

**LEX总览**

Intents意图-意图表示用户要执行的操作。您创建自动程序以支持一个或多个相关目的。

Utterances 话语-这些是用户为了触发意图而必须键入或说出的内容的文本表示。

Prompts 提示-Lex可以为会话的不同阶段定义不同的提示。

Slots 槽- 这些是bot意图需要的数据项，以便能够完成其任务。

Fulfilment 结束-这是目的的最终状态，通常带有结束消息和一些用户的附加信息。

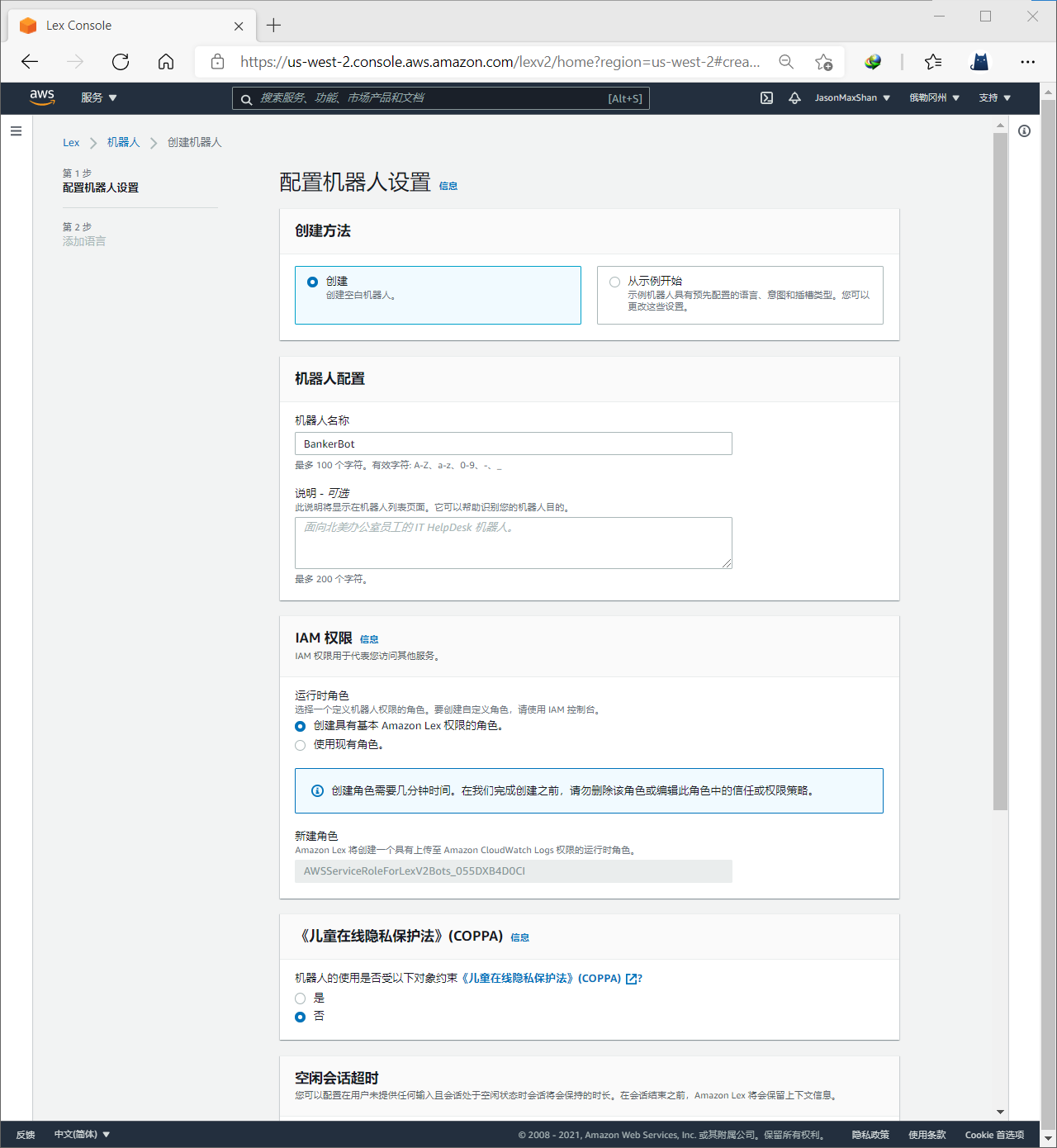
实验原网页：https://amazonlex.workshop.aws/banker-bot.html

**实验1：银行机器人**

这个介绍将创建一个简单的银行相关的聊天机器人，可以提供一些基本的银行功能，如检查余额，转移资金，或回答有关可能的贷款的问题。

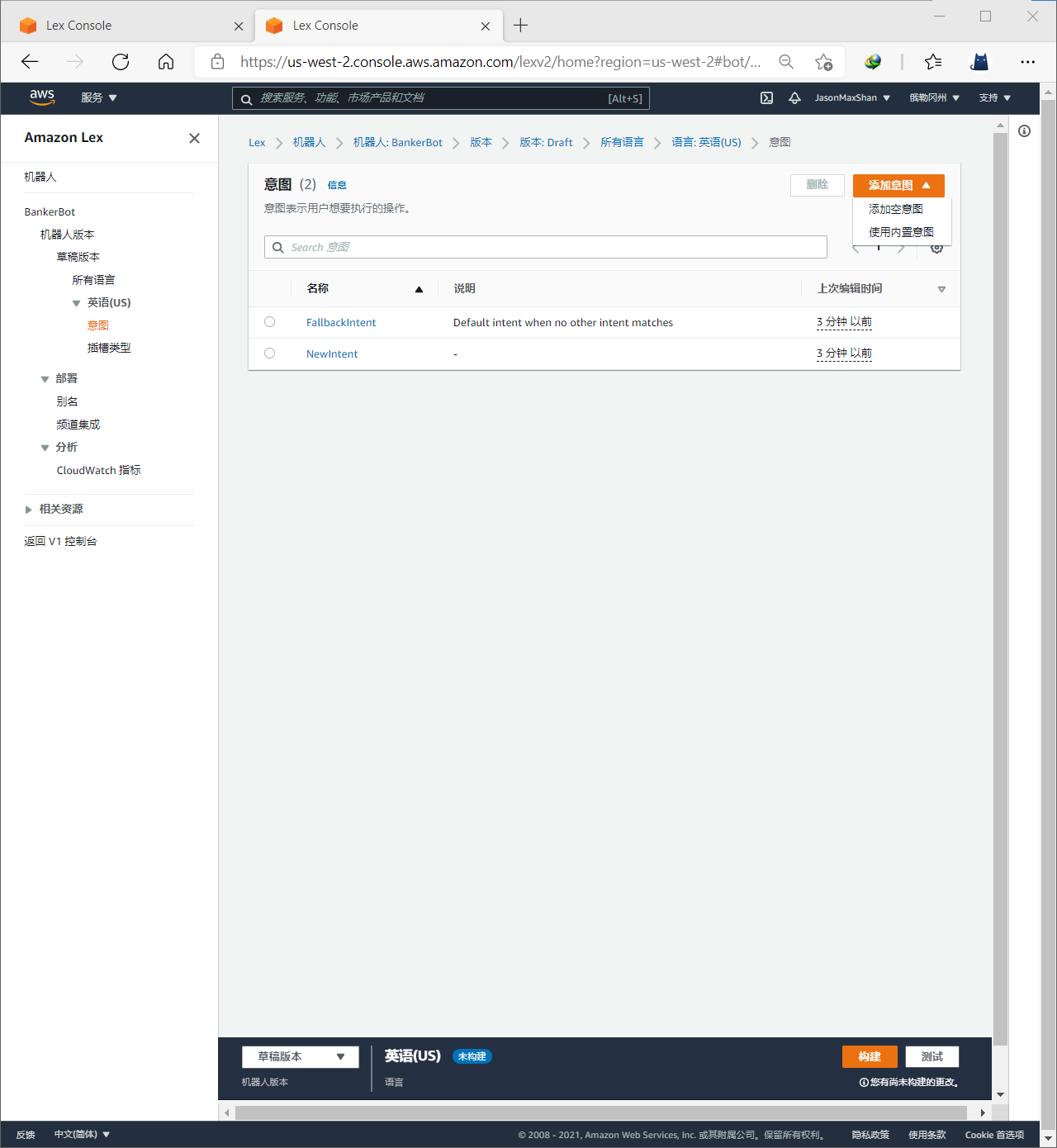
步骤1：创建对话机器人

来到AWS LEX界面，在导航栏选择Bots，注意实验使用的是V2版本，点击Create。选择Custom bot，将bot命名为BankerBot，语言选择English (US)，COPPA选择NO，其余默认，点击创建。



步骤2：创建欢迎意图

点击BankerBot，在导航栏找到BankerBot，点击Lex>机器人>机器人: BankerBot>版本>版本: Draft>所有语言>语言: 英语(US)>意图，点击添加意图，添加空意图



**意图名称**命名为Welcome

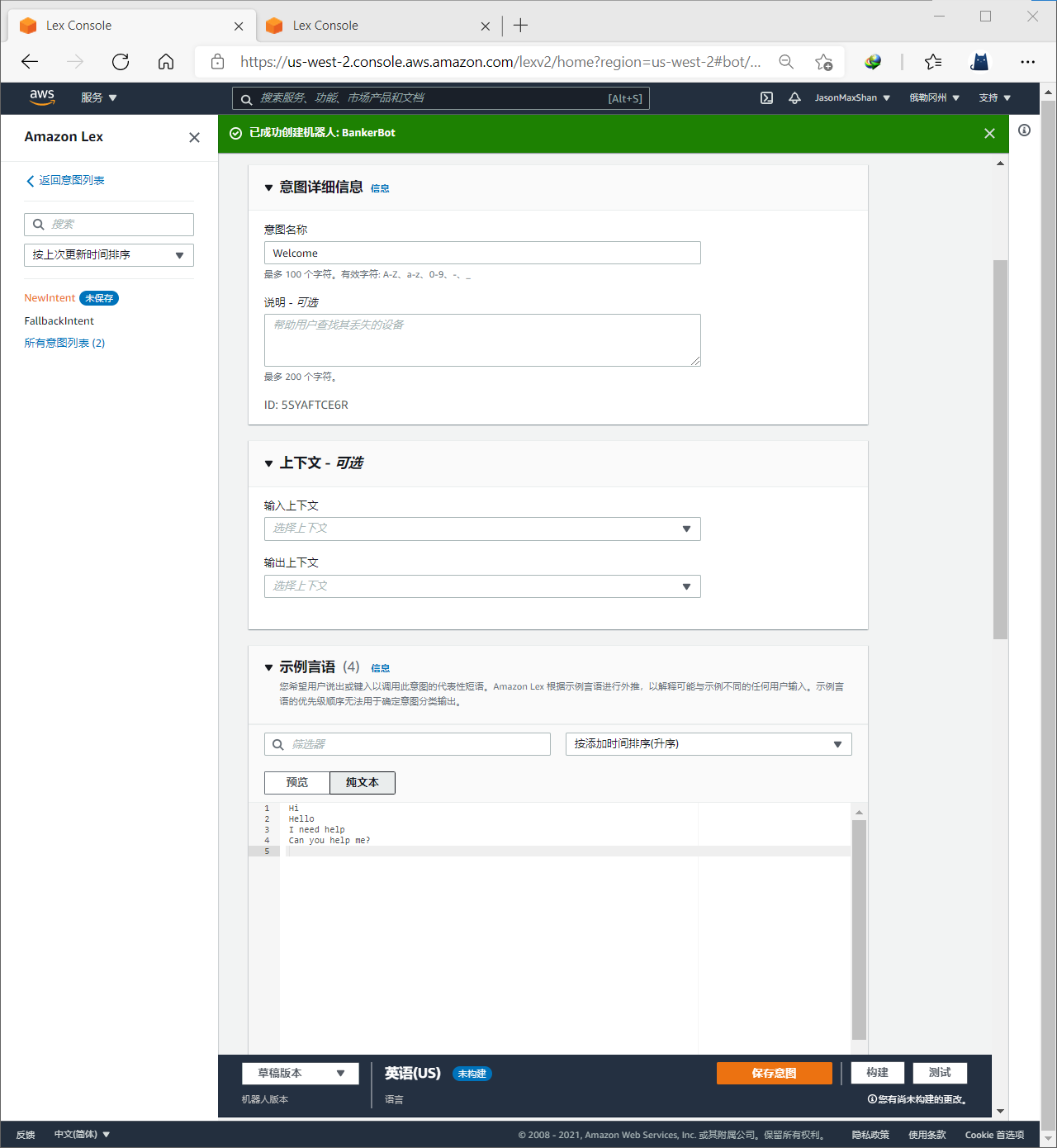
在**示例言语**框点击纯文本，添加文本：

Hi

Hello

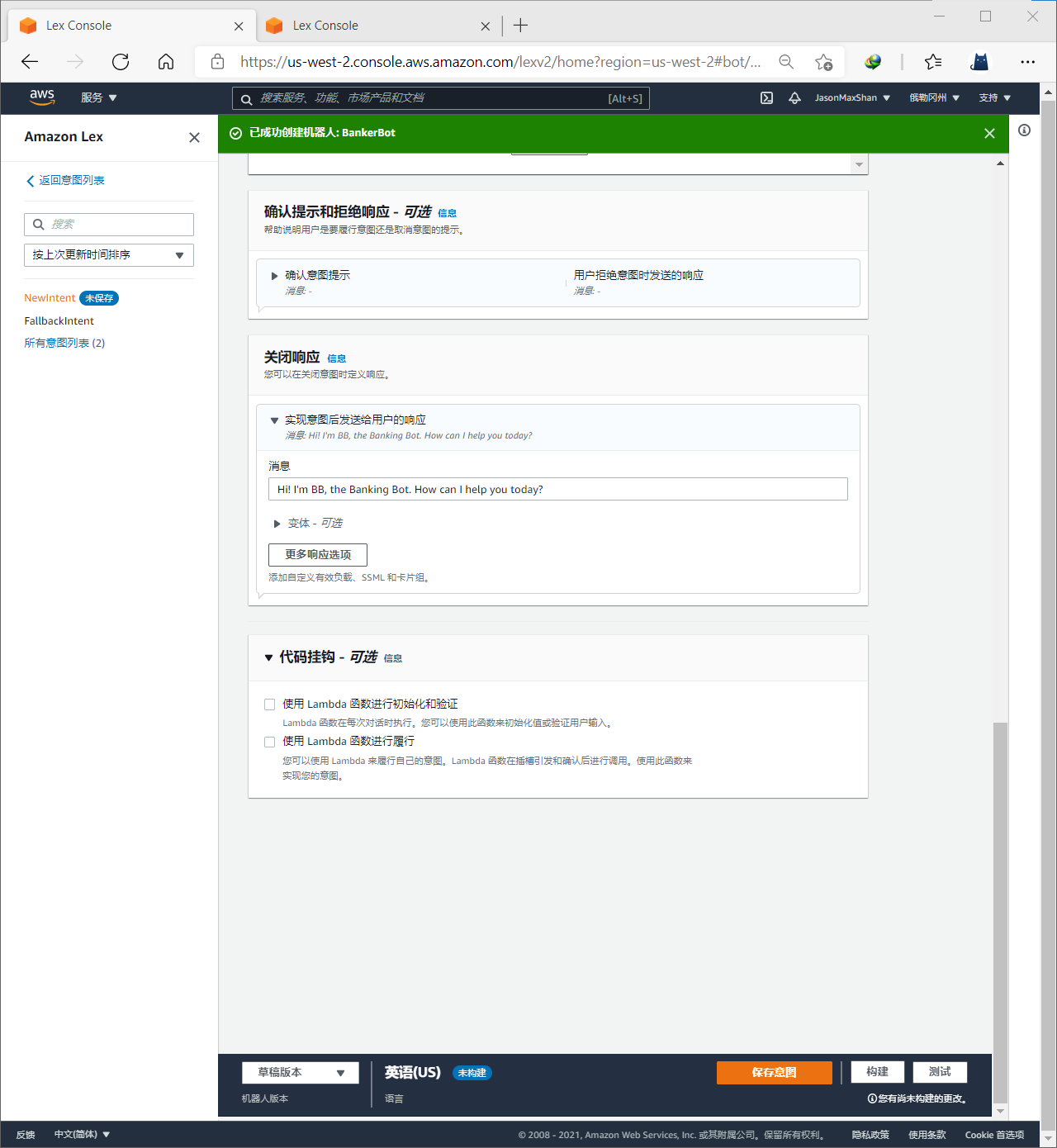
I need help

Can you help me?

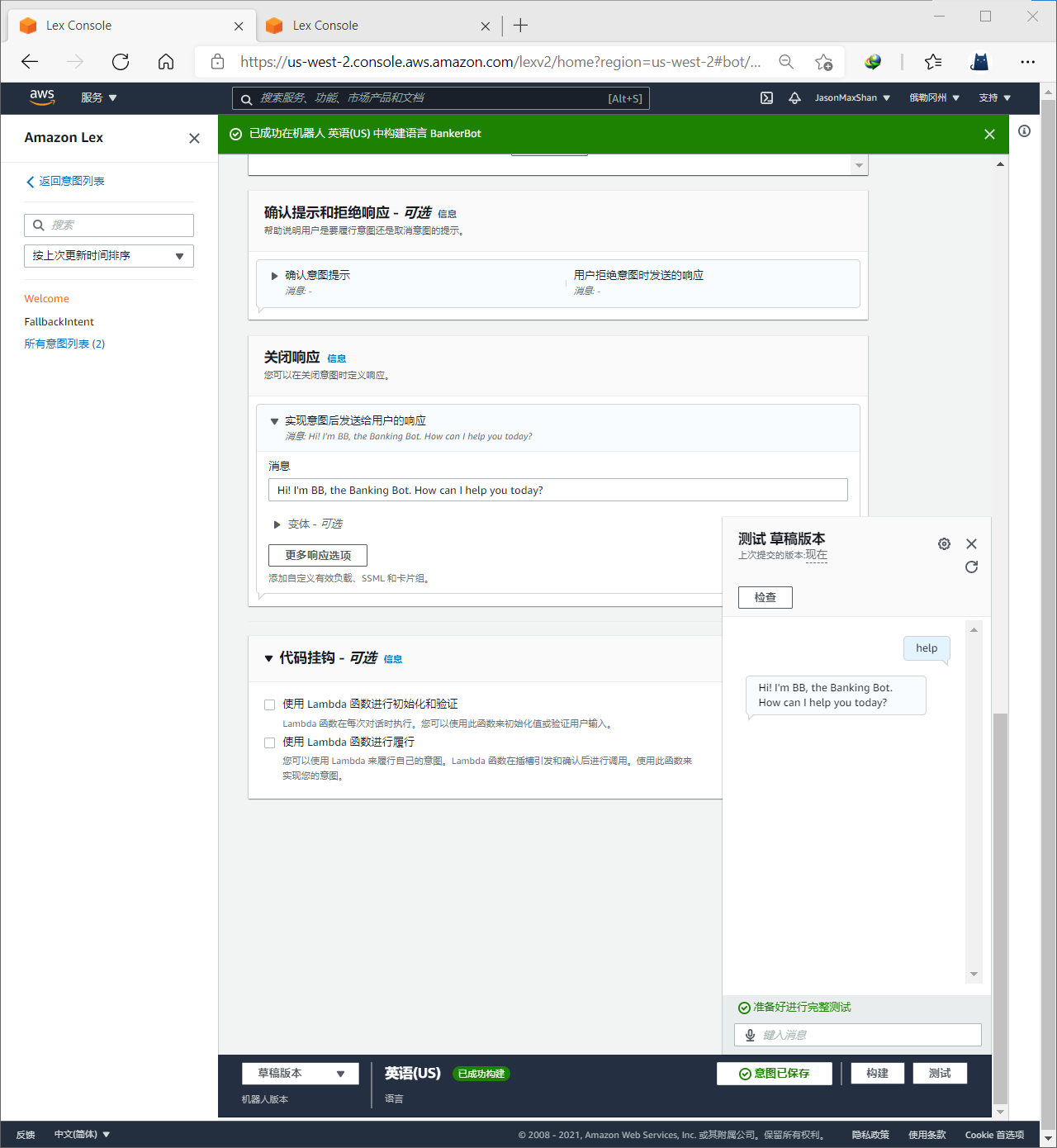


然后在**关闭响应**框中添加文本：

Hi! I'm BB, the Banking Bot. How can I help you today?



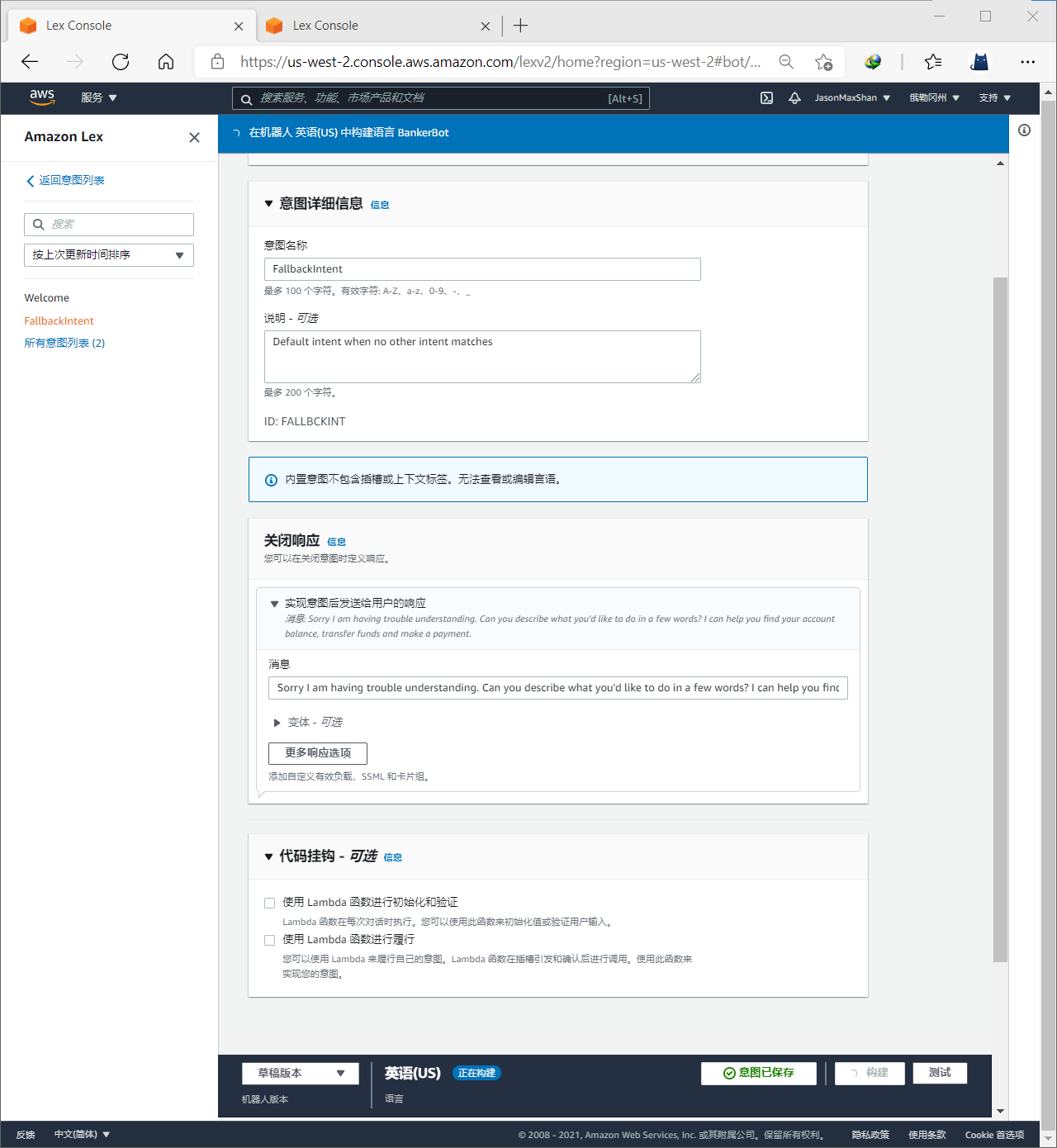
点击**保存意图，**然后点击**构建**，构建完成后点击**测试**，输入相关词语Bot就会作出反应



Bot在创建时还预生成了一个Fallback意图，我们需要更新那个意图，使得这个意图给用户一个更好的消息，应该是要求他们以更好的方式重新措辞他们的问题。

点击FallbackIntent，在**关闭响应**栏添加文本：

Sorry I am having trouble understanding. Can you describe what you'd like to do in a few words? I can help you find your account balance, transfer funds and make a payment.



然后点击**保存意图**，**构建**

步骤3：创建CHECKBALANCE意图

为账户类型创建自定义插槽

BankerBot将支持三种银行账户类型：

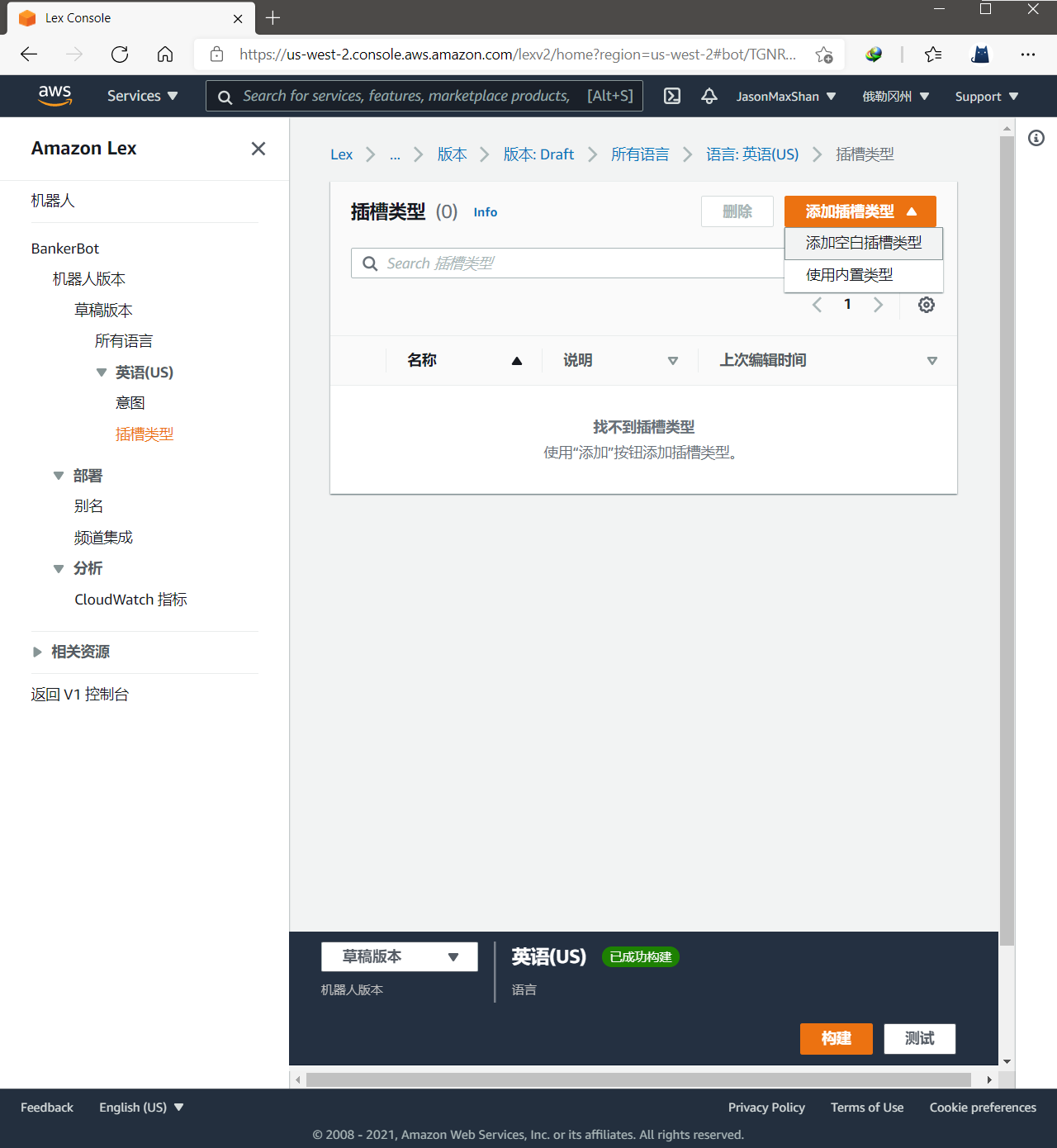
Checking

Credit

Savings

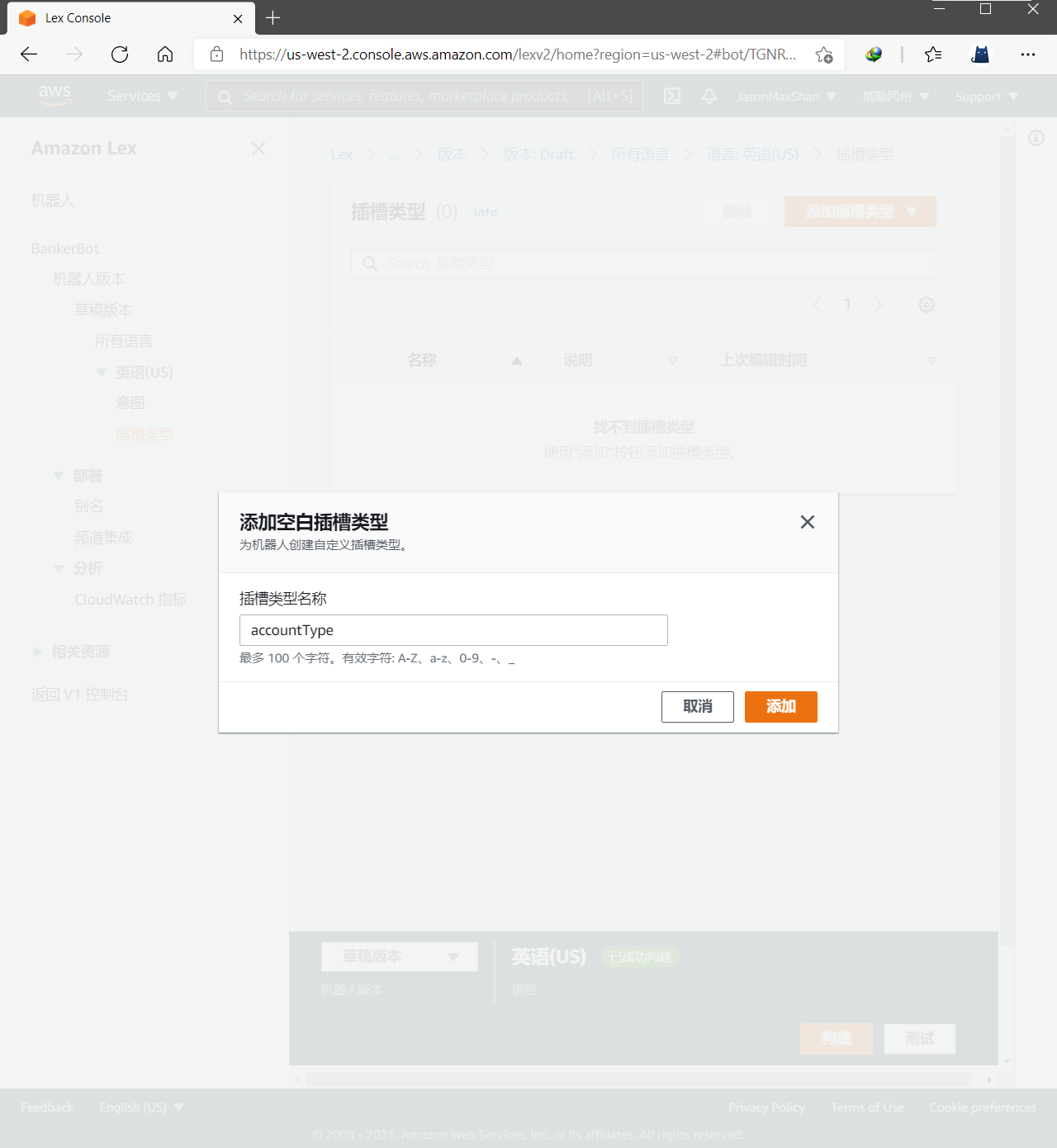
在 Lex 控制台上，前往**插槽类型**面板。

单击"**添加插槽**"类型按钮，然后选择**添加空白插槽类型**，在弹出的对话框中，请输入并单击"添加"按钮。



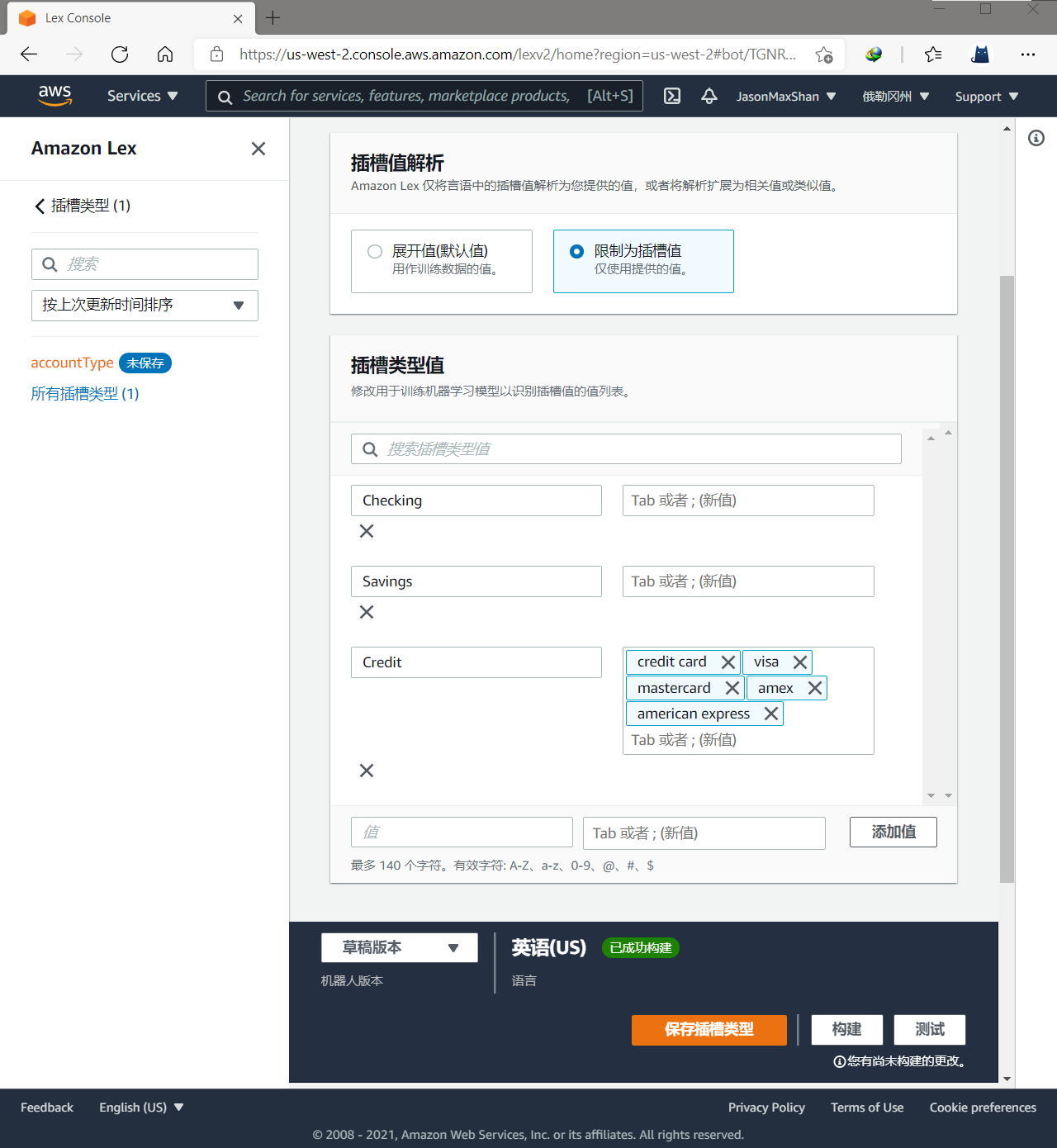
输入插槽名称：

accountType



首先要更改的是插槽值解析，将其设置为限制在插槽值。

然后将我们的三个帐户类型中的每一个添加到插槽类型值面板中 - 将每个帐户类型都放在一个"值字段"上，然后单击"添加值"按钮。



然后点击**保存**

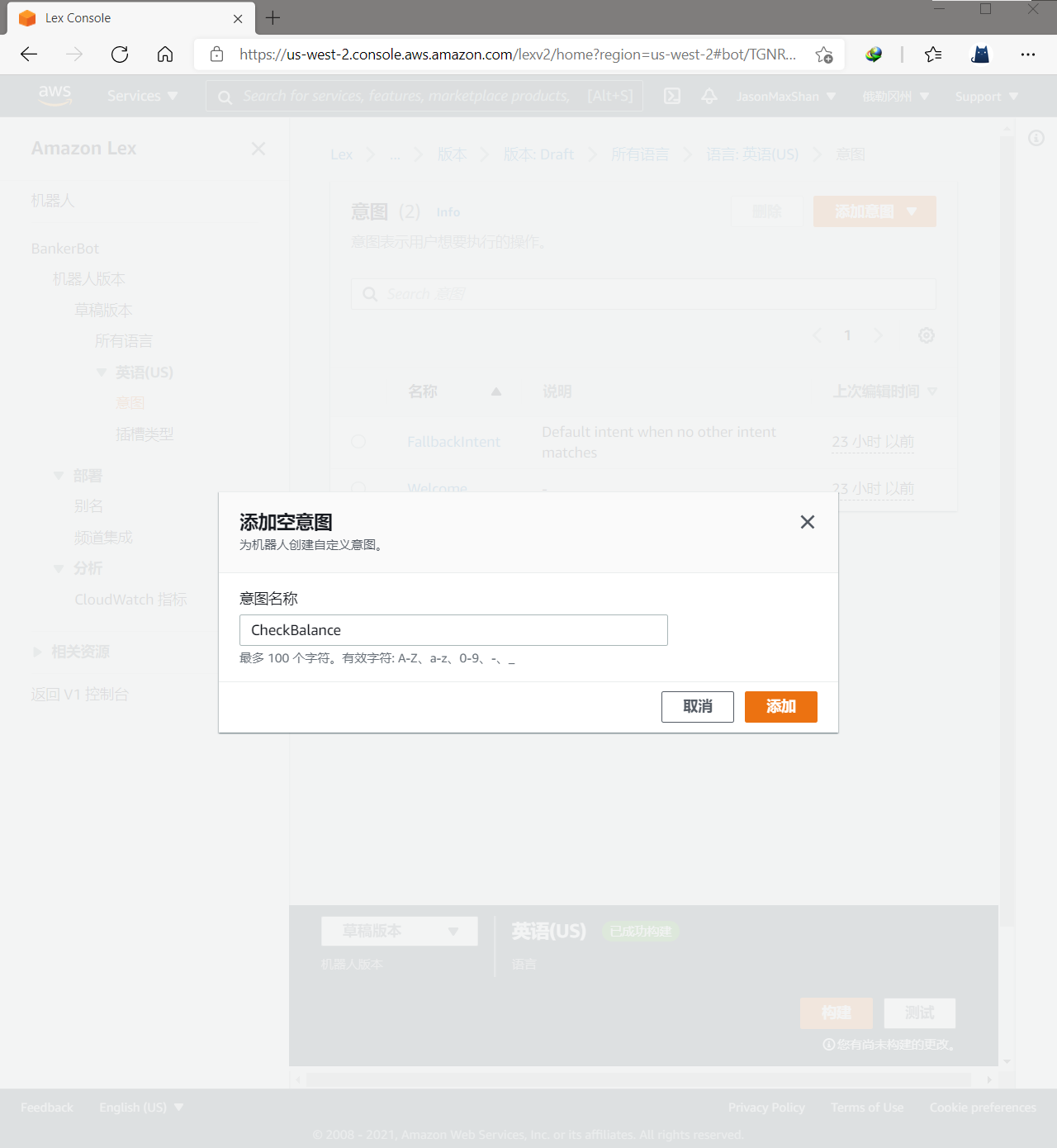
开始创建checkbalance意图

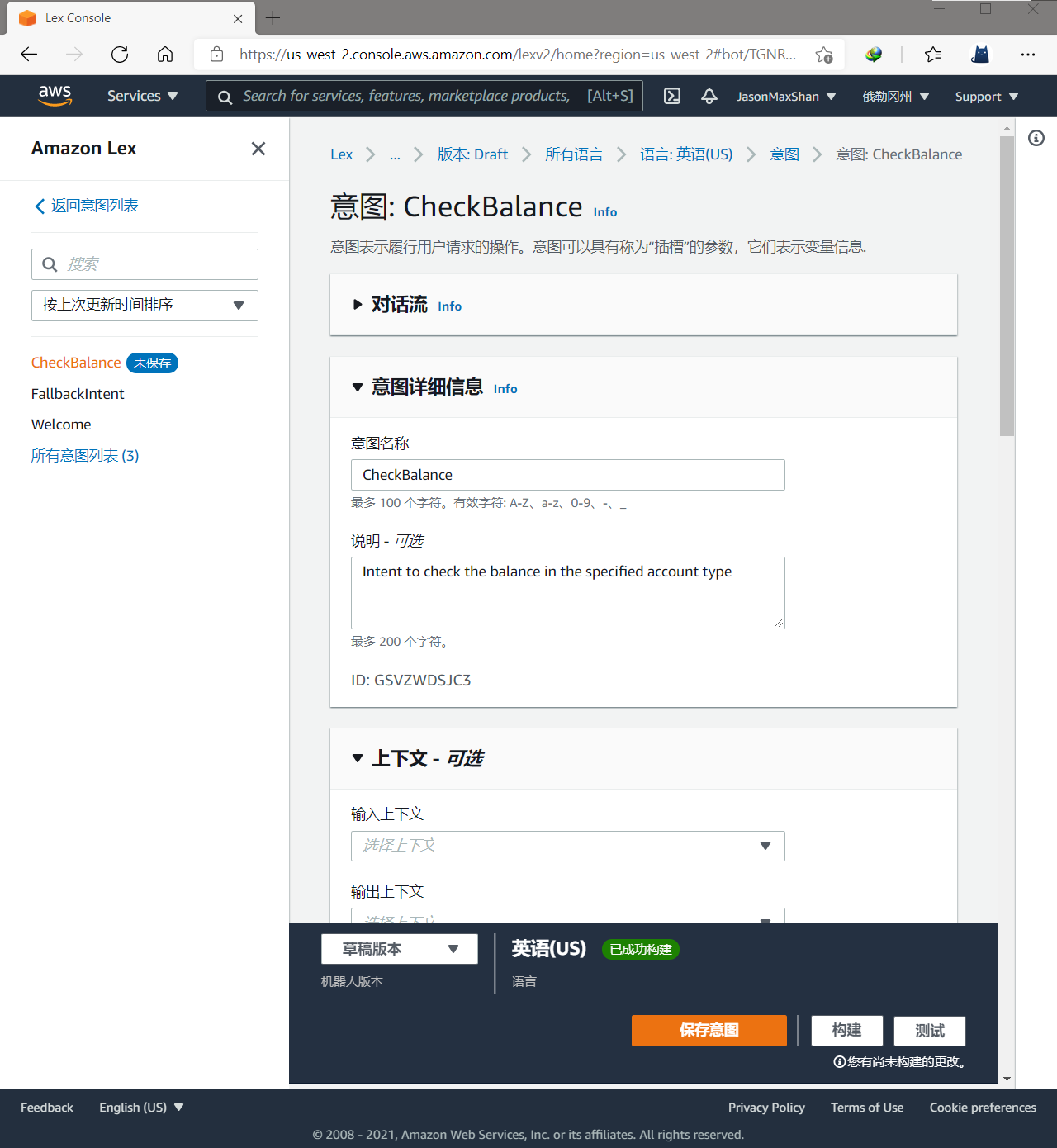
单击"意图"，然后单击"添加"意图按钮，选择"添加"空意图，对于"意图名称"请输入：

CheckBalance

然后单击"添加"按钮。在"意图"详细信息窗格中输入以下描述：

Intent to check the balance in the specified account type





来到**示例话语**。切换到**纯文本**，并粘贴在以下话语：

What’s the balance in my account ?

Check my account balance

What’s the balance in my {accountType} account ?

How much do I have in {accountType} ?

I want to check the balance

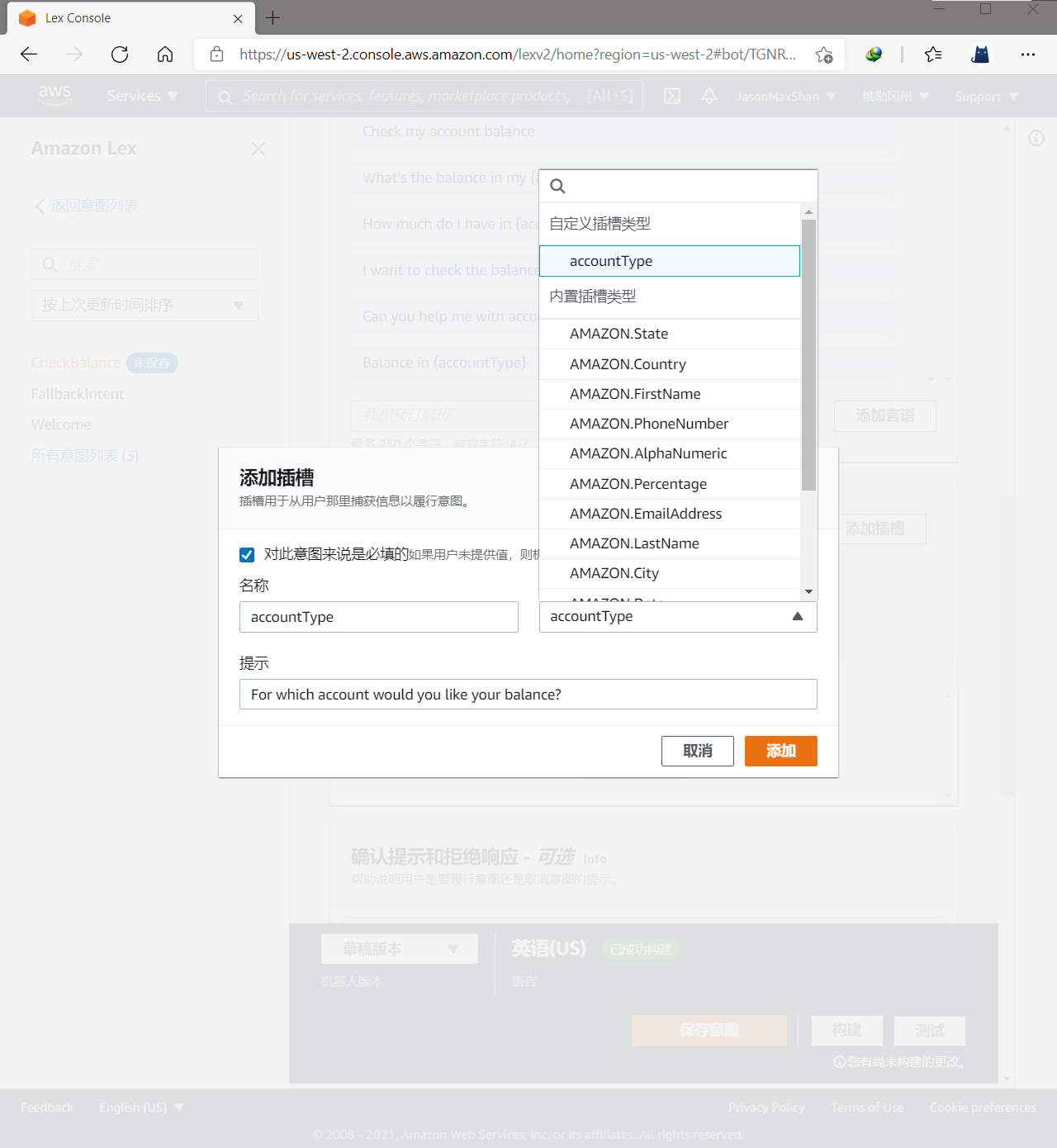
Can you help me with account balance ?

Balance in {accountType}

现在需要选择插槽的类型,单击插槽类型下拉，它将显示所有标准 Lex 插槽类型以及在此语言中定义的任何自定义插槽。只需选择我们的自定义插槽值accountType,并输入以下提示:

For which account would you like your balance?

输入信息后，单击"添加"按钮。



以同样方式添加插槽，这次使用的是内置插槽

名字：

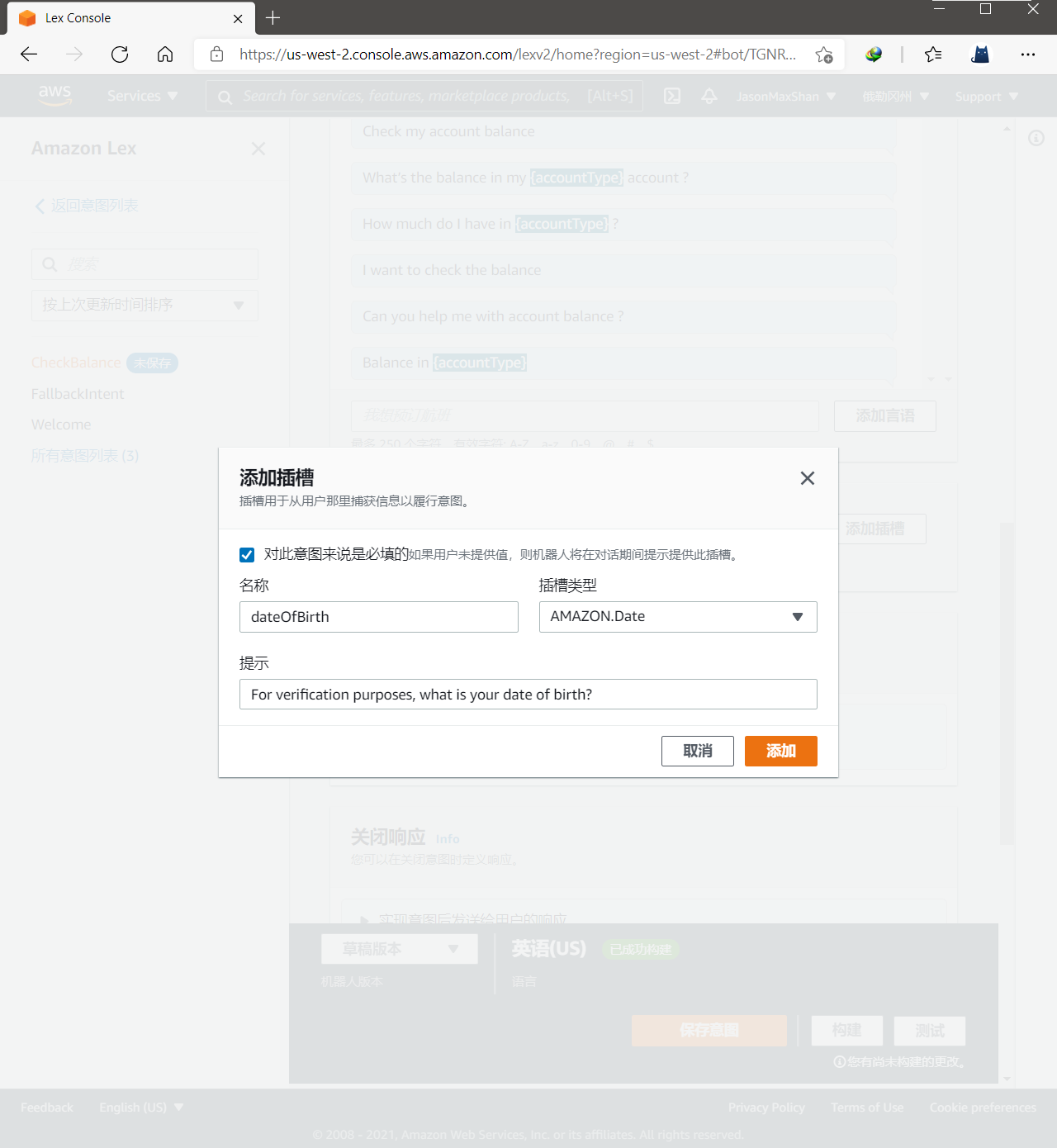
dateOfBirth

插槽类型：

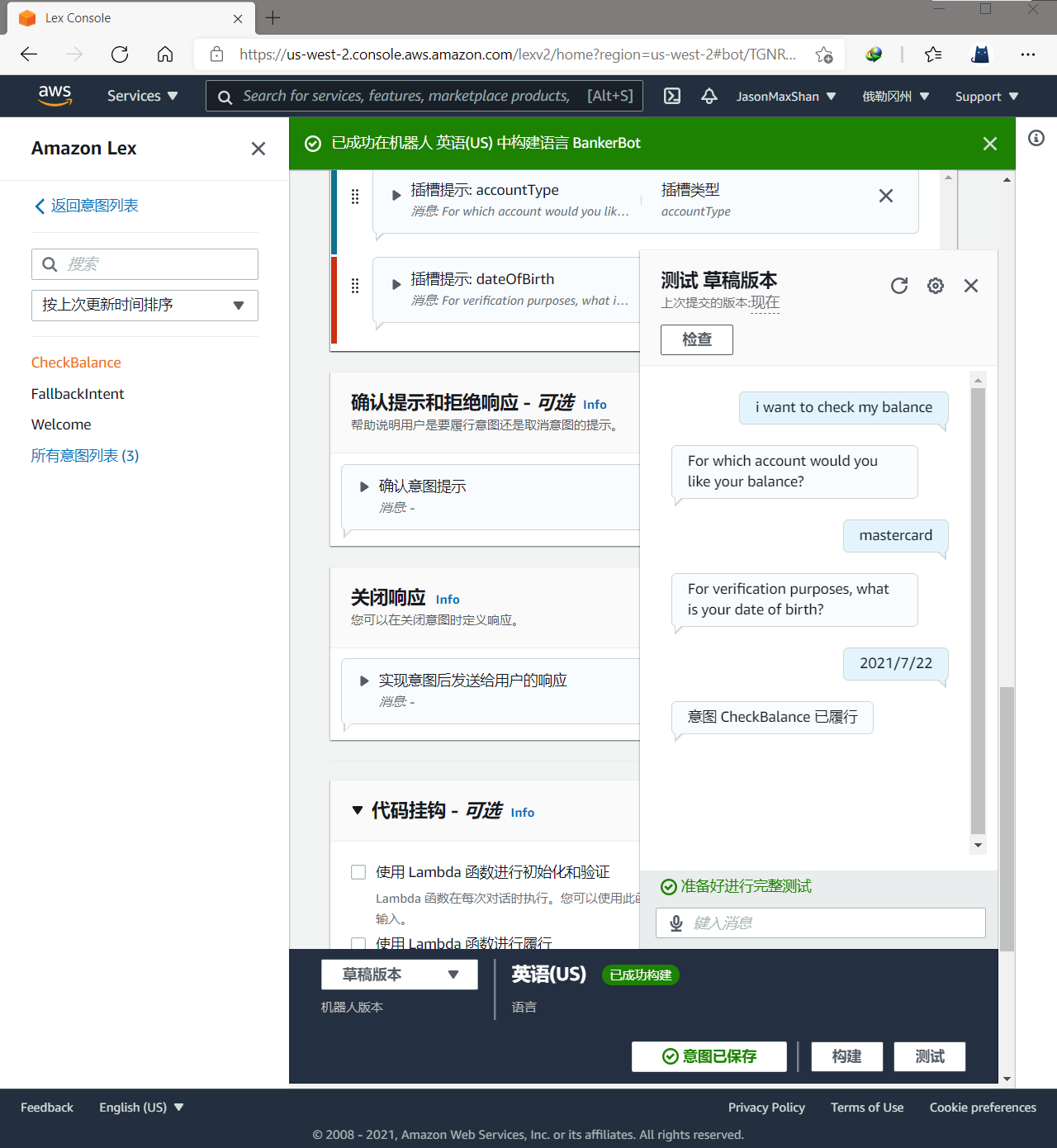
AMAZON.Date

提示：

For verification purposes, what is your date of birth?



添加完后保存意图，构建，测试



步骤三完成

步骤4：创建Bot的Lamda方法

在Lambda界面创建一个lanbda函数

方法名称：

BankingBotEnglish

运行时：

Python 3.8

函数内容：

**import json**

**import random**

**import decimal**

**def random\_num():**

**return(decimal.Decimal(random.randrange(1000, 50000))/100)**

**def get\_slots(intent\_request):**

**return intent\_request['sessionState']['intent']['slots']**

**def get\_slot(intent\_request, slotName):**

**slots = get\_slots(intent\_request)**

**if slots is not None and slotName in slots and slots[slotName] is not None:**

**return slots[slotName]['value']['interpretedValue']**

**else:**

**return None**

**def get\_session\_attributes(intent\_request):**

**sessionState = intent\_request['sessionState']**

**if 'sessionAttributes' in sessionState:**

**return sessionState['sessionAttributes']**

**return {}**

**def elicit\_intent(intent\_request, session\_attributes, message):**

**return {**

**'sessionState': {**

**'dialogAction': {**

**'type': 'ElicitIntent'**

**},**

**'sessionAttributes': session\_attributes**

**},**

**'messages': [ message ] if message != None else None,**

**'requestAttributes': intent\_request['requestAttributes'] if 'requestAttributes' in intent\_request else None**

**}**

**def close(intent\_request, session\_attributes, fulfillment\_state, message):**

**intent\_request['sessionState']['intent']['state'] = fulfillment\_state**

**return {**

**'sessionState': {**

**'sessionAttributes': session\_attributes,**

**'dialogAction': {**

**'type': 'Close'**

**},**

**'intent': intent\_request['sessionState']['intent']**

**},**

**'messages': [message],**

**'sessionId': intent\_request['sessionId'],**

**'requestAttributes': intent\_request['requestAttributes'] if 'requestAttributes' in intent\_request else None**

**}**

**def CheckBalance(intent\_request):**

**session\_attributes = get\_session\_attributes(intent\_request)**

**slots = get\_slots(intent\_request)**

**account = get\_slot(intent\_request, 'accountType')**

**#The account balance in this case is a random number**

**#Here is where you could query a system to get this information**

**balance = str(random\_num())**

**text = "Thank you. The balance on your "+account+" account is $"+balance+" dollars."**

**message = {**

**'contentType': 'PlainText',**

**'content': text**

**}**

**fulfillment\_state = "Fulfilled"**

**return close(intent\_request, session\_attributes, fulfillment\_state, message)**

**def FollowupCheckBalance(intent\_request):**

**session\_attributes = get\_session\_attributes(intent\_request)**

**slots = get\_slots(intent\_request)**

**account = get\_slot(intent\_request, 'accountType')**

**#The account balance in this case is a random number**

**#Here is where you could query a system to get this information**

**balance = str(random\_num())**

**text = "Thank you. The balance on your "+account+" account is $"+balance+" dollars."**

**message = {**

**'contentType': 'PlainText',**

**'content': text**

**}**

**fulfillment\_state = "Fulfilled"**

**return close(intent\_request, session\_attributes, fulfillment\_state, message)**

**def dispatch(intent\_request):**

**intent\_name = intent\_request['sessionState']['intent']['name']**

**response = None**

**# Dispatch to your bot's intent handlers**

**if intent\_name == 'CheckBalance':**

**return CheckBalance(intent\_request)**

**elif intent\_name == 'FollowupCheckBalance':**

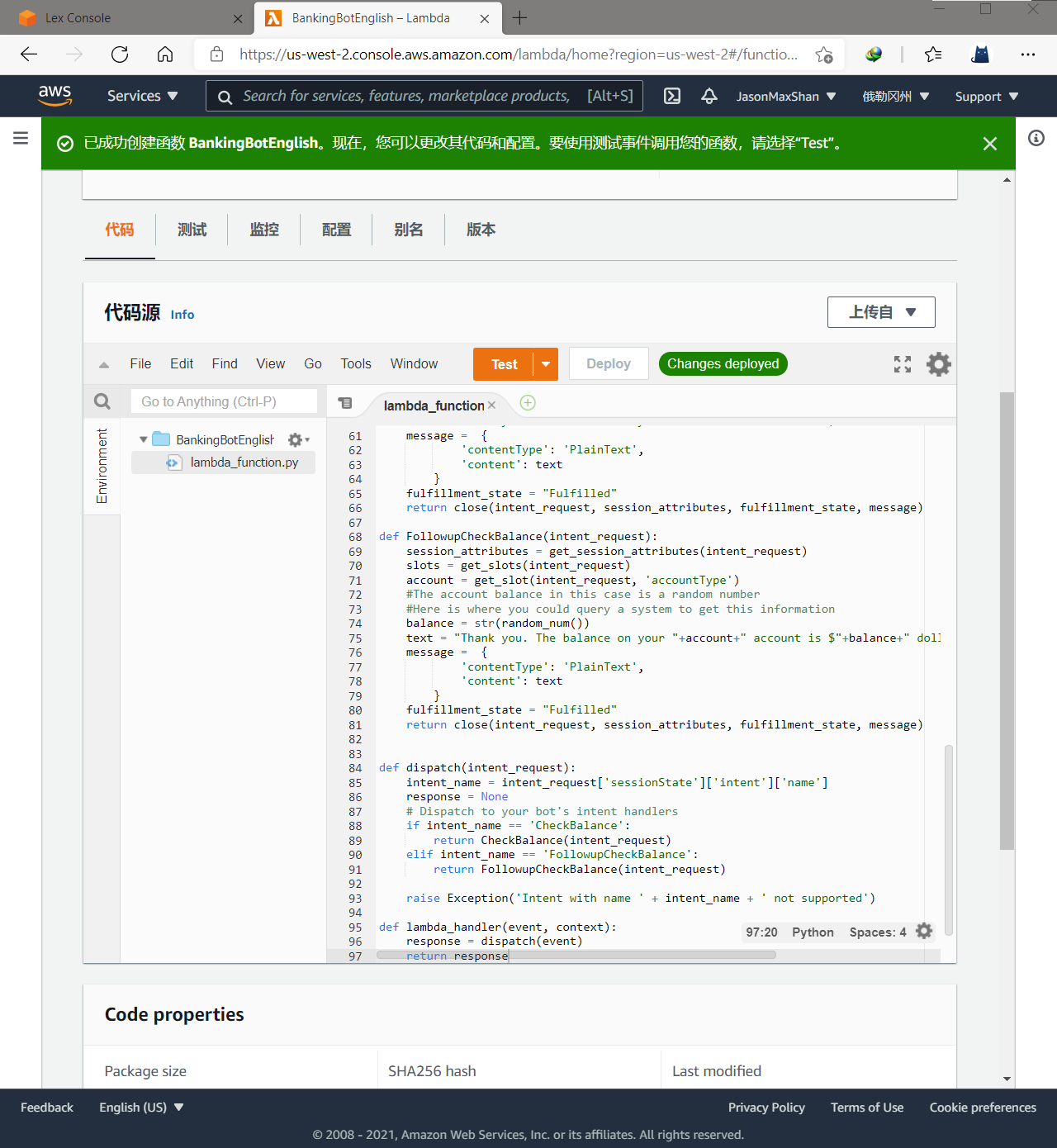
**return FollowupCheckBalance(intent\_request)**

**raise Exception('Intent with name ' + intent\_name + ' not supported')**

**def lambda\_handler(event, context):**

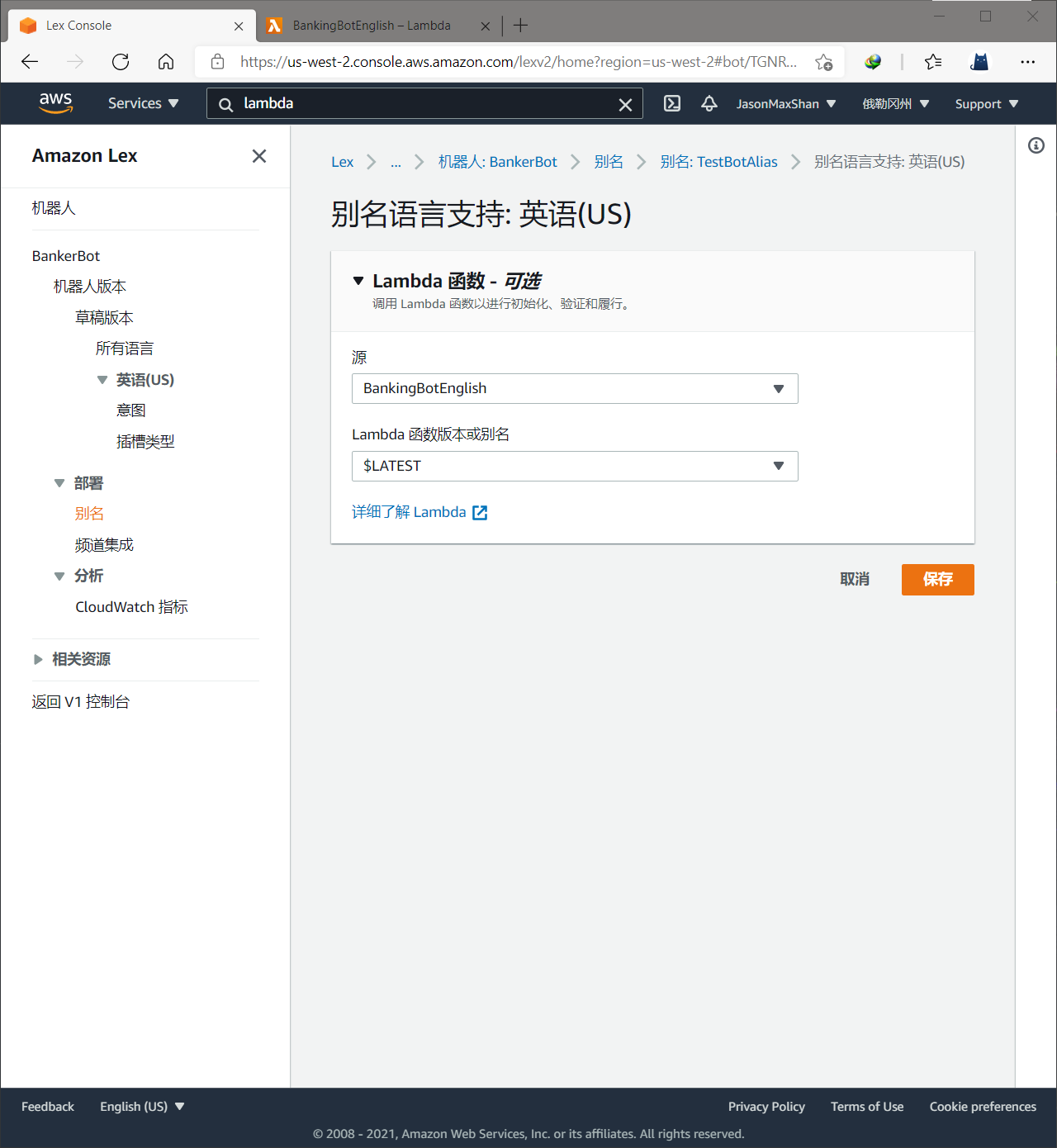
**response = dispatch(event)**

**return response**

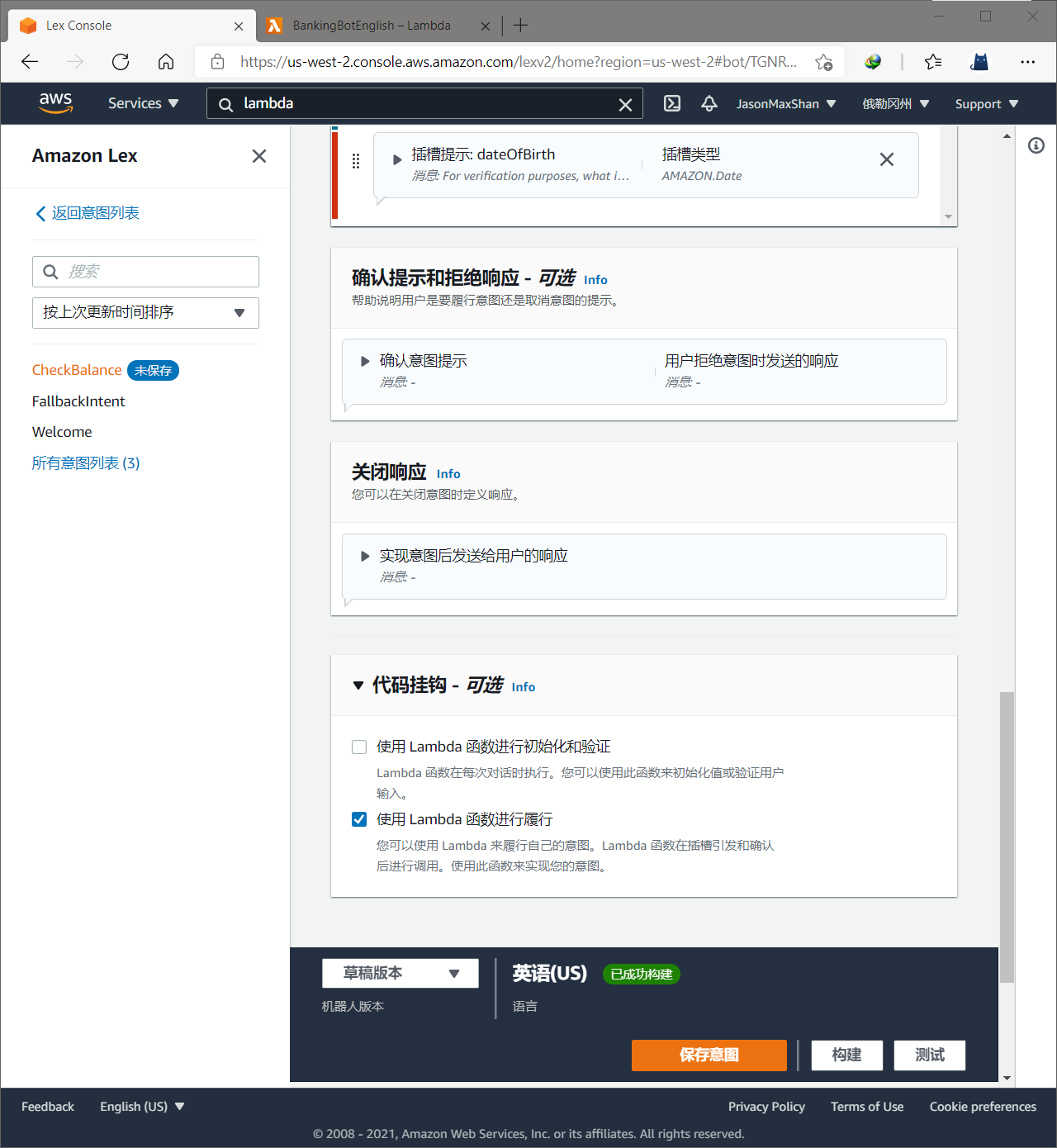


创建完后点击Deploy

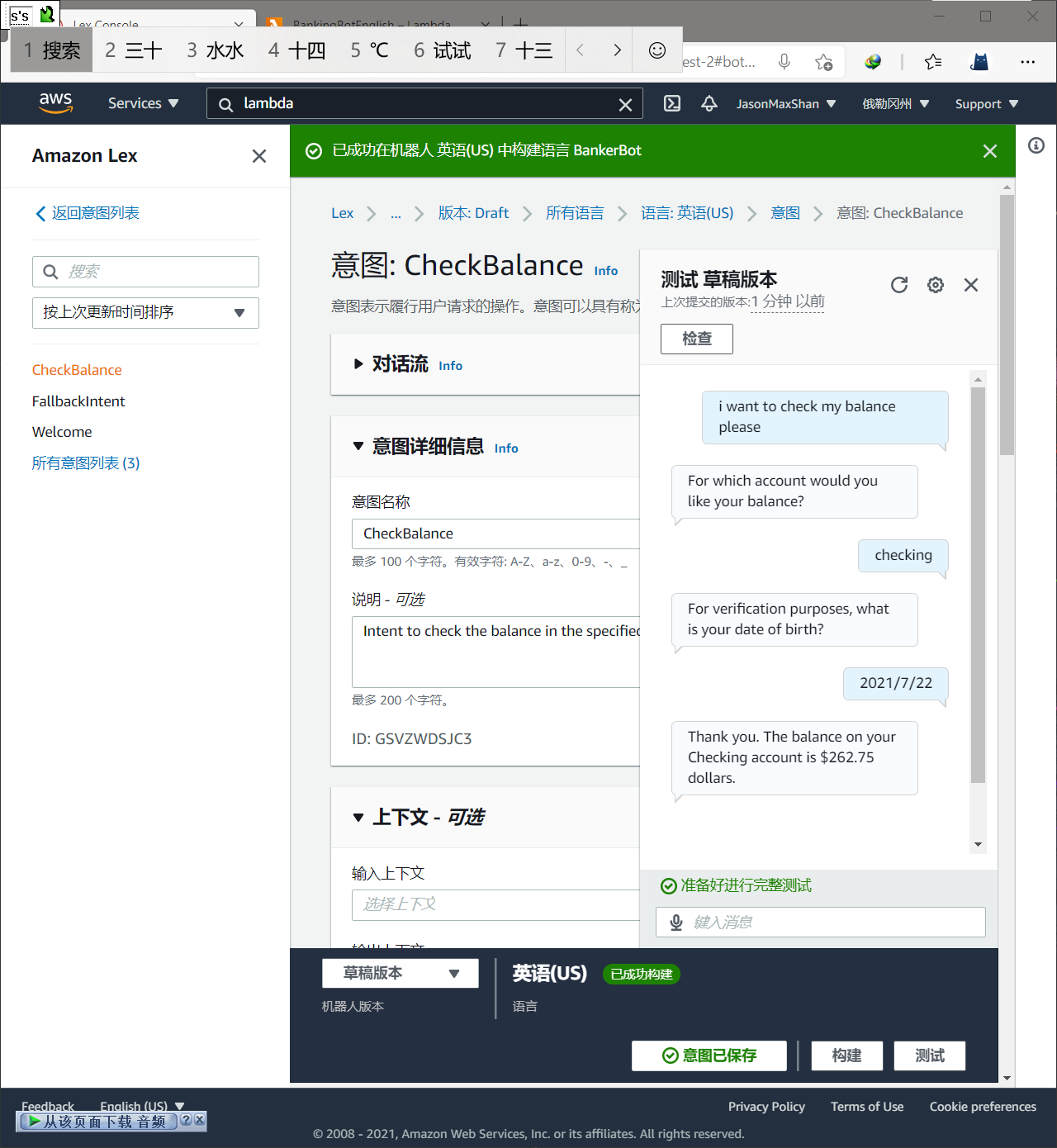
回到Lex界面，点击Lex>机器人>机器人: BankerBot>部署>别名: TestBotAlias >所有语言>语言: 英语(US)>源，选择Lambda函数



然后回到CheckBalance意图，在**代码挂钩处**选择**使用 Lambda 函数进行履行**



点击保存意图，构建，测试

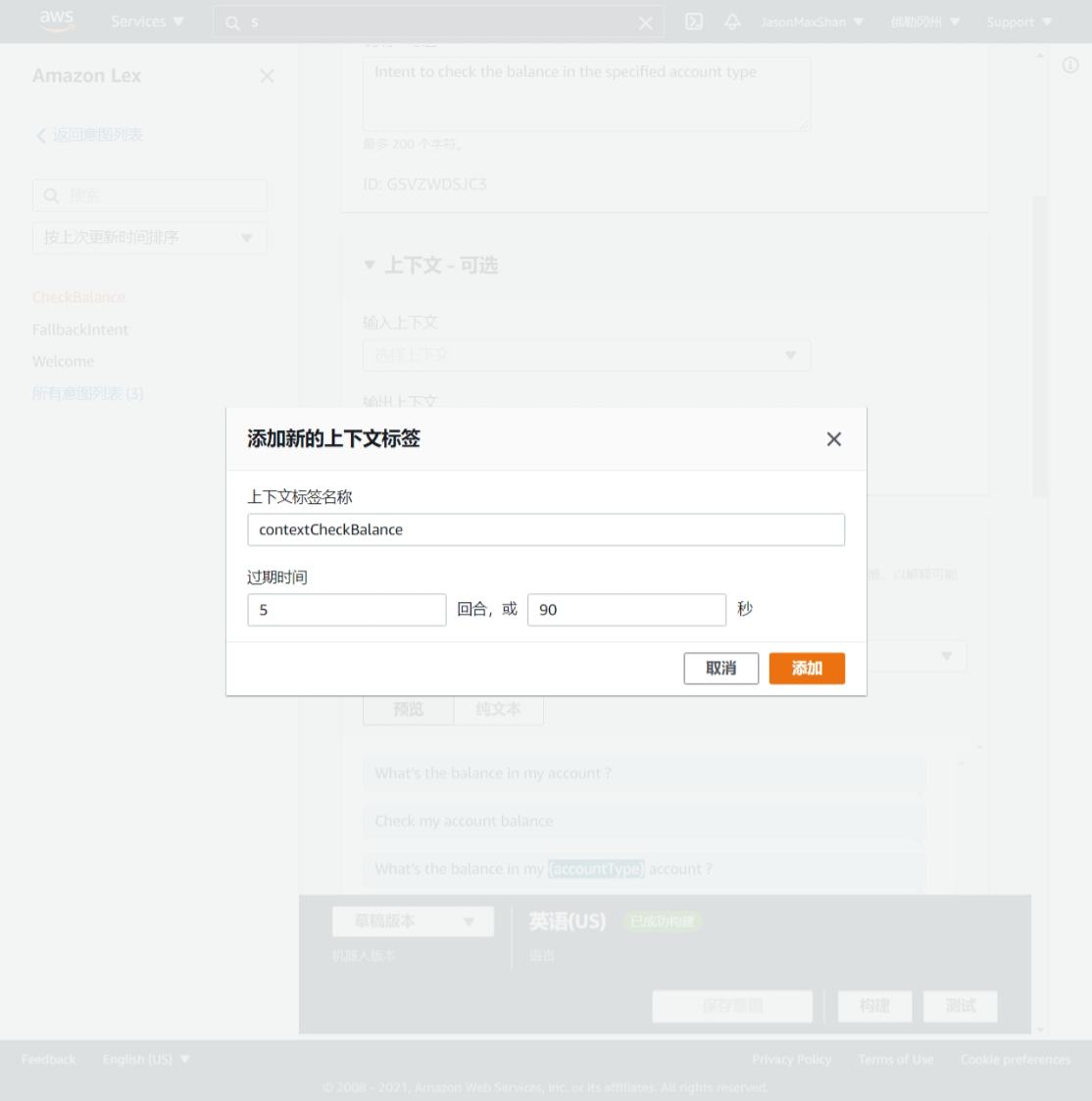


返回语句发生变化，Lambda函数可以成功被调用

步骤五：创建上下文结转

来到CheckBalance意图，找到**上下文**，选择**输出上下文**，新建上下文标签。

命名新上下文contextCheckBalance，并将过期时间分别设置为5 圈和90 秒，然后单击"添加"按钮。



然后保存，构建意图

然后我们创建一个新意图

意图名为：

FollowupCheckBalance

描述：

Intent to allow a follow-up balance check request without authentication

选择输入上下文：

contextCheckBalance

实例语言：

How about my {accountType} account

What about {accountType}

And in {accountType} ?

添加插槽：

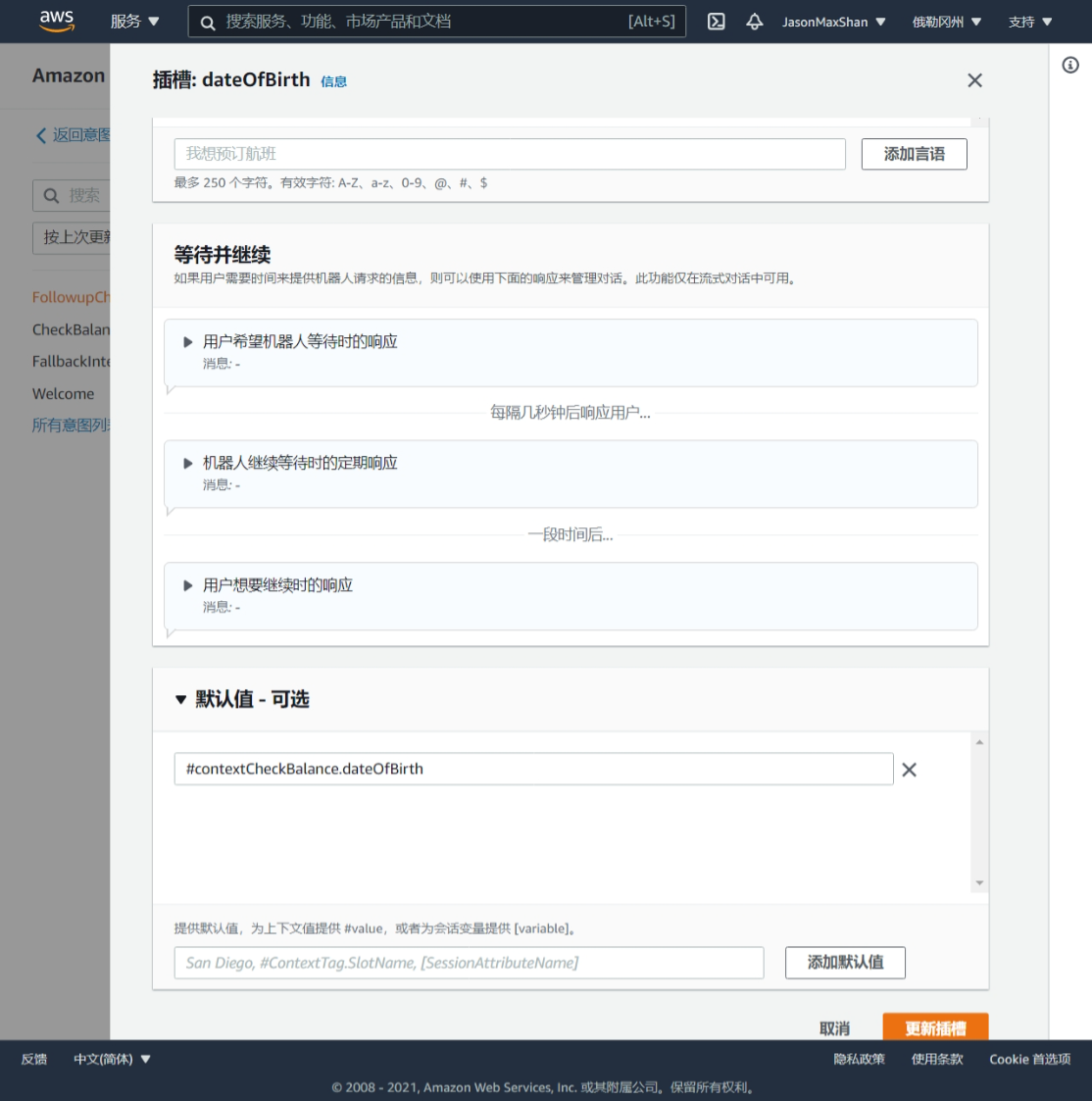
accountType

dateOfBirth

创建完后选择插槽dateOfBirth，选择**高级选项**，，拉到最后，添加**默认值**:

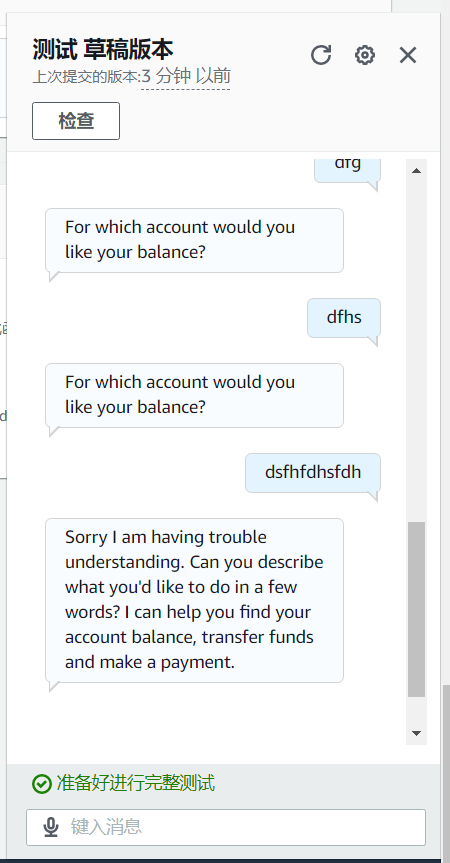
#contextCheckBalance.dateOfBirth

然后点击更新插槽



随后在**代码挂钩**处选择**使用 Lambda 函数进行履行**

保存，构建函数



此时输入五次无关信息会获得fallback意图里的回复

步骤六：重复使用插槽

创建TransferFunds意图

说明：

Help user transfer funds between bank accounts

实例语句：

I want to transfer funds

Can I make a transfer?

I want to make a transfer

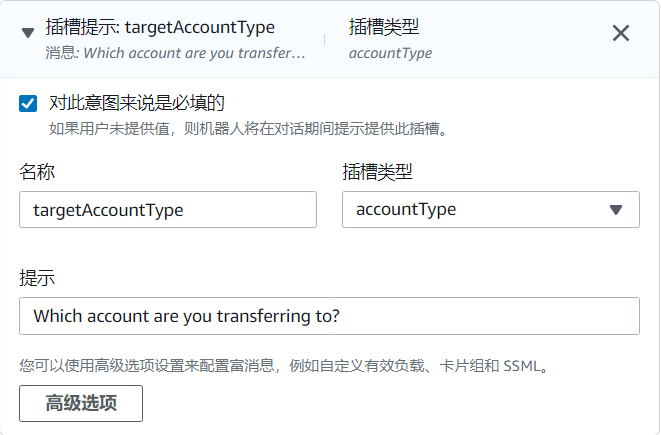
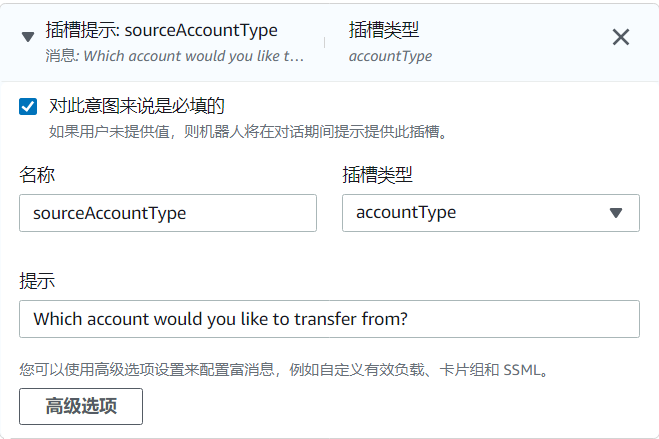
I'd like to transfer {transferAmount} from {sourceAccountType} to {targetAccountType}

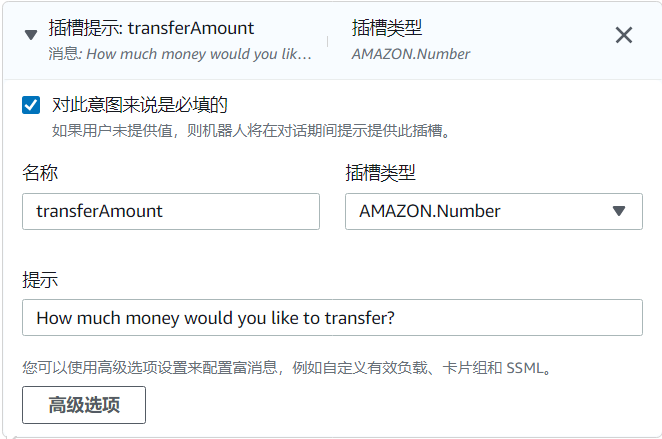
Can I transfer {transferAmount} to my {targetAccountType}

Would you be able to help me with a transfer?

Need to make a transfer

随后添加三个插槽：



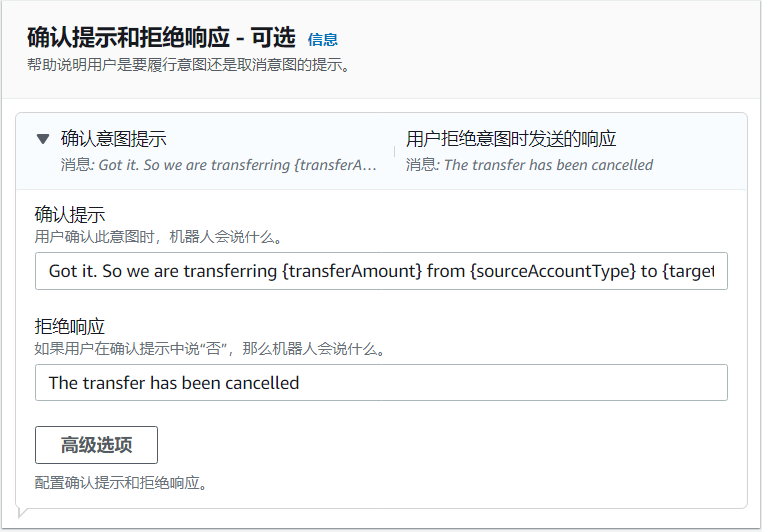


找到**确认提示和拒绝相应**，在**确认提示**中输入：

Got it. So we are transferring {transferAmount} from {sourceAccountType} to {targetAccountType}. Can I go ahead with the transfer?

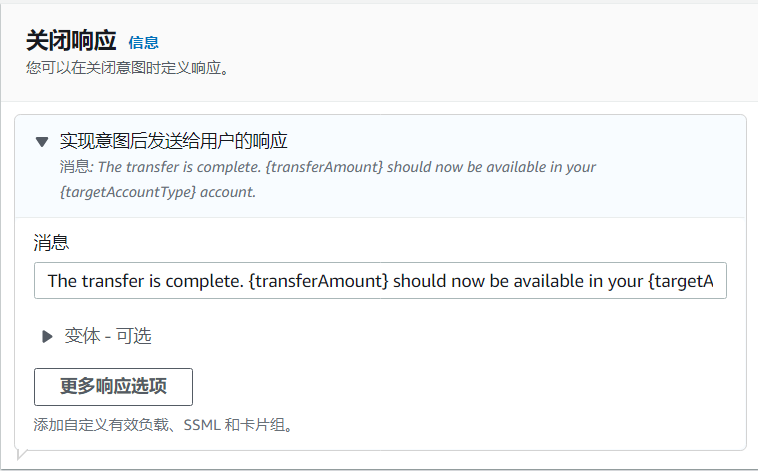
在**拒绝相应**中输入：

The transfer has been cancelled

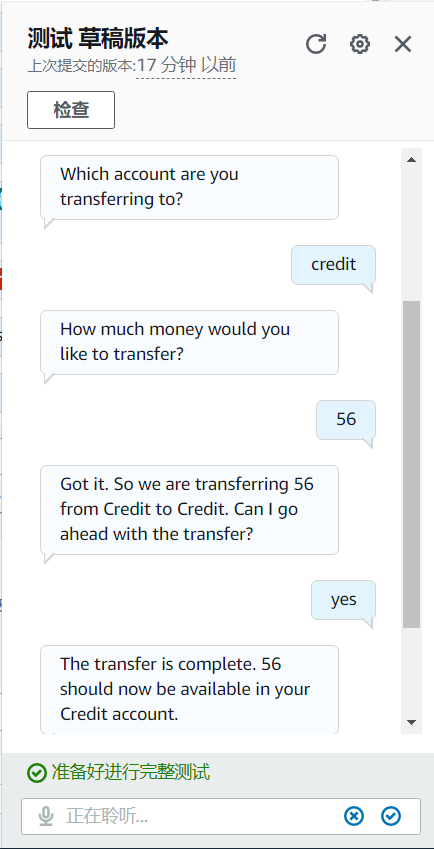


找到**关闭相应**，输入：

The transfer is complete. {transferAmount} should now be available in your {targetAccountType} account.



随后保存构建及测试



实验成功，记得删除资源