents\MATLAB\work\example.mlapp'
        AdditionalFiles: {}
    AutoDetectDataFiles: on
              OutputDir: '.\examplewebAppArchive'
                Verbose: off
```

Build a web app archive by passing the `WebAppArchiveOptions` object as an input to the build function.

```
buildResults = compiler.build.webAppArchive(opts);
```

### Customize Web App Archive Options Object Using Name-Value Pairs

Create a `WebAppArchiveOptions` object on a Windows system from an existing app named `example.mlapp` to specify the archive name and output directory, add a function file, and automatically include data files provided as inputs to functions such as `load`.

```
opts = compiler.build.webAppArchive('example.mlapp',...
    'ArchiveName','MyWebApp',...
    'OutputDir','D:\Documents\MATLAB\work\WebApps',...
    'AdditionalFiles','mysubfunction.m',...
    'AutoDetectDataFiles','on')
```

```
opts = 

  WebAppArchiveOptions with properties:

            ArchiveName: 'MyWebApp'
                AppFile: 'D:\Documents\MATLAB\work\example.mlapp'
        AdditionalFiles: {D:\Documents\MATLAB\work\mysubfunction.m}
    AutoDetectDataFiles: on
              OutputDir: 'D:\Documents\MATLAB\work\WebApps'
                Verbose: off
```

You can modify property values of an existing `WebAppArchiveOptions` object using dot notation.

```
opts.Verbose = 'on'
```

```
opts = 

  WebAppArchiveOptions with properties:

            ArchiveName: 'MyWebApp'
                AppFile: 'D:\Documents\MATLAB\work\example.mlapp'
        AdditionalFiles: {D:\Documents\MATLAB\work\mysubfunction.m}
    AutoDetectDataFiles: on
              OutputDir: 'D:\Documents\MATLAB\work\WebApps'
                Verbose: on
```

## Input Arguments

### `AppFile` — Path to main file

character vector | string scalar

Path to the main file, specified as a row character vector or a string scalar. The file must be a MATLAB app with the `.mlapp` extension. The path can be relative to the current working directory or absolute.

Example: `'mywebapp.mlapp'`

Data Types: `char` | `string`

### Name-Value Pair Arguments

Specify optional comma-separated pairs of `Name,Value` arguments. `Name` is the argument name and `Value` is the corresponding value. `Name` must appear inside quotes. You can specify several name and value pair arguments in any order as `Name1,Value1,...,NameN,ValueN`.

Example: `'Verbose','on'`

**`AdditionalFiles` — Additional files**
character vector | string scalar | cell array of character vectors | string array

Additional files, specified as a character vector, a string scalar, a string array, or a cell array of character vectors. The listed files are included in the web app archive. File paths can be relative to the current working directory or absolute.

Example: `'AdditionalFiles',["myvars.mat","myfunc.m"]`

Data Types: `char` | `string` | `cell`

**`ArchiveName` — Name of generated web app archive**
`AppFile` (default) | character vector | string scalar

Name of the generated web app archive, specified as a character vector or a string scalar. The default value is the file name of `AppFile`.

Example: `'ArchiveName','MyWebApp'`

Data Types: `char` | `string`

**`AutoDetectDataFiles` — Flag to automatically include data files**
`'on'` (default) | on/off logical value

Flag to automatically include data files, specified as `'on'` or `'off'`, or as numeric or logical `1` (`true`) or `0` (`false`). A value of `'on'` is equivalent to `true`, and `'off'` is equivalent to `false`. Thus, you can use the value of this property as a logical value. The value is stored as an on/off logical value of type `matlab.lang.OnOffSwitchState`.

- If you set this property to `'on'`, then data files that are provided as inputs to certain functions (`load`, `fopen`, etc) are automatically included with the web app archive.
- If you set this property to `'off'`, then data files must be added to the web app archive using the `AdditionalFiles` property.

Example: `'AutoDetectDataFiles','Off'`

Data Types: `logical`

**`OutputDir` — Path to output directory**
`'`*`ArchiveName`*`webAppArchive'` (default) | character vector | string scalar

Path to the output directory, specified as a character vector or a string scalar. The path can be relative to the current working directory or absolute.

If no path is specified, a build folder named *`ArchiveName`*`webAppArchive` is created in the current working directory.

Example: `'OutputDir','D:\Documents\MATLAB\work\mymagicwebAppArchive'`

Data Types: `char` | `string`

**`Verbose` — Flag to control build verbosity**
`'off'` (default) | on/off logical value

Flag to control build verbosity, specified as `'on'` or `'off'`, or as numeric or logical `1` (`true`) or `0` (`false`). A value of `'on'` is equivalent to `true`, and `'off'` is equivalent to `false`. Thus, you can use the value of this property as a logical value. The value is stored as an on/off logical value of type `matlab.lang.OnOffSwitchState`.

- If you set this property to `'on'`, then the MATLAB Command Window displays progress information indicating code generation stages and compiler output during the build process.
- If you set this property to `'off'`, then the command window does not display progress information.

Example: `'Verbose','On'`

Data Types: `logical`

## Output Arguments

**`opts` — Web app archive build options**
`WebAppArchiveOptions` object

Web app archive build options, returned as a `WebAppArchiveOptions` object.

## See Also

`compiler.build.webAppArchive` | `webAppCompiler`

**Introduced in R2020b**

# compiler.build.Results

Compiler build results object

## Description

A `compiler.build.Results` object contains the build type, files, and build options of a `compiler.build` function.

All `Results` properties are read-only. You can use dot notation to query these properties.

## Creation

There are several ways to create a `compiler.build.Results` object.

- Create a standalone application using `compiler.build.standaloneApplication`.
- Create a standalone Windows application using `compiler.build.standaloneWindowsApplication`.
- Create a web app archive using `compiler.build.webAppArchive`.
- Create a production server archive using `compiler.build.productionServerArchive`.

### Properties

**`BuildType` — Build type**
`'standaloneApplication'` | `'standaloneWindowsApplication'` | `'webAppArchive'` | `'productionServerArchive'`

This property is read-only.

The name of the `compiler.build` function used to generate the results, specified as one of these character vectors:

- `'standaloneApplication'`—indicates results from the `compiler.build.standaloneApplication` function.
- `'standaloneWindowsApplication'`—indicates results from the `compiler.build.standaloneWindowsApplication` function.
- `'webAppArchive'`—indicates results from the `compiler.build.webAppArchive` function.
- `'productionServerArchive'`—indicates results from the `compiler.build.productionServerArchive` function.

Example: `'productionServerArchive'`

Data Types: `char`

**`Files` — Paths to build files**
cell array of character vectors

This property is read-only.

Paths to the compiled build files of the associated `compiler.build` function, specified as a cell array of character vectors.

| Build Type | Files |
|---|---|
| `standaloneApplication` | 2×1 cell array<br>`{'path\to\`*`ExecutableName`*`.exe'}`<br>`{'path\to\readme.txt'}` |
| `standaloneWindowsApplication` | 3×1 cell array<br>`{'path\to\`*`ExecutableName`*`.exe'}`<br>`{'path\to\splash.png'}`<br>`{'path\to\readme.txt'}` |
| `webAppArchive` | 1×1 cell array<br>`{'path\to\`*`ArchiveName`*`.ctf'}` |
| `productionServerArchive` | 1×1 cell array<br>`{'path\to\`*`ArchiveName`*`.ctf'}` |

Example: `{'D:\Documents\MATLAB\work\MagicSquarewebAppArchive\MagicSquare.ctf'}`

Data Types: `cell`

**Options — Build options**
`StandaloneApplicationOptions` | `WebAppArchiveOptions` | `ProductionServerArchiveOptions`

This property is read-only.

Build options from the associated `compiler.build` function, specified as an options object of the corresponding build type.

| Build Type | Options |
|---|---|
| `standaloneApplication` | `StandaloneApplicationOptions` |
| `standaloneWindowsApplication` | `StandaloneApplicationOptions` |
| `webAppArchive` | `WebAppArchiveOptions` |
| `productionServerArchive` | `ProductionServerArchiveOptions` |

## Examples

### Get Build Information From Standalone Application

Create a standalone application and save information about the build type, included files, and build options to a `compiler.build.Results` object.

Save the `compiler.build.standaloneApplication` information to a `Results` object by declaring an output variable.

```
results = compiler.build.standaloneApplication('mymagic.m')
```

```
results =
  Results with properties:

            BuildType: 'standaloneApplication'
                Files: {2×1 cell}
              Options: [1×1 compiler.build.StandaloneApplicationOptions]
```

The `Files` property contains the paths to the generated standalone executable and readme files.

### Get Build Information From Standalone Windows Application

Create a standalone Windows application and save information about the build type, included files, and build options to a `compiler.build.Results` object on a Windows system.

Save the `compiler.build.standaloneWindowsApplication` information to a `Results` object by declaring an output variable.

```
results = compiler.build.standaloneWindowsApplication('mymagic.m','AdditionalFiles',["myvars.mat","mysubfunction.m"])
```

```
results = 

  Results with properties:

            BuildType: 'standaloneWindowsApplication'
                Files: {3×1 cell}
              Options: [1×1 compiler.build.StandaloneApplicationOptions]
```

The `Files` property contains the paths to the generated standalone executable, splash image, and readme files.

### Get Build Information From Web App Archive

Create a web app archive and save the build type, path to the archive file, and build options to a `compiler.build.Results` object.

Save the `compiler.build.webAppArchive` information to a `Results` object by declaring an output variable.

```
results = compiler.build.webAppArchive('example.mlapp')
```

```
results = 

  Results with properties:

    BuildType: 'webAppArchive'
        Files: {'D:\Documents\MATLAB\work\examplewebAppArchive\example.ctf'}
      Options: [1×1 compiler.build.WebAppArchiveOptions]
```

### Get Build Information From Deployable Archive

Create a production server archive and save the build type, path to the archive file, and build options to a `compiler.build.Results` object.

Save the `compiler.build.productionServerArchive` information to a `Results` object by declaring an output variable.

```
results = compiler.build.productionServerArchive('mymagic.m')
```

```
results = 

  Results with properties:

            BuildType: 'productionServerArchive'
                Files: 'D:\Documents\MATLAB\work\mymagicproductionServerArchive\mymagic.ctf'
              Options: [1×1 compiler.build.ProductionServerArchiveOptions]
```

## See Also

`compiler.build.productionServerArchive` | `compiler.build.standaloneApplication` | `compiler.build.standaloneWindowsApplication` | `compiler.build.webAppArchive`

**Introduced in R2020b**

# webAppCompiler

Package App Designer apps for web deployment

## Syntax

```
webAppCompiler
webAppCompiler project_name
```

## Description

`webAppCompiler` opens the **Web App Compiler** for the creation of a new project. For more information on the app, see **Web App Compiler**.

`webAppCompiler project_name` opens the **Web App Compiler** with the project preloaded. If the project does not exist, an error will be generated, and the program will exit.

## Examples

### Create a New Web App Project

Open the **Web App Compiler** to create a new project.

```
webAppCompiler
```

### Open an Existing Web App Project

Open an existing web app project.

```
webAppCompiler myWebApp
```

## Input Arguments

**`project_name` — name of the project to be opened or packaged**
character array or string

Specify the name of a previously saved web app project. The project must be on the current path.

## Compatibility Considerations

**-package option will be removed**
*Not recommended starting in R2020a*

The `-package` option will be removed. To package MATLAB apps into web apps, use the **Web App Compiler** app.

## See Also

**Web App Compiler** | `deploytool` | `mcc`

**Introduced in R2018a**

8

# Web App Examples

## Simple Mortgage Calculator Web App

This example shows how to create a web app and host it on the development version of MATLAB Web App Server . The example uses the simple calculator app from App Designer as a basis for the web app. For information about the app and the numerical values you can enter, see “App that Calculates and Plots Data Based on Numerical Input”. In the workflow, you:

- Package the simple calculator app from App Designer using the Web App Compiler app in MATLAB Compiler. This step creates a web app archive (`.ctf`) file.
- Deploy the web app archive (`.ctf`) file to the development version of MATLAB Web App Server.
- Run the web app from the web apps home page.

### Prerequisites

1. Install and configure the development version of MATLAB Web App Server.
   - For information on installing the server, see “Install the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-2.
   - For information on configuring the server, see “Configure the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-5.
2. Copy the App Designer file `Mortgage.mlapp` to your current working directory. The default location of the file is:

| Operating System | Default File Location |
|---|---|
| Windows | `C:\Program Files\MATLAB\R2020b\examples\matlab\main\Mortgage.mlapp` |
| Linux | `/usr/local/MATLAB/R2020b/examples/matlab/main/Mortgage.mlapp` |
| macOS | `/Applications/MATLAB/R2020b.app/examples/matlab/main/Mortgage.mlapp` |

### Package and Create Web App

1. Start MATLAB.
2. Type `webAppCompiler` at the MATLAB command line to open the **Web App Compiler** app.
3. In the **MAIN FILE** section of the toolstrip, click the ✚ button to add the `Mortgage.mlapp` file to the project. The Web App Compiler automatically resizes to include an **App details** section that contains information about the app such as app name, author, summary, description, and version. You can edit information about the app in App Designer by clicking **Edit App Details**. Click **Refresh** to update Web App Compiler with any changes you have made.
   - *(Optional)* Make sure to use a display name that is easy to distinguish when your web app is deployed to the server.
   - *(Optional)* Provide a version number for tracking purposes. The version number is visible on the web apps home page.
   - *(Optional)* Add a description for your web app in the **Summary** field. This description is visible on the web apps home page.

4 In the **Archive information** section, specify the archive name as `myMortgageWebApp`.

5 Click **Package** to package the app, and create a web app archive (`.ctf` file).

In the **Save Project** dialog box that opens, specify a project name and a location where you want to save the web app project. **Web App Compiler** saves your project and opens a **Package** dialog box.

6 Once packaging is complete, in the **Package** dialog box, click **Open output folder**. This opens the project folder, which contains the following files:

- `myMortgageWebApp.ctf`
- `mccExcludedFiles.log`
- `PackagingLog.html`
- `requiredMCRProducts.txt`

You can view the log file, `PackagingLog.html`, to see the exact `mcc` syntax used to package and create the web app archive.

## Deploy Web App

1 Navigate to the project folder generated by Web App Compiler during the packaging process.

2 Copy the web app archive file `myMortgageWebApp.ctf` to the app folder configured by the server. The default location is:

| Operating System | Apps Folder Location |
|---|---|
| Windows | `%ProgramData%\MathWorks\webapps\R2020b\apps` |
| Linux | `/local/MathWorks/webapps/R2020b/apps` |
| macOS | `/Library/Application Support/MathWorks/webapps/R2020b/apps` |

You can also open the apps folder by clicking the **Open App Folder** button in the server application. For more information, see “Start the Server Application” on page 3-11.

3 Click the **Open Home Page** button in the server application.

This action opens the web apps home page using your default web browser. You see a tile displaying the simple mortgage calculator web app. Your web app is now deployed.

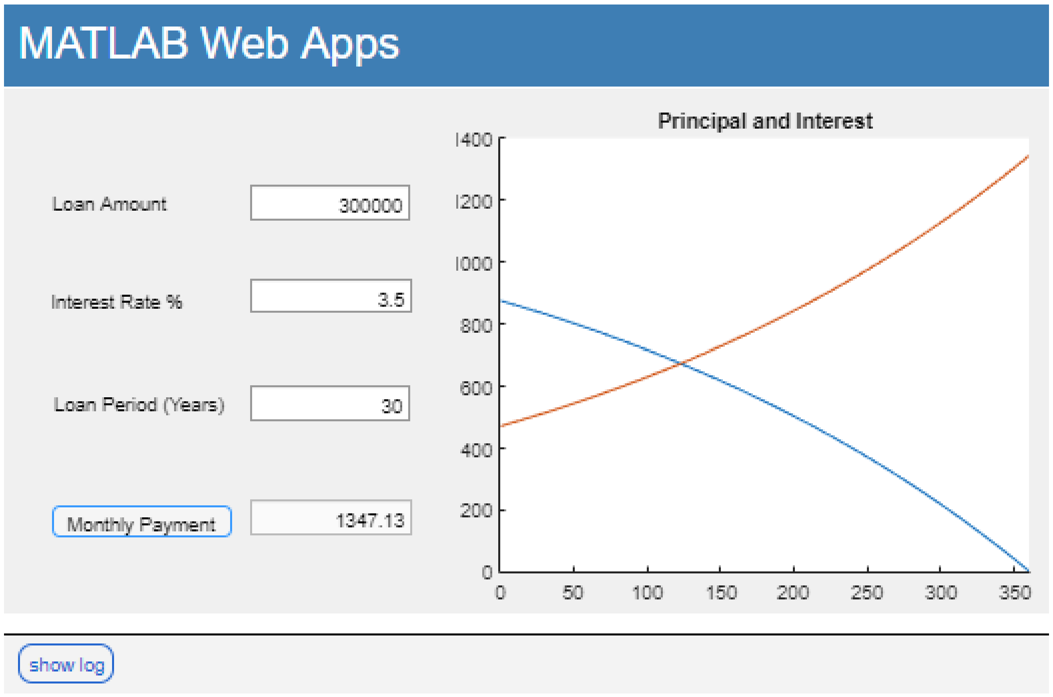
## Run Web App

1 To run a web app, click the `myMortgageWebApp` tile on the web apps home page.

The web app opens in a new tab.

2 Click the **Monthly Payment** button to get the monthly payment and the principal and interest graph.

You have successfully created, deployed, and run a web app.



Loan Amount: 300000
Interest Rate %: 3.5
Loan Period (Years): 30
Monthly Payment: 1347.13

**Mortgage Calculator Web App**

## See Also

## More About

- “Create Web App” on page 4-2
- “Deploy Web App” on page 4-4
- “Run Web App” on page 4-6
- “Simulink Simulation Web App” on page 8-5

## Simulink Simulation Web App

This example shows how to create a web app containing a Simulink simulation and host it on the development version of MATLAB Web App Server. The example uses the mass spring damper model in Simulink and a MATLAB app that invokes the model as a basis for the web app. The APIs for creating a simulation can be found in the Simulink Compiler product. In the workflow, you:

- Package the MATLAB app containing Simulink simulation using the Web App Compiler app in MATLAB Compiler. This step creates a web app archive (`.ctf`) file.
- Deploy the web app archive (`.ctf`) file to the development version of MATLAB Web App Server.
- Run the web app from the web apps home page.

### Prerequisites

**Note** This example requires the Simulink Compiler product. For details, see “Simulink Compiler Workflow Overview” (Simulink Compiler).

1. Install and configure the development version of MATLAB Web App Server.
   - For information on installing the server, see “Install the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-2.
   - For information on configuring the server, see “Configure the Development Version of MATLAB Web App Server in MATLAB Compiler” on page 3-5.
2. Copy the Simulink model file `MassSpringDamperModel.slx` and the corresponding MATLAB app `MassSpringDamperApp.mlapp` to your current working directory. The default location for the files is:

| Operating System | Default Location for Files |
|---|---|
| Windows | `C:\Program Files\MATLAB\R2020b\examples\simulinkcompiler\main\` |
| Linux | `/usr/local/MATLAB/R2020b/examples/simulinkcompiler/main` |
| macOS | `/Applications/MATLAB/R2020b.app/examples/simulinkcompiler/main` |

3. Open the `MassSpringDamperApp.mlapp` in MATLAB App Designer and switch to **Code View**. Verify that the Simulink Compiler APIs to create a simulation are present in the `SimulateButtonPushed` callback.

### Package and Create Web App

1. Start MATLAB.
2. Type `webAppCompiler` at the MATLAB command line to open the **Web App Compiler** app.
3. In the **MAIN FILE** section of the toolstrip, click the ➕ button to add the `MassSpringDamperApp.mlapp` file to the project. The Web App Compiler automatically resizes to include an **App details** section that contains information about the app such as app name, author, summary, description, and version. You can edit information about the app in App

Designer by clicking **Edit App Details**. Click **Refresh** to update Web App Compiler with any changes you have made.

- *(Optional)* Make sure to use a display name that is easy to distinguish when your web app is deployed to the server.
- *(Optional)* Provide a version number for tracking purposes. The version number is visible on the web apps home page.
- *(Optional)* Add a description for your web app in the **Summary** field. This description is visible on the web apps home page.

4. In the **Archive information** section, specify the archive name as `mySimulinkSimulationWebApp`.
5. Click **Package** to package the app, and create a web app archive (`.ctf` file).

   In the **Save Project** dialog box that opens, specify a project name and a location where you want to save the web app project. **Web App Compiler** saves your project and opens a **Package** dialog box.
6. Once packaging is complete, in the **Package** dialog box, click **Open output folder**. This opens the project folder, which contains the following files:

   - `mySimulinkSimulationWebApp.ctf`
   - `mccExcludedFiles.log`
   - `PackagingLog.html`
   - `requiredMCRProducts.txt`

   You can view the log file, `PackagingLog.html`, to see the exact `mcc` syntax used to package and create the web app archive.

## Deploy Web App

1. Navigate to the project folder generated by Web App Compiler during the packaging process.
2. Copy the web app archive file `mySimulinkSimulationWebApp.ctf` to the app folder configured by the server. The default location is:

| Operating System | Apps Folder Location |
|---|---|
| Windows | `%ProgramData%\MathWorks\webapps\R2020b\apps` |
| Linux | `/local/MathWorks/webapps/R2020b/apps` |
| macOS | `/Library/Application Support/MathWorks/webapps/R2020b/apps` |

   You can also open the apps folder by clicking the **Open App Folder** button in the server application. For more information, see “Start the Server Application” on page 3-11.
3. Click the **Open Home Page** button in the server application.

   This action opens the web apps home page using your default web browser. You see a tile displaying the simple mortgage calculator web app. Your web app is now deployed.

## Run Web App

1. To run a web app, click the `mySimulinkSimulationWebApp` tile on the web apps home page.

   The web app opens in a new tab.
2. Click the **Simulate** button to run the simulation.

   You have successfully created, deployed, and run a web app.

## See Also

## More About

- “Create Web App” on page 4-2
- “Deploy Web App” on page 4-4
- “Run Web App” on page 4-6
- “Simple Mortgage Calculator Web App” on page 8-2