
ORACLE®

Oracle Digital Assistant

The Complete Training

Webview component

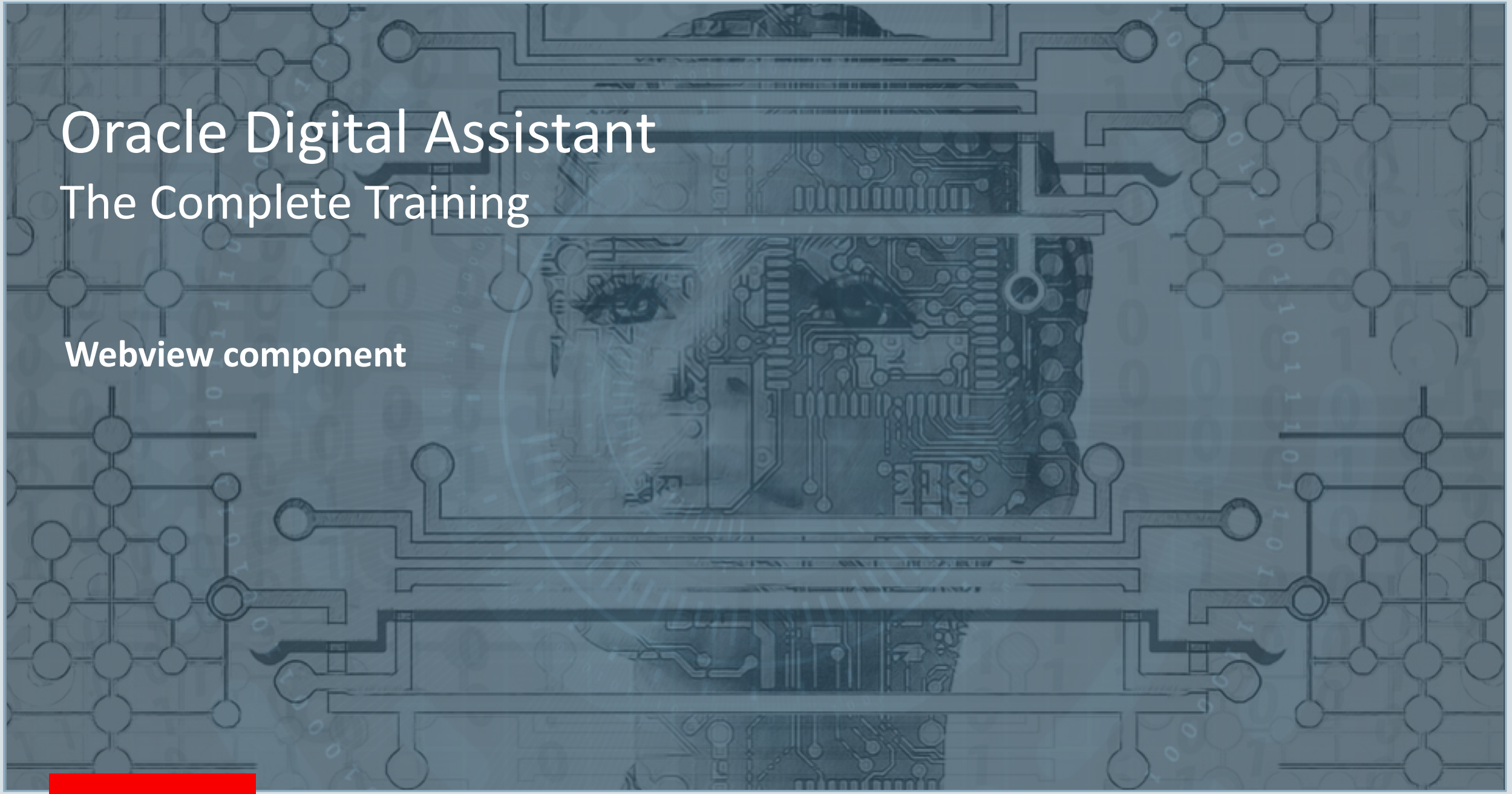


Image courtesy of pixabay.com

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Topic agenda

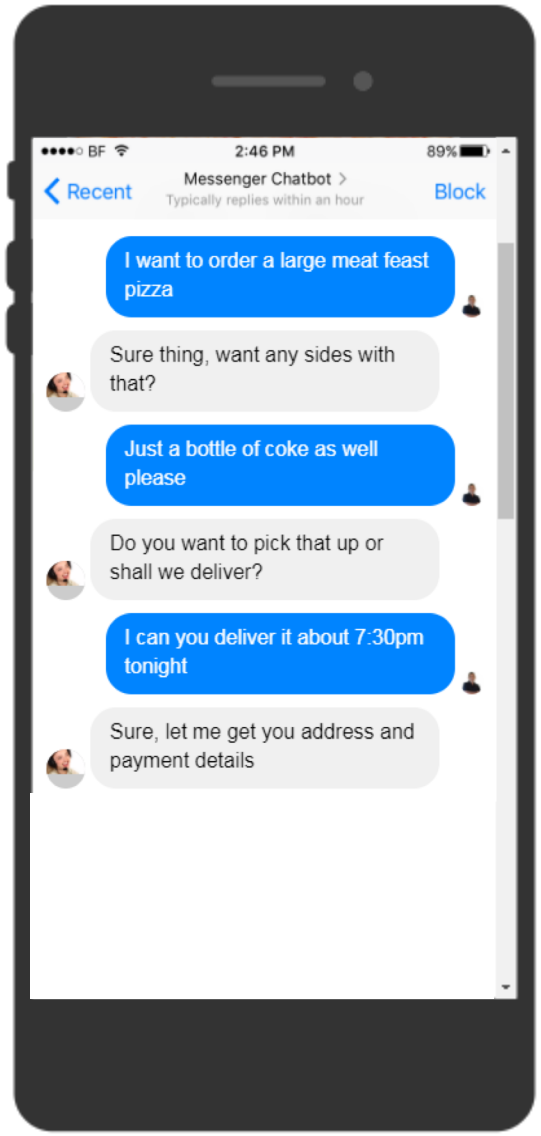
1. When conversation is not enough
2. System.Webview
3. Local web applications
4. Remote web applications
5. Remote web app vs. local web app

Topic agenda

1. When conversation is not enough
2. System.Webview
3. Local web applications
4. Remote web applications
5. Remote web app vs. local web app

A typical conversation

- Natural language conversation
  - Unstructured
  - Free flowing
  - Conversational



Not all use cases are perfect matches for the conversational channel.

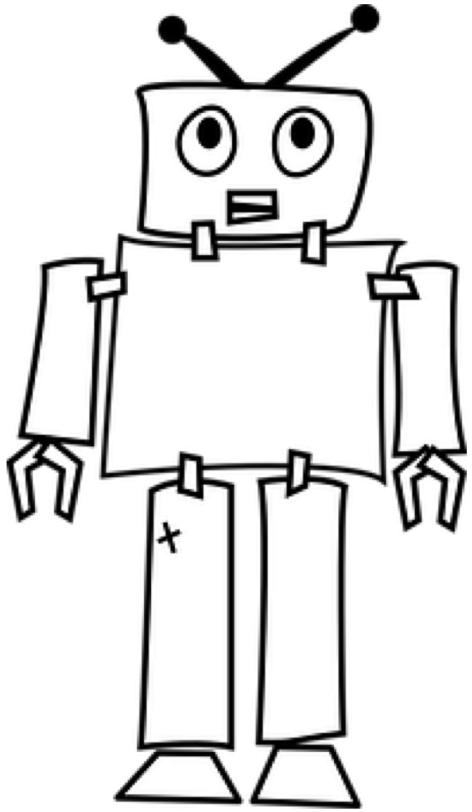


Image courtesy of Farbentek at FreeDigitalPhotos.net

Limitations of the conversational channel

- Long conversations are difficult to have in a chat
  - Users have a limited attention rate and are easily distracted
  - Hard to change user input at a later point in a conversation
  - Lack of support for structured data input
- UI limitations
  - Missing widgets (e.g. date picker or calendar, LOV)
  - No type-ahead support
- Display of sensitive information
  - No way to obfuscate password or credit card details

Humans use a whiteboards and lists if conversation is not enough.

So what's the equivalent for chatbots?

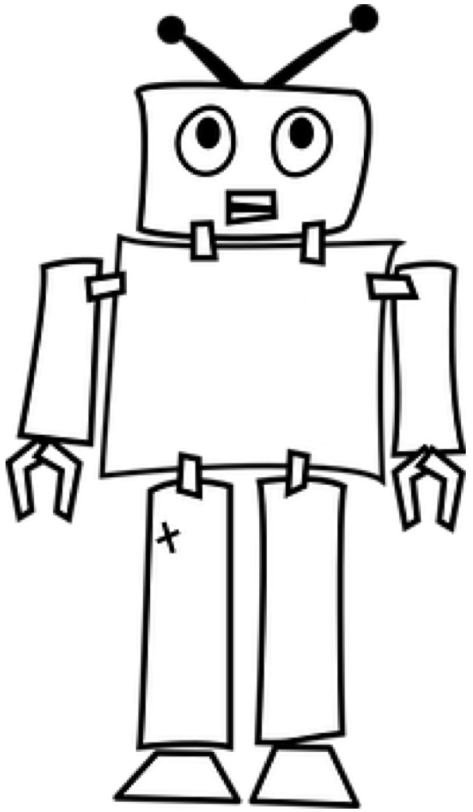
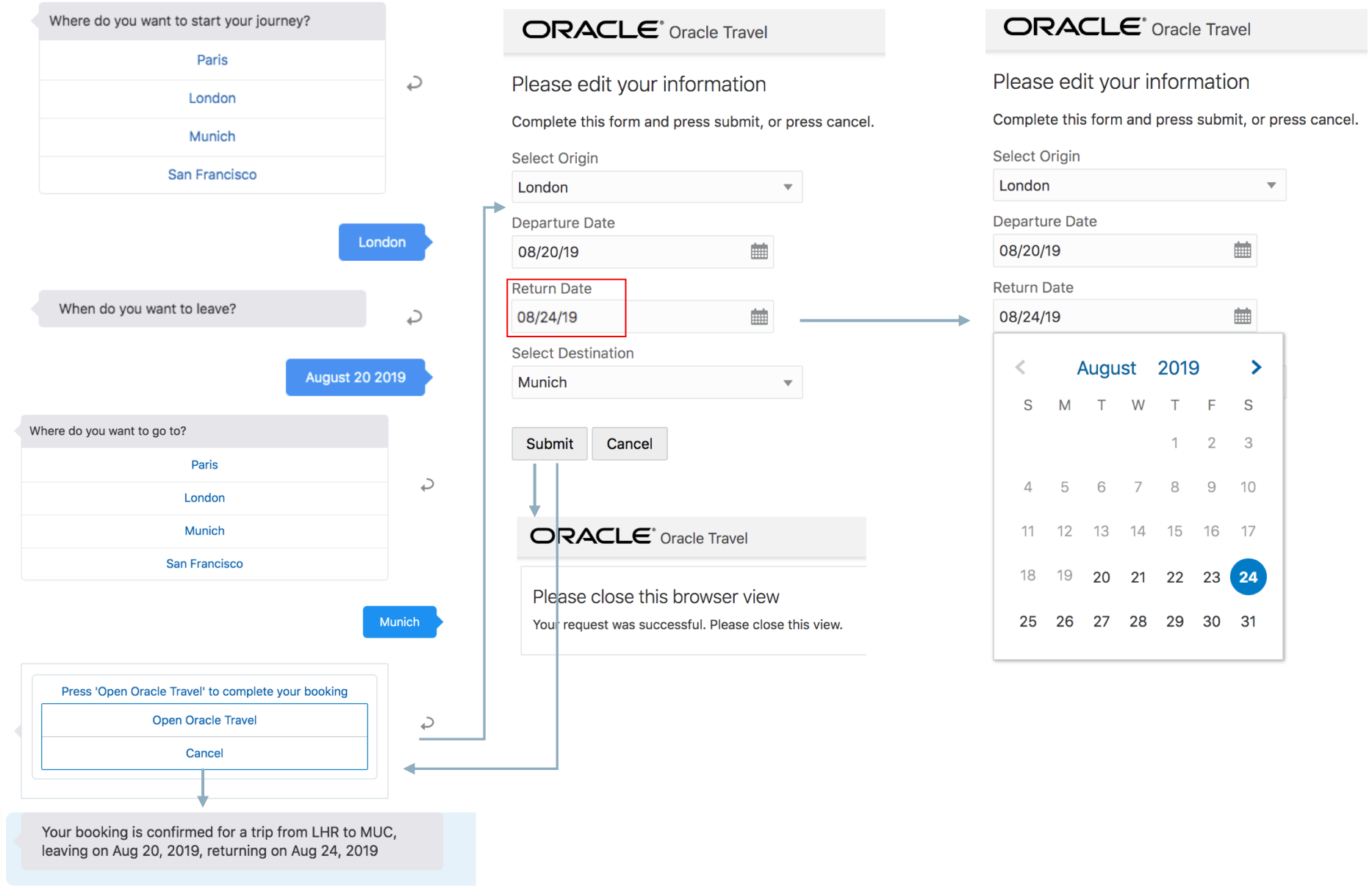


Image courtesy of Farbentek at FreeDigitalPhotos.net

Example

Oracle Travel



Topic agenda

1. When conversation is not enough
2. System.Webview
3. Local web applications
4. Remote web applications

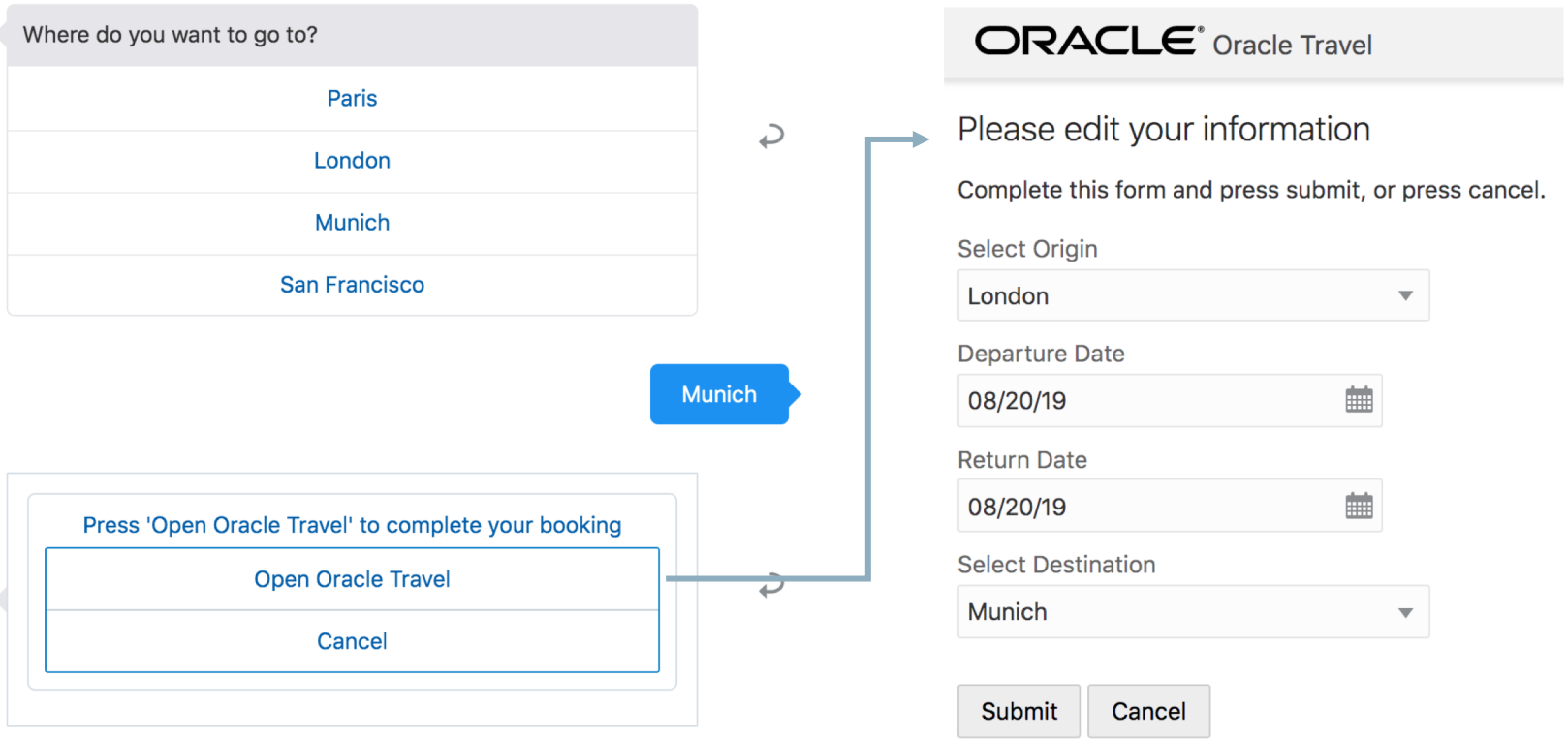
`System.Webview` component overview

Component

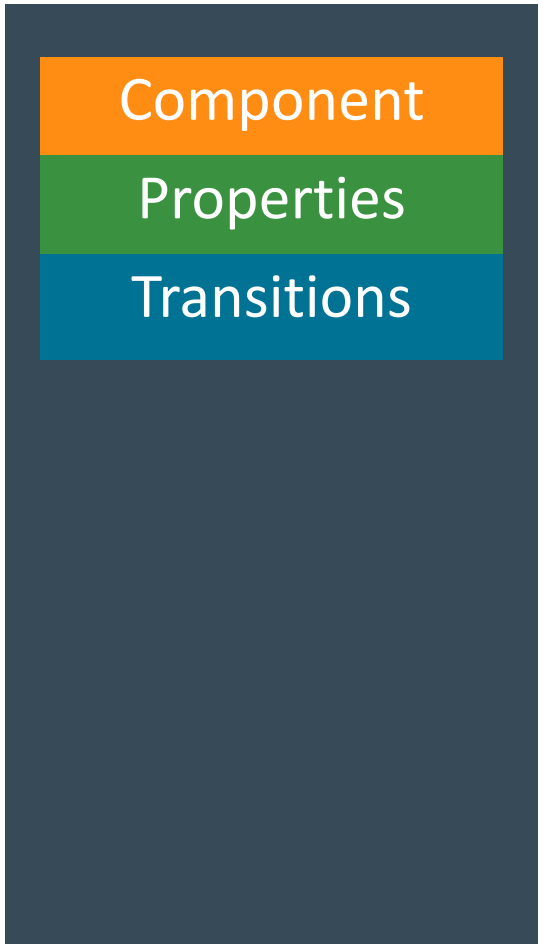
Properties

Transitions

- Displays link to launch web application in separate browser tab (web) or webview (mobile)

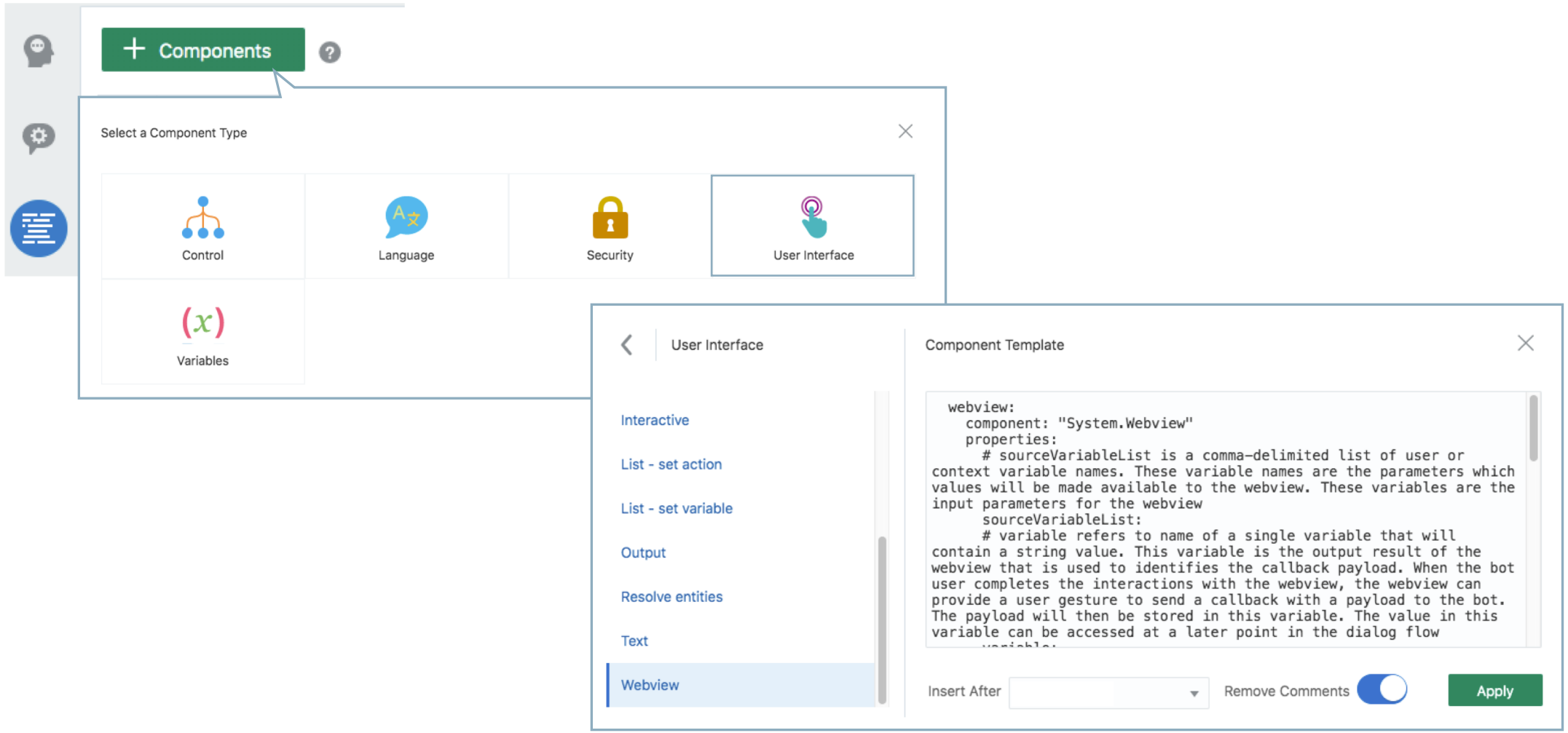


`System.Webview` component overview

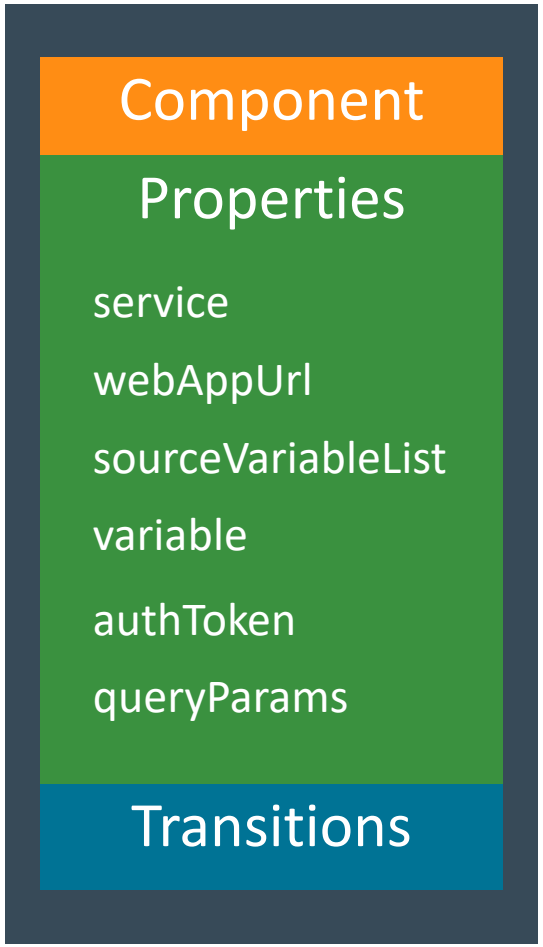


- Displays link to launch web application in separate browser tab (web) or webview (mobile)
  - Remote web application
  - Locally deployed Single Page Application (SPA)
- Passes parameters from bot to web application (optional)
- Web application uses callback URL to return to bot
  - Optionally, passes data back

Using the component template



`System.Webview` component properties



- service
  - Named configuration for local or remote app access
- webAppUrl
  - Remote web application URL (legacy property)
- variable
  - receives data object returned from web application (callback)
- authToken
  - If remote application requires authentication
- queryParams (optional)
  - JSON formatted key-value pairs added to webAppUrl

`System.Webview` component transitions

- Use **next** transition to navigate to next state upon web application success
- Use **return** transition to exit conversation upon web application success
- Use **error** transition to handle component exceptions
- Use actions transitions
  - **Cancel** to handle case where users presses "cancel" link
  - **textReceived** to handle case in which user types a text

Topic agenda

1. When conversation is not enough
2. System.Webview
3. Local web applications
4. Remote web applications
5. Remote web app vs. local web app

Like custom components, single page applications can be hosted within a skill

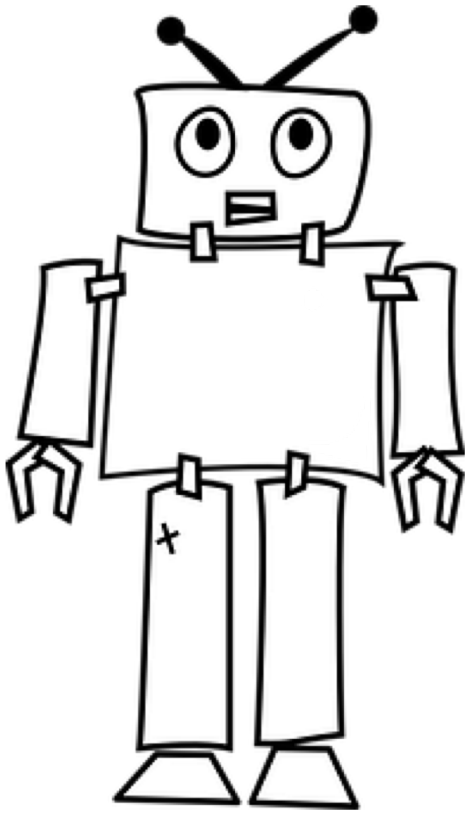
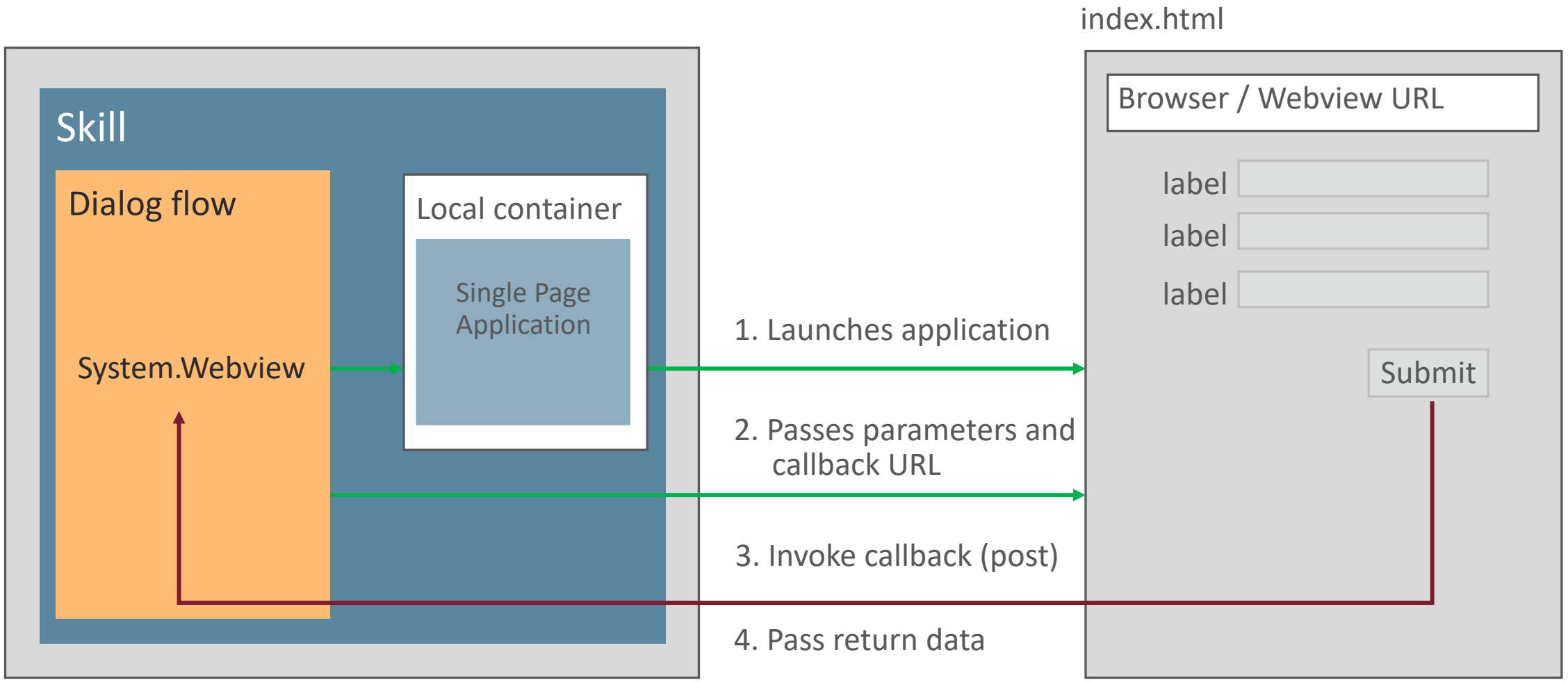


Image courtesy of pixabay.com

Single Page Application (SPA)

- Client side web applications that load a single HTML page
  - Dynamically updated in response to user interaction
  - No constant page reloads
  - Uses JavaScript, Ajax and HTML5
- Frameworks
  - Oracle JET, Oracle Visual Builder Cloud Service, Angular, React, etc.
  - Model–View–ViewModel (MVVM) architectures
  - Separation of data logic from UI definition and UI logic

Local invocation architecture



Parameters passed from the skill to the web application are accessible through JavaScript

```
window.webviewParameters['parameters'];
```

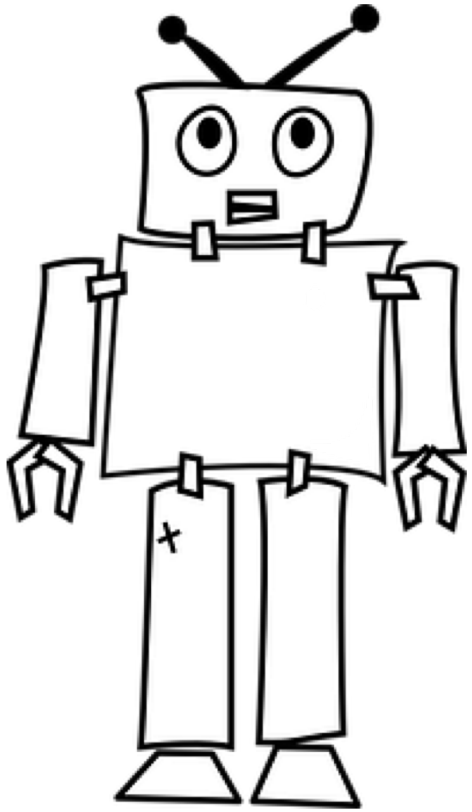


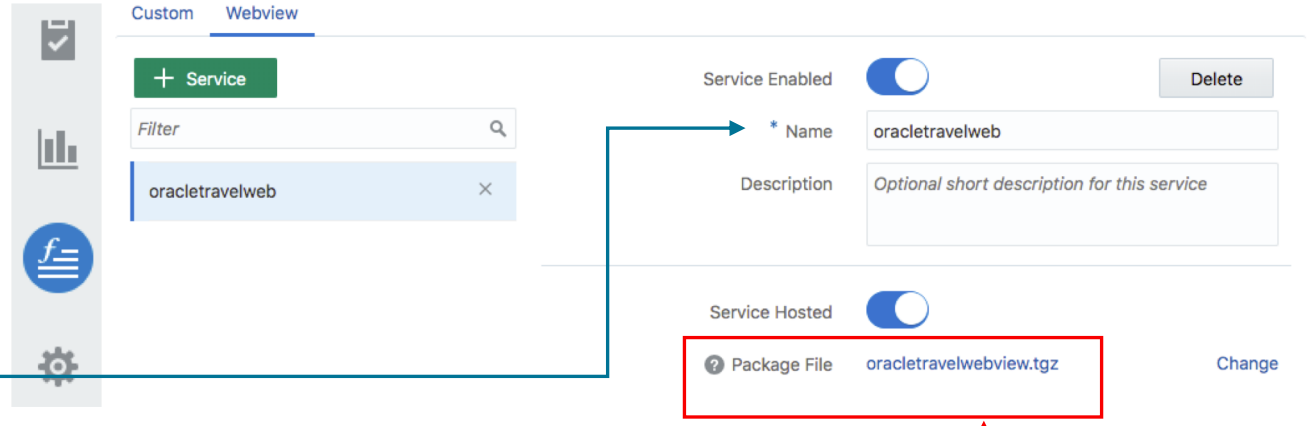
Image courtesy of Farbentek at FreeDigitalPhotos.net

SPA Application Reference

Skill

Dialog Flow

```
callWebview:
  component: "System.Webview"
  properties:
    service: "oracletravelweb"
    sourceVariableList: "origin,destination"
    variable: "webviewresponse"
    prompt: "Press 'Open Oracle Travel' ..."
    linkLabel: "Open Oracle Travel"
    cancelLabel: "Cancel"
  transitions:
    next: "handleResponse"
    actions:
      textReceived: "onCancel"
      cancel: "onCancel"
```



- Package web application
  - web folder
  - tar -zcvf <name>.tgz *
- Upload per drag and drop

SPA Application Parameter Passing

Skill

Dialog Flow

```
callWebview:
  component: "System.Webview"
  properties:
    service: "oracletravelweb"
    sourceVariableList: "origin,destination"
    variable: "webviewresponse"
    prompt: "Press 'Open Oracle Travel' …"
    linkLabel: "Open Oracle Travel"
    cancelLabel: "Cancel"
  transitions:
    next: "handleResponse"
    actions:
      textReceived: "onCancel"
      cancel: "onCancel"
```

```
{  parameters": [
   {
    "key":"origin",
    "value":"LHR"
   }, {
    "key":"destination",
    "value":"SFO"
   }, {
       "key":"webview.onDone",
       "value":"http://<url>/connectors/v1/callback?state=cb5443 . .. 2c"
   }
]}
```

Single Page Application

```
let webviewParameters = window.webviewParameters !=
null?window.webviewParameters['parameters'] : null;
```

The **webview.onDone** parameter is automatically added to the payload and passes the skill's **callback URL** property to the web application

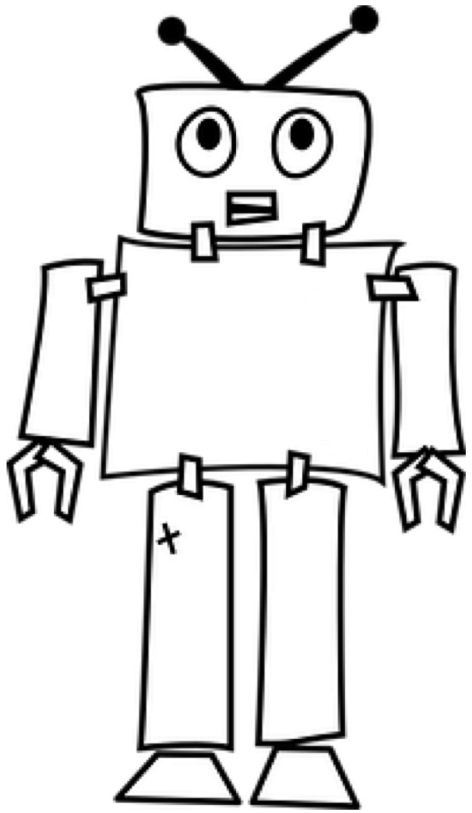


Image courtesy of pixabay.com

SPA Application Response

Skill

Dialog Flow

```
callWebview:
  component: "System.Webview"
  properties:
    service: "oracletravelweb"
    sourceVariableList: "origin,destination"
    variable: "webviewresponse"
    prompt: "Press 'Open Oracle Travel' ..."
    linkLabel: "Open Oracle Travel"
    cancelLabel: "Cancel"
  transitions:
    next: "handleResponse"
    actions:
      textReceived: "onCancel"
      cancel: "onCancel"
```

```
let data = {};
data.origin = self.origin();
data.destination = self.destination();
data.departureDate = (new Date(self.departureDate())).getTime();
data.returnDate = (new Date(self.returnDate())).getTime();
let webViewCallback = queryParameters.get('webview.onDone');

$.post(webViewCallback,JSON.stringify(data),function(data,status){
  if(status == "success"){
    console.info("success");
  }
  else {
    console.error(" error status: " + status);
  }
});
```

```
${webviewresponse.value.origin}
${webviewresponse.value.destination}
${webviewresponse.value.departureDate?numberToDate},
${webviewresponse.value.returnDate?numberToDate}"
```

When registering web applications in a skill, bot designers cannot see the data structure returned by the app.

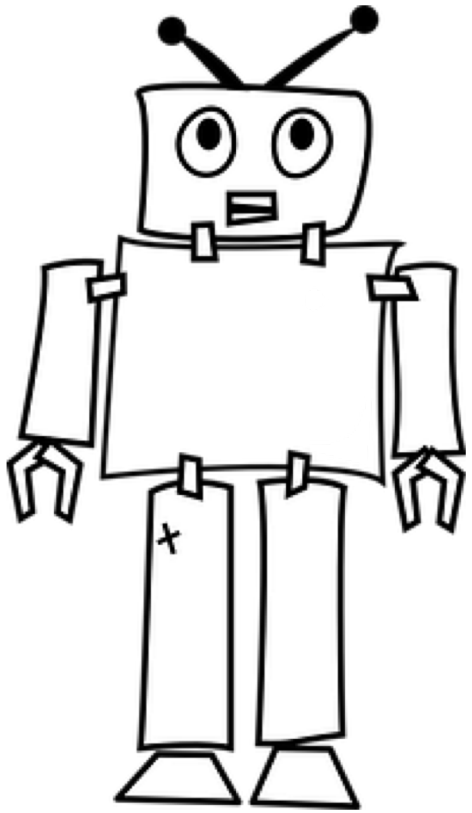
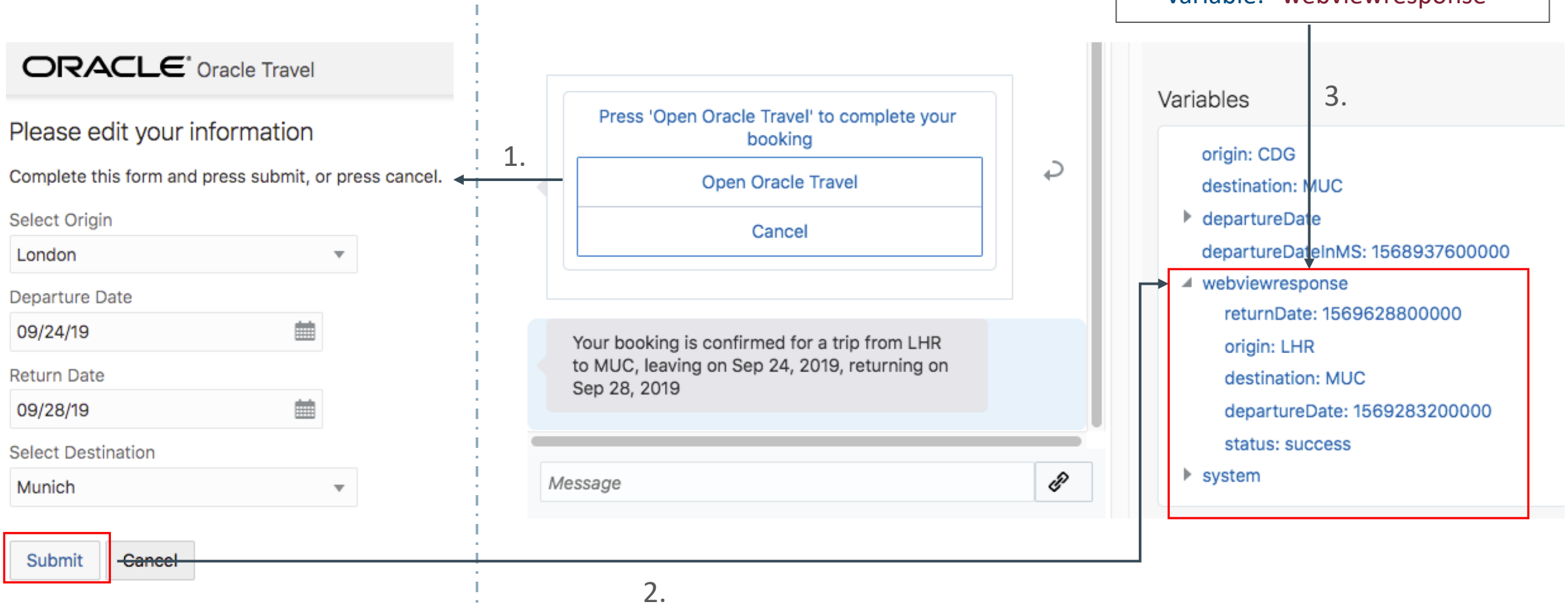


Image courtesy of pixabay.com

Debugging using the conversation tester

```
callWebview:
  component: "System.Webview"
  properties:
    ...
    variable: "webviewresponse"
```

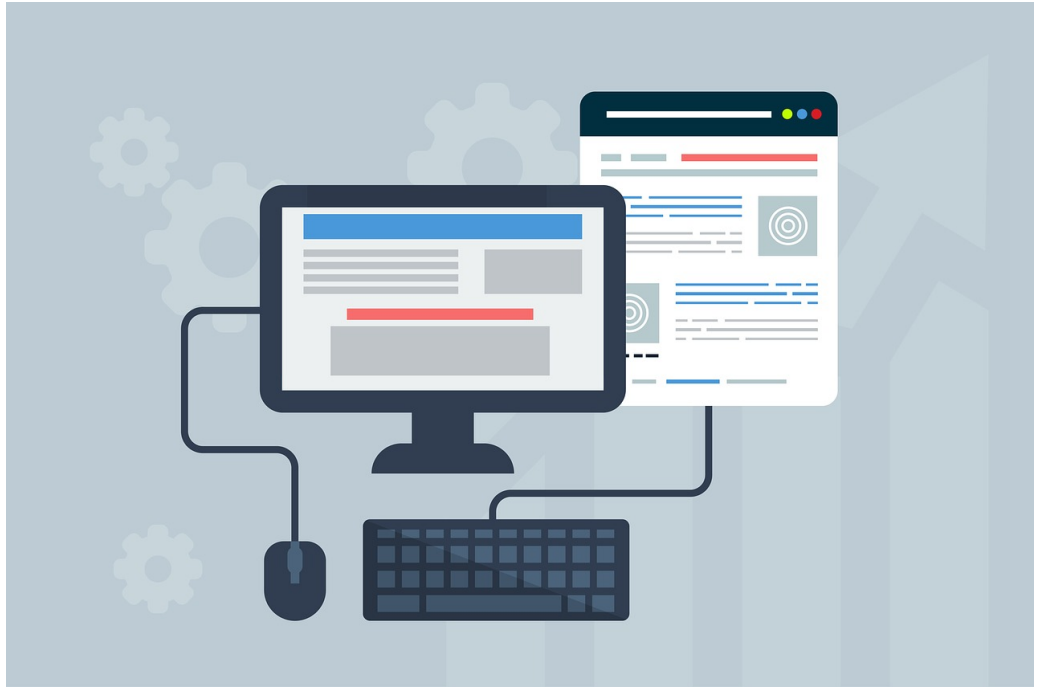


Topic agenda

1. When conversation is not enough
2. System.Webview
3. Local web applications
4. Remote web applications
5. Remote web app vs. local web app

Remote web applications

- Existing enterprise web applications
  - Leverage functionality of existing web applications
    - E.g. payment, approvals
- Applications specifically built for use with chatbots
  - Remote deployment because of
    - Data integration requirements
    - Security requirements
    - Central administration requirements



Remote web applications are **launched through an intermediary service** that prepares the web application request.

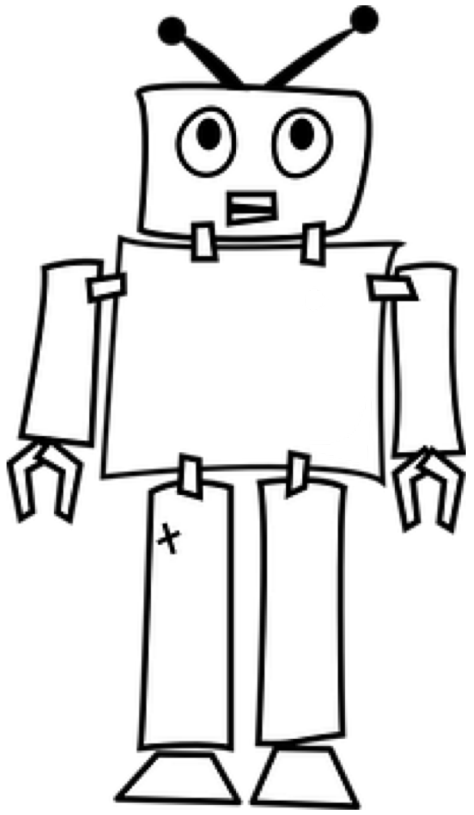
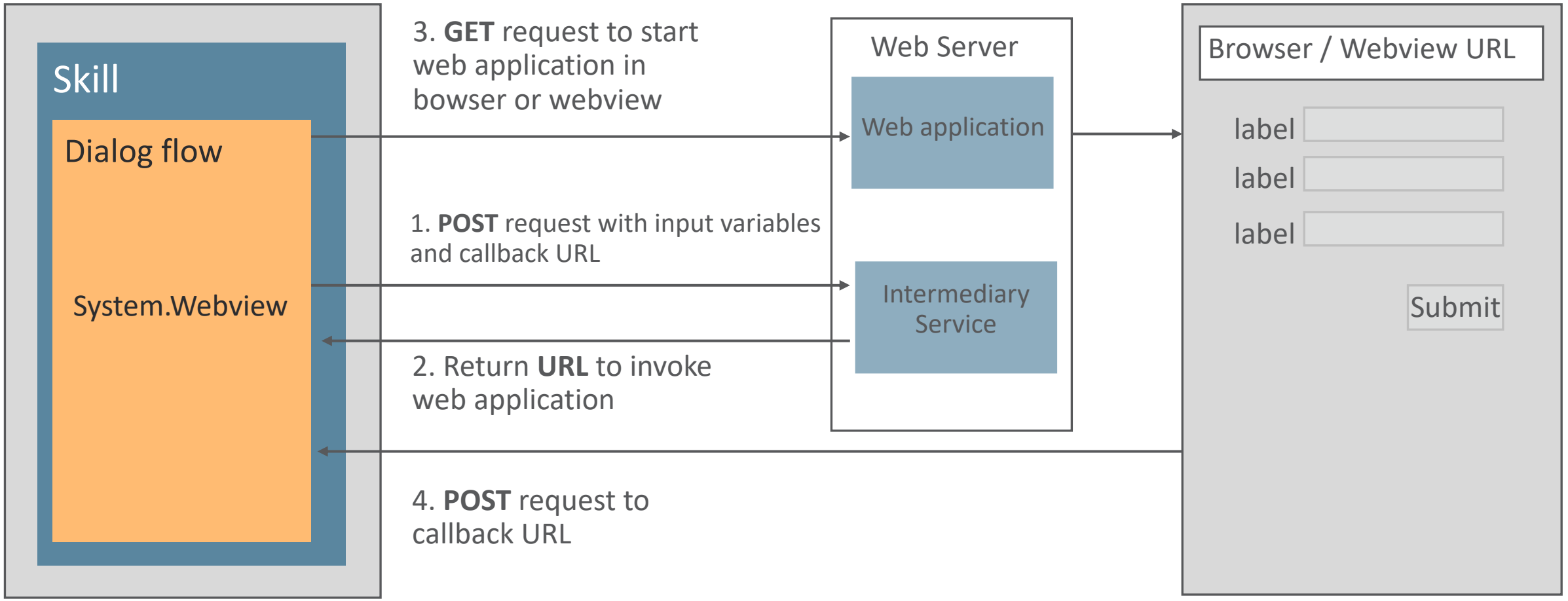
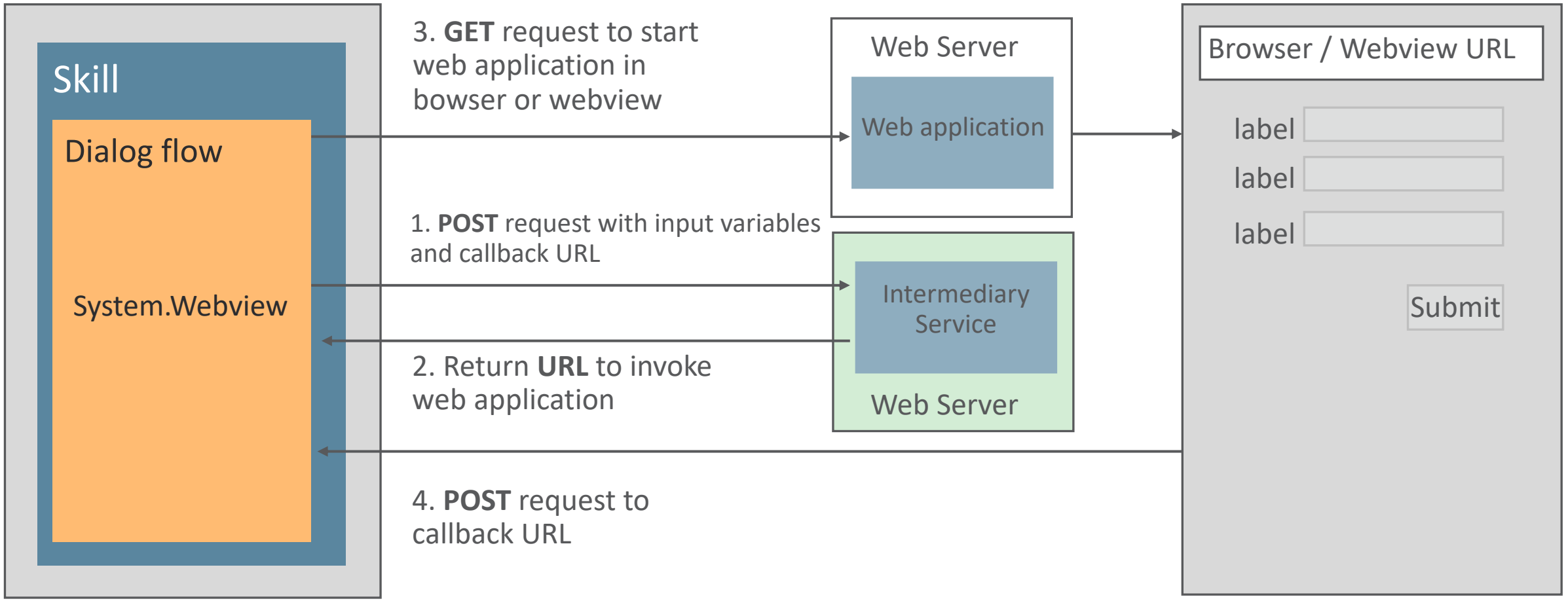


Image courtesy of pixabay.com

Remote invocation architecture



Alternative remote invocation architecture

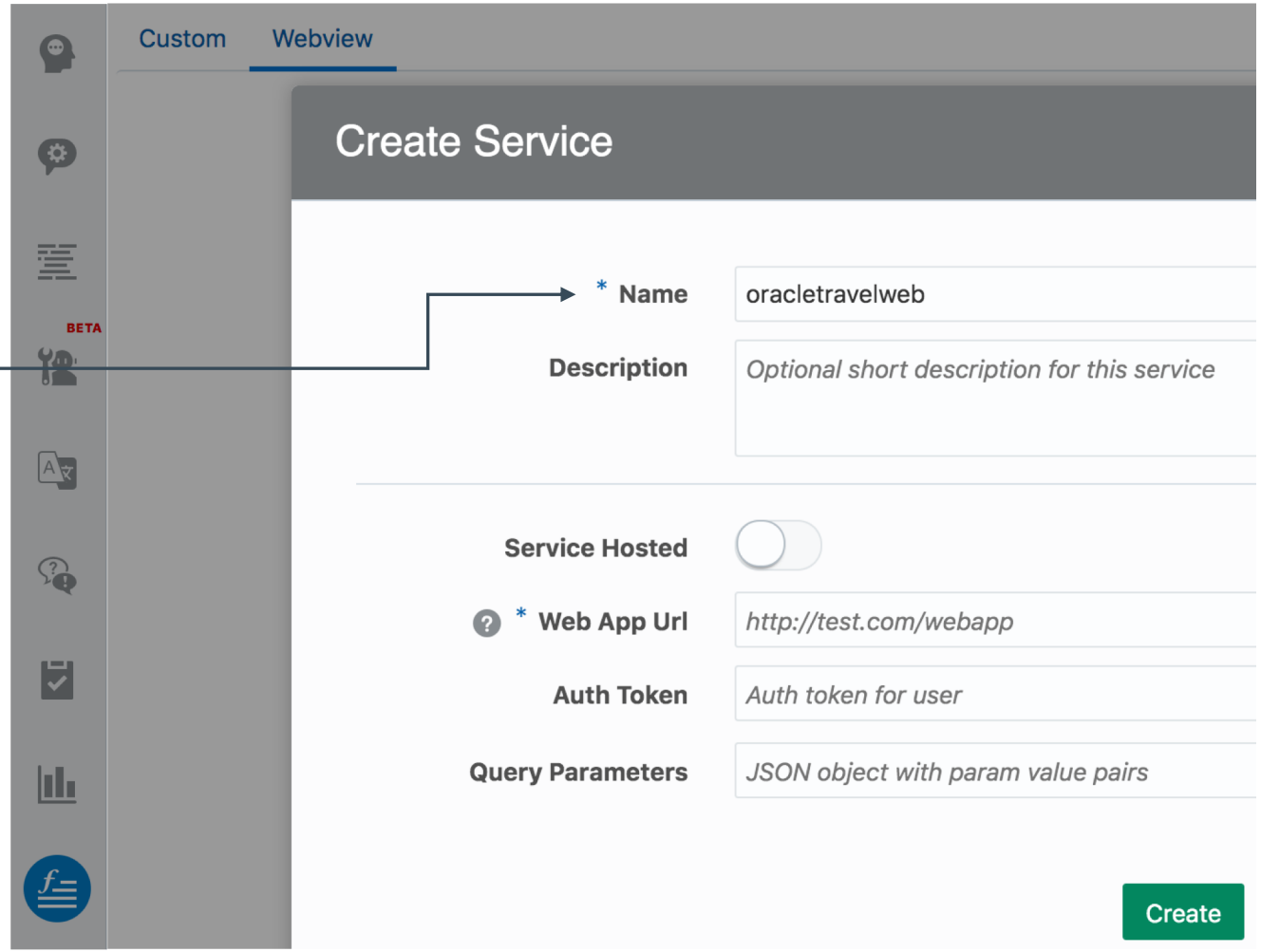


Intermediary service configuration

Skill

Dialog Flow

```
callWebview:
  component: "System.Webview"
  properties:
    service: "oracletravelweb"
    sourceVariableList: "origin,destination"
    variable: "webviewresponse"
    prompt: "Press 'Open Oracle Travel' ..."
    linkLabel: "Open Oracle Travel"
    cancelLabel: "Cancel"
  transitions:
    next: "handleResponse"
    actions:
      textReceived: "onCancel"
      cancel: "onCancel"
```



Intermediary service

- Prepares web application access
  - Creates web application access URL with all request parameters required
    - Chatbot parameters
    - Application specific parameters
- Service can be on same web server or different web server than the web application
  - If on the same server, may be able to save bot request parameters into session, thus avoiding issues with long URL
    - E.g. Internet Explorer supports 2,048 characters, minus the number of characters in the actual path
  - If on different servers, all web application request parameters need to be encoded as query parameters to the web application redirect URL sent to the bot

Payload sent to intermediary service

- POST call with JSON object containing key-value pairs
  - Keys are the names of parameters referenced in the `System.Webview` **sourceVariableList** property and "webview.onDone"

Payload of Travel Example

```
{
 "parameters": [{
  "value": "CDG",
  "key": "origin"
 }, {
  "value": "MUC",
  "key": "destination"
 }, {
  "value": "https://<url>:443/connectors/v1/callback?state= cb5443 . .. 2c ",
  "key": "webview.onDone"
}]
```

Intermediary service response payload

- Response from the intermediary service must be a JSON payload
- Payload contains single property **webview.url** with URL to remote web application
- Service developer defines the webview.url content
  - Can also chance parameter names

Service response of Travel Example

```
{
  "webview.url": "https://<app url>?callbackUrl=https://botphx1.botmxp.ocp.oraclecloud.com:443/
    connectors/v1/callback?state=7648f4288b&origin=CDG&destination=MUC
}
```

Web application response

- Web application invokes callback URL sent with the initial request to return control to the bot and to optionally send data as a JSON payload

```
let data = {};
data.origin = self.origin();
data.destination = self.destination();
data.departureDate =
    (new Date(self.departureDate())).getTime();
data.returnDate =
    (new Date(self.returnDate())).getTime();

$.post(webViewCallback,JSON.stringify(data));
```



```
{
 "origin":"CDG",
 "destination":"MUC",
 "departureDate":1568937600000,
 "returnDate":1568937600000
}
```

Property names in the web application response payload don't need to match variable names in the dialog flow.

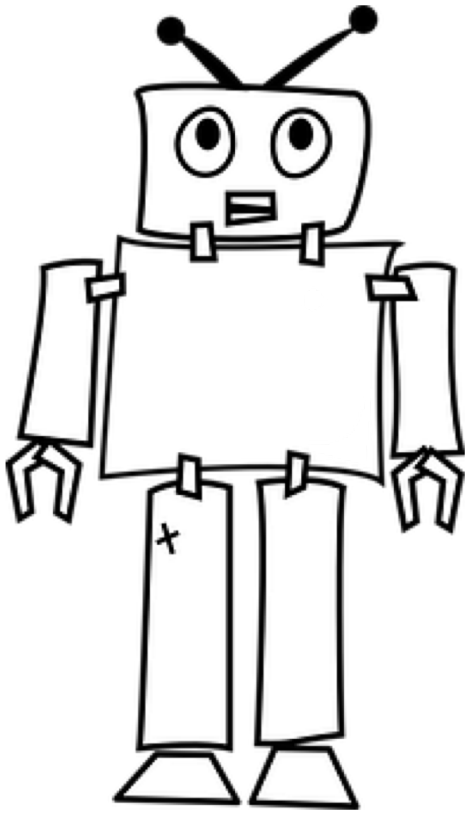


Image courtesy of pixabay.com

Topic agenda

1. When conversation is not enough
2. System.Webview
3. Local web applications
4. Remote web applications
5. Remote web app vs. local web app

Remote vs. local web application

Remote web application

- Application needs to be modified or created to work with webview
- Good for Integrating existing web applications or functionality
- Security through web application authentication
- Leverage server side infrastructure
  - E.g. environment variables

Local application

- Application created for deployment to local container
- Easy support for structured data input and missing widgets (e.g. date picker support)
- No application security
- Good to use for self-contained skills deployed via skill store

The limitation of web applications displayed in webviews is that they don't work with voice.

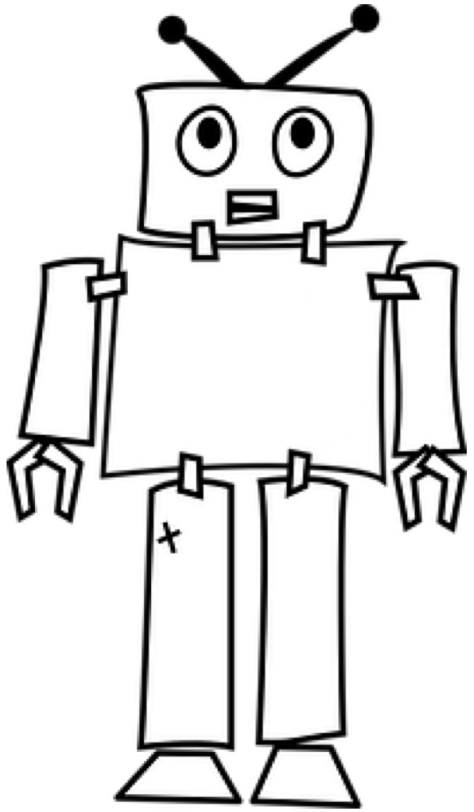


Image courtesy of Farbentek at FreeDigitalPhotos.net

Integrated Cloud

Applications & Platform Services


ORACLE®

Oracle Digital Assistant Hands-On