Skill: SkinGuide

December 2, 2021

[1]:
```
# -*- coding: utf-8 -*-

# This sample demonstrates handling intents from an Alexa skill using the Alexa␣
 ↪Skills Kit SDK for Python.
# Please visit https://alexa.design/cookbook for additional examples on␣
 ↪implementing slots, dialog management,
# session persistence, api calls, and more.
# This sample is built using the handler classes approach in skill builder.
import logging
import ask_sdk_core.utils as ask_utils

from ask_sdk_core.skill_builder import SkillBuilder
from ask_sdk_core.dispatch_components import AbstractRequestHandler
from ask_sdk_core.dispatch_components import AbstractExceptionHandler
from ask_sdk_core.handler_input import HandlerInput

from ask_sdk_model import Response

logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)


class LaunchRequestHandler(AbstractRequestHandler):
    """Handler for Skill Launch."""
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool

        return ask_utils.is_request_type("LaunchRequest")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Willkommen bei Skin Guide! Schön, dass du da bist! Hilf␣
 ↪mir dich besser kennenzulernen. Wie heißt du?"

        return (
            handler_input.response_builder
                .speak(speak_output)
```

```
                .ask(speak_output)
                .response
        )


class HelloWorldIntentHandler(AbstractRequestHandler):
    """Handler for Hello World Intent."""
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.is_intent_name("HelloWorldIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Hello World!"

        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
 ↪the user to respond")
                .response
        )


class CancelOrStopIntentHandler(AbstractRequestHandler):
    """Single handler for Cancel and Stop Intent."""
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return (ask_utils.is_intent_name("AMAZON.CancelIntent")(handler_input)␣
 ↪or
                ask_utils.is_intent_name("AMAZON.StopIntent")(handler_input))

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Goodbye!"

        return (
            handler_input.response_builder
                .speak(speak_output)
                .response
        )


class SessionEndedRequestHandler(AbstractRequestHandler):
    """Handler for Session End."""
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
```

```
        return ask_utils.is_request_type("SessionEndedRequest")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response

        # Any cleanup logic goes here.

        return handler_input.response_builder.response


class IntentReflectorHandler(AbstractRequestHandler):
    """The intent reflector is used for interaction model testing and debugging.
    It will simply repeat the intent the user said. You can create custom␣
↪handlers
    for your intents by defining them above, then also adding them to the␣
↪request
    handler chain below.
    """
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.is_request_type("IntentRequest")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        intent_name = ask_utils.get_intent_name(handler_input)
        speak_output = "You just triggered " + intent_name + "."

        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )


class CatchAllExceptionHandler(AbstractExceptionHandler):
    """Generic error handling to capture any syntax or routing errors. If you␣
↪receive an error
    stating the request handler chain is not found, you have not implemented a␣
↪handler for
    the intent being invoked or included it in the skill builder below.
    """
    def can_handle(self, handler_input, exception):
        # type: (HandlerInput, Exception) -> bool
        return True
```

```
    def handle(self, handler_input, exception):
        # type: (HandlerInput, Exception) -> Response
        logger.error(exception, exc_info=True)

        speak_output = "Sorry, I had trouble doing what you asked. Please try␣
↪again."

        return (
            handler_input.response_builder
                .speak(speak_output)
                .ask(speak_output)
                .response
        )


class HauttypBekanntHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_hauttyp_inhaltsstoffe("HauttypBekanntIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Schön das du es weißt. Welchen Hauttyp hast du {name}?␣
↪Es gibt folgende Hauttypen: Normale Haut, fettige Haut, trockene Haut und␣
↪Mischhaut."

        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )


class HauttypBekanntNormaleHautHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_hauttyp_bekannt_normale_haut("HauttypBekanntNormaleHautIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
```

```
        speak_output = "Super! Für deine normale Haut kann ich dir folgende␣
↪Inhaltsstoffe empfehlen:Polyhydroxysäure, weißer Tee, Squalan, Mandelsäure,␣
↪Panthenol, Arganöl, Hyaluronsäure, Glykolsäure und Ceramide.Möchtest du eine␣
↪passende DIY-Maske für deine normale Haut wissen?"

        return (
            handler_input.response_builder
                .speak(speak_output)
                .ask(speak_output)
                .response
        )


class HauttypBekanntFettigeHautHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_hauttyp_bekannt_fettige_haut("HauttypBekanntFettigeHautIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Super! Für deine fettige Haut kann ich folgende␣
↪Inhaltsstoffe empfehlen:Saliclysäure, Niacinamid, Hagebutten Kernöl,␣
↪Mandelsäure, Tonerde, Teebaumöl, Benzoylperoxid, Urea und Squalan.Möchtest␣
↪du eine passende DIY-Maske für deine fettige Haut wissen?"

        return (
            handler_input.response_builder
                .speak(speak_output)
                .ask(speak_output)
                .response
        )

class HauttypBekanntTrockeneHautHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_hauttyp_bekannt_trockene_haut("HauttypBekanntTrockeneHautIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Super! Für deine trockene Haut kann ich dir folgende␣
↪Inhaltsstoffe empfehlen:Milchsäure, Urea, Glykolsäure, Ceramide, Sheabutter,␣
↪Hyaluronsäure, Vaseline und Arganöl.Möchtest du eine passende DIY-Maske für␣
↪deine trockene Haut wissen?"

        return (
```

```
            handler_input.response_builder
                .speak(speak_output)
                .ask(speak_output)
                .response
        )


class HauttypBekanntMischhautHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_hauttyp_bekannt_mischhaut("HauttypBekanntMischhautIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Super! Für deine Mischhaut kann ich dir folgende␣
↪Inhaltsstoffe empfehlen:Salicylsäure, Glycerin, Wildrosenöl, Hyaluronsäure,␣
↪Niacinamid, Jojoba-Öl, Glykolsäure, Aloe Vera und Squalan.Möchtest du eine␣
↪passende DIY-Maske für deine Mischhaut wissen?"

        return (
            handler_input.response_builder
                .speak(speak_output)
                .ask(speak_output)
                .response
        )

class HauttypUnbekanntHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_hauttyp_inhaltsstoffe("HauttypUnbekanntIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Okay, dann finden wir das gemeinsam heraus! Wird deine␣
↪Haut im Laufe des Tages fettig, trocken, fettig und trocken oder unverändert?
↪ "

        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )
```

```
class HauttypUnbekanntFettigeHautHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_hauttyp_bekannt_fettige_haut("HauttypUnbekanntFettigeHautIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Beobachtest Du auf deiner Haut Rötungen oder Juckreiz,␣
↪Mitesser, alle drei zusammen oder keines dieser drei Antwortmöglichkeiten? "

        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntTrockeneHautHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_hauttyp_bekannt_trockene_haut("HauttypUnbekanntTrockeneHautIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Beobachtest Du auf deiner Haut Rötungen oder Juckreiz,␣
↪Mitesser, alle drei zusammen oder keines dieser drei Antwortmöglichkeiten? "

        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )


class HauttypUnbekanntNormaleHautHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_normale_haut("HauttypUnbekanntNormaleHautIntent")(handler_input)
```

```
    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Beobachtest Du auf deiner Haut Rötungen oder Juckreiz,␣
↪Mitesser, alle drei zusammen oder keines dieser drei Antwortmöglichkeiten? "

        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )


class HauttypUnbekanntMischhautHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_mischhaut_antwort_eins("HauttypUnbekanntNormaleHautIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Beobachtest Du auf deiner Haut Rötungen oder Juckreiz,␣
↪Mitesser, alle drei zusammen oder keines dieser drei Antwortmöglichkeiten? "

        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntFettigeHautZweiHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_fettige_haut_antwort_zwei("HauttypUnbekanntFettigeHautZweiIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Würdest du dein Hautbild eher rissig und spröde oder␣
↪glänzend oder rosig und zart einschätzen?"
        return (
            handler_input.response_builder
                .speak(speak_output)
```

```
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntTrockeneHautZweiHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_trockene_haut_antwort_zwei("HauttypUnbekanntTrockeneHautZweiIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Würdest du dein Hautbild eher rissig und spröde oder␣
↪glänzend oder rosig und zart einschätzen?"
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntNormaleHautZweiHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_normale_haut_antwort_zwei("HauttypUnbekanntNormaleHautZweiIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Würdest du dein Hautbild eher rissig und spröde oder␣
↪glänzend oder rosig und zart einschätzen?"
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )


class HauttypUnbekanntMischhautZweiHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
```

```
        return ask_utils.
↪is_mischhhaut_antwort_zwei("HauttypUnbekanntMischhautZweiIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Würdest du dein Hautbild eher rissig und spröde oder␣
↪glänzend oder rosig und zart einschätzen?"
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntFettigeDreiHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_fettige_haut_antwort_drei("HauttypUnbekanntFettigeDreiIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Betrachte mal deine Poren. Sind sie eher groß, klein,␣
↪beides oder hast du keine sichtbaren Poren?"
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )


class HauttypUnbekanntTrockeneDreiHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_trockene_haut_antwort_drei("HauttypUnbekanntTrockeneDreiIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Betrachte mal deine Poren. Sind sie eher groß, klein,␣
↪beides oder hast du keine sichtbaren Poren?"
        return (
            handler_input.response_builder
```

```
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntNormaleDreiHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_normale_haut_antwort_drei("HauttypUnbekanntNormaleDreiIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Betrachte mal deine Poren. Sind sie eher groß, klein,␣
↪beides oder hast du keine sichtbaren Poren?"
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntMischhautDreiHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_mischhaut_antwort_drei("HauttypUnbekanntMischhautDreiIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = "Betrachte mal deine Poren. Sind sie eher groß, klein,␣
↪beides oder hast du keine sichtbaren Poren?"
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntFettigeHautVierHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
```

```
        return ask_utils.
↪is_fettige_haut_antwort_vier("HauttypUnbekanntFettigeHautVierIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = ""
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntTrockeneHautVierHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_trockene_haut_antwort_vier("HauttypUnbekanntTrockeneHautVierIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = ""
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
        )

class HauttypUnbekanntNormaleHautVierHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
↪is_normale_haut_antwort_vier("HauttypUnbekanntNormaleHautVierIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = ""
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
↪the user to respond")
                .response
```

```
        )


class HauttypUnbekanntMischautVierHandler(AbstractRequestHandler):
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.
 ↪is_mischhaut_antwort_vier("HauttypUnbekanntMischautVierIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        speak_output = ""
        return (
            handler_input.response_builder
                .speak(speak_output)
                # .ask("add a reprompt if you want to keep the session open for␣
 ↪the user to respond")
                .response
        )


sb = SkillBuilder()

sb.add_request_handler(LaunchRequestHandler())
sb.add_request_handler(HelloWorldIntentHandler())
sb.add_request_handler(HelpIntentHandler())
sb.add_request_handler(CancelOrStopIntentHandler())
sb.add_request_handler(SessionEndedRequestHandler())
sb.add_request_handler(HauttypBekanntHandler())
sb.add_request_handler(HauttypBekanntNormaleHautHandler())
sb.add_request_handler(HauttypBekanntFettigeHautHandler())
sb.add_request_handler(HauttypBekanntTrockeneHautHandler())
sb.add_request_handler(HauttypBekanntMischhautHandler())
sb.add_request_handler(HauttypUnbekanntHandler())
sb.add_request_handler(HauttypUnbekanntFettigeHautHandler())
sb.add_request_handler(HauttypUnbekanntTrockeneHautHandler())
sb.add_request_handler(HauttypUnbekanntNormaleHautHandler())
sb.add_request_handler(HauttypUnbekanntMischhautHandler())
sb.add_request_handler(HauttypUnbekanntFettigeHautZweiHandler())
sb.add_request_handler(HauttypUnbekanntTrockeneHautZweiHandler())
sb.add_request_handler(HauttypUnbekanntNormaleHautZweiHandler())
sb.add_request_handler(HauttypUnbekanntMischhautZweiHandler())
sb.add_request_handler(HauttypUnbekanntFettigeDreiHandler())
sb.add_request_handler(HauttypUnbekanntTrockeneDreiHandler())
sb.add_request_handler(HauttypUnbekanntNormaleDreiHandler())
sb.add_request_handler(HauttypUnbekanntMischhautDreiHandler())
sb.add_request_handler(HauttypUnbekanntFettigeHautVierHandler())
```

```
sb.add_request_handler(HauttypUnbekanntTrockeneHautVierHandler))
sb.add_request_handler(HauttypUnbekanntNormaleHautVierHandler))
sb.add_request_handler(HauttypUnbekanntMischautVierHandler))
sb.add_request_handler(IntentReflectorHandler()) # make sure␣
↪IntentReflectorHandler is last so it doesn't override your custom intent␣
↪handlers

sb.add_exception_handler(CatchAllExceptionHandler())

lambda_handler = sb.lambda_handler()
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
/tmp/ipykernel_175/714281275.py in <module>
      6 # This sample is built using the handler classes approach in skill␣
 ↪builder.
      7 import logging
----> 8 import ask_sdk_core.utils as ask_utils
      9
     10 from ask_sdk_core.skill_builder import SkillBuilder

ModuleNotFoundError: No module named 'ask_sdk_core'
```

[ ]: