How to deploy WebChat on the web

Deployment options

There are several options for deploying WebChat to a site, which vary depending on how much customization of its appearance or behavior is needed.

a) Unpacking chat

The quickest way is to just insert the simple code you get from the virtual assistant (VA) creator before the `</body>` end tag. It will look similar to the no-config.html sample, but with the bot ID filled in. Of course, you can also use services such as Google Tag Manager to deploy it without having to manually intervene in the code.

The design and behavior of the webchat can then be set from Feedyou (or in Designer if you have access). All domains where VA will run must be manually enabled from Feedyou. If VA is in test mode, it will not be displayed on the site unless `#feedbot-test-mode` is added to the end of the URL.

b) Chat to custom element

If it is desired to modify the appearance or behavior of the chat, it is possible to insert it into a custom HTML element that can be styled in your own way. Insert the DOM element as the second parameter of the `BotChat.App` function. The chat will then be rendered into it. The element doesn't have to be a popup, it can be a part of the page.

> If the chat is hidden when the page is loaded and is only displayed based on user action, it is important to call `BotChat.App` only after it has been displayed to avoid unnecessary VA calls.

Examples of possible uses are available on GitHub, e.g.:

- samples/feedyou/expandable.html - popup window in custom element
- samples/feedyou/embed.html - bot placed in a specific page element
- samples/feedyou/persistent.html - a special variant that saves the user ID and conversation in the browser so that it can refer to it the next time it visits the site

c) Advanced deployment methods

WebChat is an open library built on top of React, so it can be easily plugged into an existing React application or modified in any way. For more information, please contact Feedyou.

d) Testing on any page

If you just want to test how the VA will look on any page, you can set the appropriate data in the following code and paste it all into the console in the web inspector (usually just right click anywhere on the page and then *Inspect* or *Explore*). **DO NOT** use this code in a production environment, that is what options a) - c) are for. If the bot is still in test mode, you must first add `#feedbot-test-mode` to the end of the page URL

```
(function (f,y,b,o,t,s){(t = y.createElement(b)), (s = y.getElementsByTagName(b)[0]);t.async = 1;t.src = o;t.onload=function(){
BotChat.App({
  bot: { id: 'feedbot-...' }, // can be found at the end of the "Code" link in the "Channels settings" section of the Designer
  channel: { id: '...' }, // is a 6-letter code that can be found at the end of the "Preview" link for a given WebChat in the "Channels settings" section of the Designer
})
};s.parentNode.insertBefore(t, s);
})(window,document,"script","https://feedyou.azureedge.net/webchat/latest/botchat-es5.js");
(function (f,y,b,o,t,s){(t = y.createElement(b)), (s = y.getElementsByTagName(b)[0]);t.rel = 'stylesheet';t.type = 'text/css';t.href = o;s.parentNode.insertBefore(t, s);
})(window,document,"link","https://feedyou.azureedge.net/webchat/latest/botchat.css");
```

Linking to the bot

Regardless of which insertion method you choose, there are a few basic settings that need to be set to connect to the correct bot. **These are indicated in the examples by `...` and you get them from Feedyou** - specifically `directLine.secret` and `bot.id`. During testing, you can use the values listed in the comments in that demo.

Other settings

There are other settings that can be used to modify the behavior of WebChat:

- `locale` - allows to change the language of texts in the WebChat component from the default

Czech (e.g. en, sk , sr, hu ...)
- `disableInputWhenNotNeeded` - if set to `true`, hides the text input if the bot does not ask the user anything (suitable for virtual assistants without NLP model)
- `user.id` and `user.name` - if we know the logged in user, we can give the bot his ID or name
- `userData` - object that can be used to pass any information that we know about the user to the bot (e.g. email, phone, etc.)
- `startOverTrigger` - allows you to attach an event to any element that causes the conversation to restart - e.g. the "Start over" button (more info on GitHub and on the README NA GitHubu)

```
directLine: { secret: '...' },
bot: { id: '...', name: 'Chatbot' },
theme: { mainColor: '#d83838' },
locale: 'en', // Abbreviation of the language in which the webchat elements should be, e.g.
input text for text
disableInputWhenNotNeeded: true // Hide input for text until the user is prompted
```

Appearance modifications

The appearance of the chat can then be freely customized using CSS styles - ideally all selector rules should start with `.wc-app`. Because of the openness to such customization options, on the other hand, it may happen that WebChat is negatively affected by web styling (e.g. generic rules on basic elements like `<button>` etc.) - in this case it is necessary to resolve these problems by refining them to target only `.wc-app`.

URL parameters

WebChat can be influenced by the following parameters in the URL suitable e.g. during testing:

- `#feedbot-test-mode` sets `testMode: true` to the `userData` setting, which enables the test mode for the user in the bot
- `#feedbot-intro-dialog` overrides the default dialog that the bot starts the conversation with (can also be set using the `introDialog.id` parameter in the webchat settings)
- the logic for passing parameters from the `#` part of the URL to the bot can be customized in two ways
  - e.g. if we send a link to users we know, we can set the `user.id`, according to it, so it will be possible to trace which conversation was made by which user and, in addition, it will be possible to recognize and react if they return to the link in the future
  - set any variable in `userData` according to the parameter and then use it in the bot communication, or decide according to it (e.g. if it accesses directly, i.e. without the parameter in the URL or from the QR code, where some parameter will be or e.g. from the newsstand, where some parameter is also preset)

- select which dialog to run instead of the default `main` dialog according to the parameter

```
{
  // ...
  userData: { zdroj: location.hash.substr(1) }, // everything after # in the URL is passed to
the "source" variable
  introDialog: { id: location.hash.substr(1) === 'xyz' ? 'abc' : 'main' } // if #xyz is in
the URL, run the "abc" dialog
}
```

Background communication

WebChat can also communicate with VA in the background using events (so-called back-channel) in addition to the transmission of classic messages.

Bot → WebChat

The following is a demonstration of how to react to any event defined anywhere in the communication tree at the exact moment the user passes through the location:

```
// instead of the classic setting of the secret in the directLine property, it is necessary
to create
// BotConnection object and pass it to the settings
var botConnection = new BotChat.DirectLine({
  secret: "...",
});

BotChat.App({
  botConnection: botConnection,
  bot: { id: "feedbot-..." },
});

// it is then possible to register callbacks to certain events and from activity.value
// use any value passed from the bot
botConnection.activity$.filter(function (activity) {
  return (
    activity.type === "event" && activity.name === "set-reload-timeout"
  );
}).subscribe(function (activity) {
  setTimeout(function () {
```

```
    location.reload();
  }, parseInt(activity.value || 60000));
});
```

WebChat → Bot

Conversely, the page on which WebChat is running can trigger any of the following events in the bot by simply using the event trigger on the `window` object:

- `feedbot:trigger-dialog` starts selected dialog specified in dialog property of CustomEvent (for example `window.dispatchEvent(new CustomEvent('feedbot:trigger-dialog', { detail: 'package-status' }))`)
- `feedbot:start-over` restarts conversation, which is the same behavior as startOverTrigger callback in config above (for example `window.dispatchEvent(new CustomEvent('feedbot:start-over'))`)

If support of Internet Explorer is required, please provide use fallback to support custom event creation there.

Revision #3
Created 18 August 2021 08:48:02
Updated 29 March 2023 11:58:04