

Developing Applications - Bots

Module 4 - Use adaptive cards with Bots Student Lab Manual

Lab 4: Use adaptive cards with Bots

Introduction

Adaptive Cards are an open card exchange format that enables developers to exchange UI content in a common and consistent way. The Bot Framework has the ability to use this type of cards and provide a richer interaction experience.

The Adaptive Card can contain any combination of text, speech, images, buttons, and input fields. Adaptive Cards are created using the JSON format specified in Adaptive Cards schema, which gives you full control over card content and format.

In this lab, you will go through the creation of adaptive cards with NodeJs code and also how to parse a JSON payload into an Adaptive Card, which makes it easy to manipulate the object model or even render Adaptive Cards inside your app by using our [renderer SDKs](#).

Objectives

After completing this lab, you will be able to:

- build an Adaptive Card using NodeJs
- Parse an Adaptive Card from JSON
- Add adaptive Card as attachments to the bot replies

Prerequisites

- Visual Studio Code
- Bot Framework Emulator

Estimated Time to Complete This Lab

30 minutes

For More Information

JavaScript SDK for Authoring Cards: <https://docs.microsoft.com/en-us/adaptive-cards/sdk/authoring-cards/javascript>

Adaptive Cards for Bot Developers: <https://docs.microsoft.com/en-us/adaptive-cards/get-started/bots>

Exercise 1: Use adaptive cards with Bots

Introduction

This exercise will guide you to create bot and reply to users using Adaptive Cards and how to handle user interaction with them.

Objectives

After completing this lab, you will be able to:

- build an Adaptive Card using NodeJS
- Parse an Adaptive Card from JSON
- Add adaptive Card as attachments to the bot replies

Prerequisites

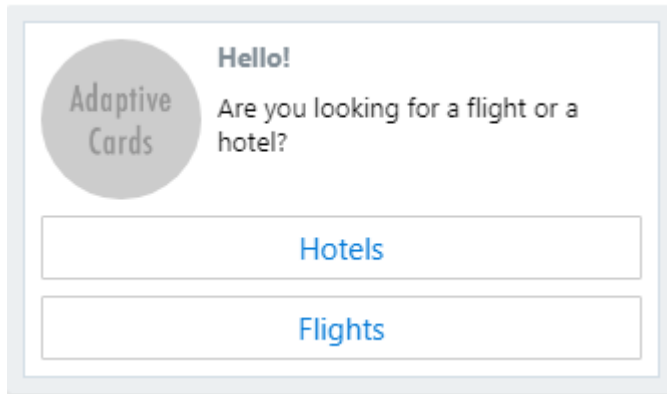
None

Task 1: build an Adaptive Cards using NodeJS

1. First of all open the folder “Begin_Lab4” in Visual Studio Code. The content of this project is a standard echo bot as you have seen in Lab1. Additionally we added some functionality for a hotel search sample.
2. Install the AdaptiveCards package which provides types for working with adaptive cards in NodeJS
`npm install --save adaptivecards`
3. Change the creation of builder.UniversalBot to use an Adaptive Card to show the user what he is able to do with the bot. Therefor replace the following with the code from **Snippet1.txt** from the **Lab4_CodeSnippets** folder.

```
var bot = new builder.UniversalBot(connector, function (session) {  
    session.send("You said: %s", session.message.text);  
});
```

The previous code will generate a card similar to this one:

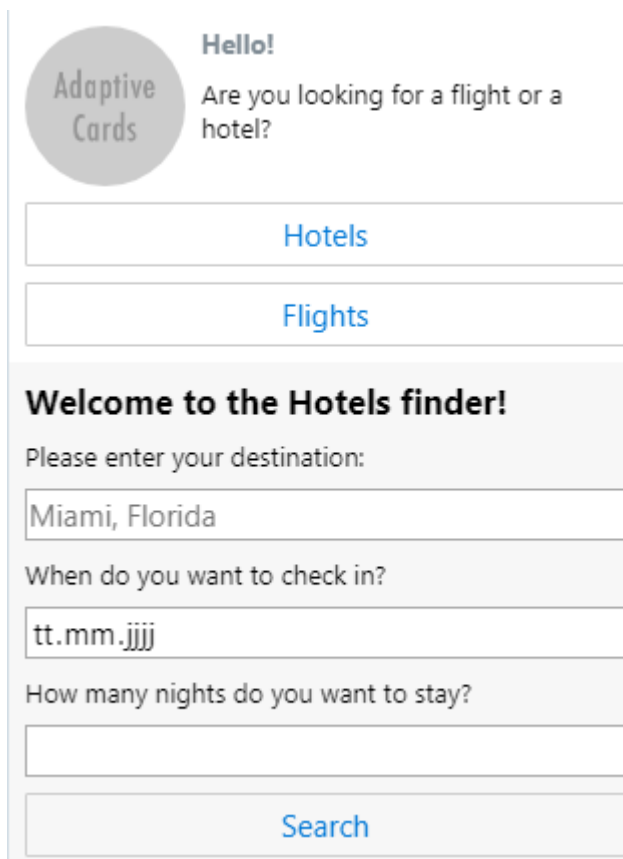


Adaptive Cards

Hello!
Are you looking for a flight or a hotel?

Hotels

Flights



Adaptive Cards

Hello!
Are you looking for a flight or a hotel?

Hotels

Flights

Welcome to the Hotels finder!

Please enter your destination:

Miami, Florida

When do you want to check in?

tt.mm.jjjj

How many nights do you want to stay?

Search

4) Add the dialog for searching hotels at the bottom of app.js

```
// Search Hotels
bot.dialog('hotels-search', require('./hotels-search'));
```

5) To log errors inside of the bot to the console add the following code

```
// log any bot errors into the console
bot.on('error', function (e) {
    console.log('And error ocurred', e);
});
```

6) Add the function to validate the hotel search and a function that processes the submit action sent to the UniversalBot.

```
function processSubmitAction(session, value) {
    var defaultMessage = 'Please complete all the search parameters';
    switch (value.type) {
        case 'hotelSearch':
            // Search, validate parameters
            if (validateHotelSearch(value)) {
                // proceed to search
                session.beginDialog('hotels-search', value);
            } else {
                session.send(defaultErrorMessage);
            }
            break;
        default:
            // A form data was received, invalid or incomplete since the
            previous validation did not pass
            session.send(defaultErrorMessage);
    }
}

function validateHotelSearch(hotelSearch) {
    if (!hotelSearch) {
        return false;
    }

    // Destination
    var hasDestination = typeof hotelSearch.destination === 'string' &&
    hotelSearch.destination.length > 3;

    // Checkin
    var checkin = Date.parse(hotelSearch.checkin);
    var hasCheckin = !isNaN(checkin);
    if (hasCheckin) {
        hotelSearch.checkin = new Date(checkin);
    }

    // Nights
    var nights = parseInt(hotelSearch.nights, 10);
    var hasNights = !isNaN(nights);
    if (hasNights) {
        hotelSearch.nights = nights;
    }

    return hasDestination && hasCheckin && hasNights;
}
```