Bot 开发介绍

ROS.AI
2018.5.1

快速入门

1. 创建Bot
2. 配置Bot
3. 开发server
4. 发布

创建Bot

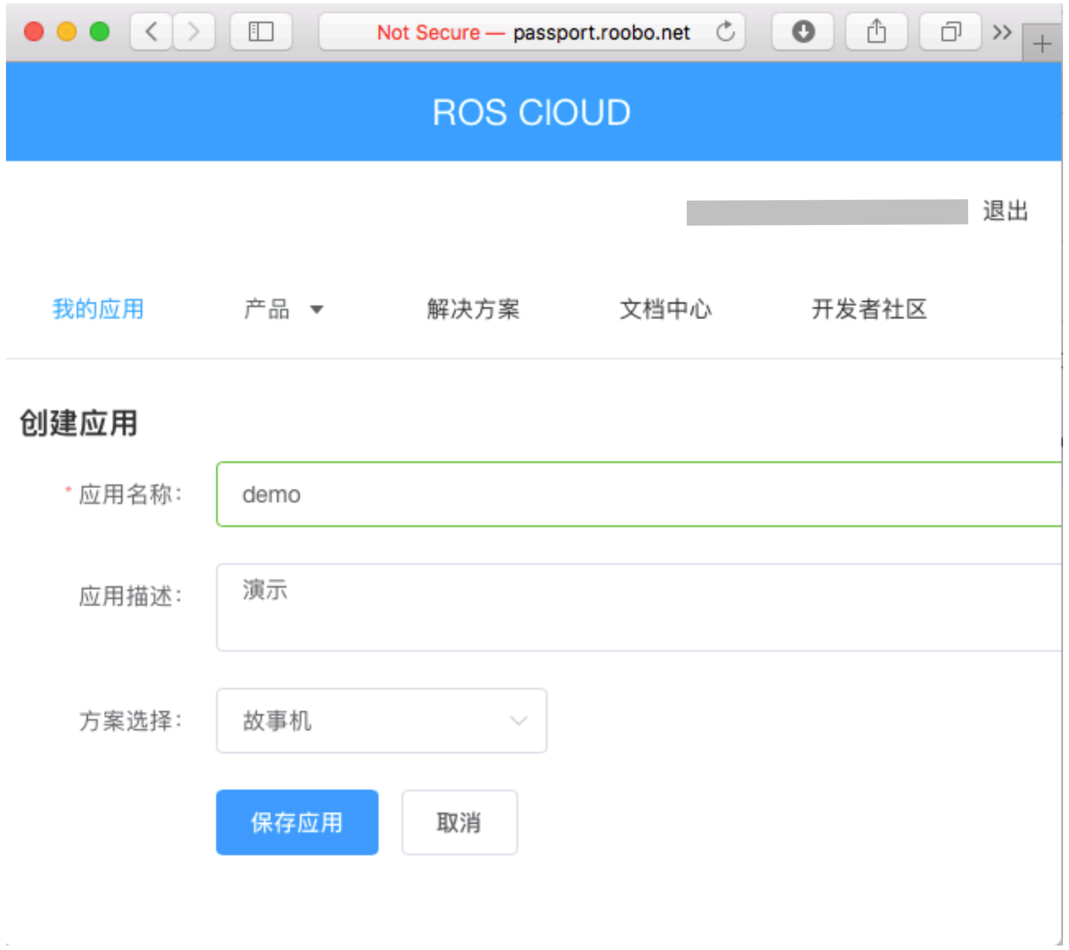
1. 创建账号

(如果已有账号，可以跳过此步骤)

url: http://passport.roobo.net/

暂不支持对外注册，试用申请请发邮件至rosai@roobo.com

2. 创建应用



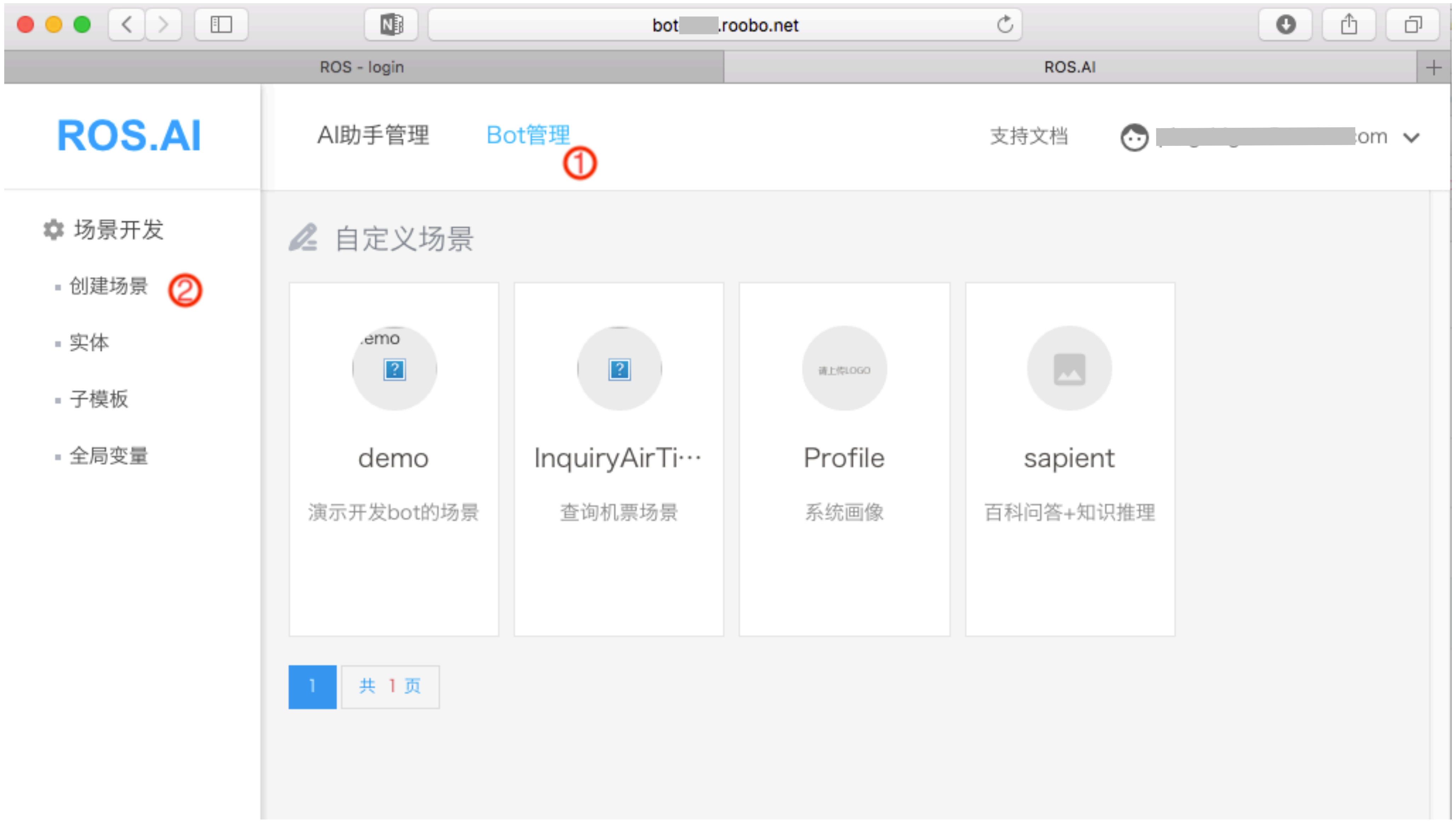
创建应用

* 应用名称:	demo
应用描述:	演示
方案选择:	故事机

创建Bot

3. 创建Bot

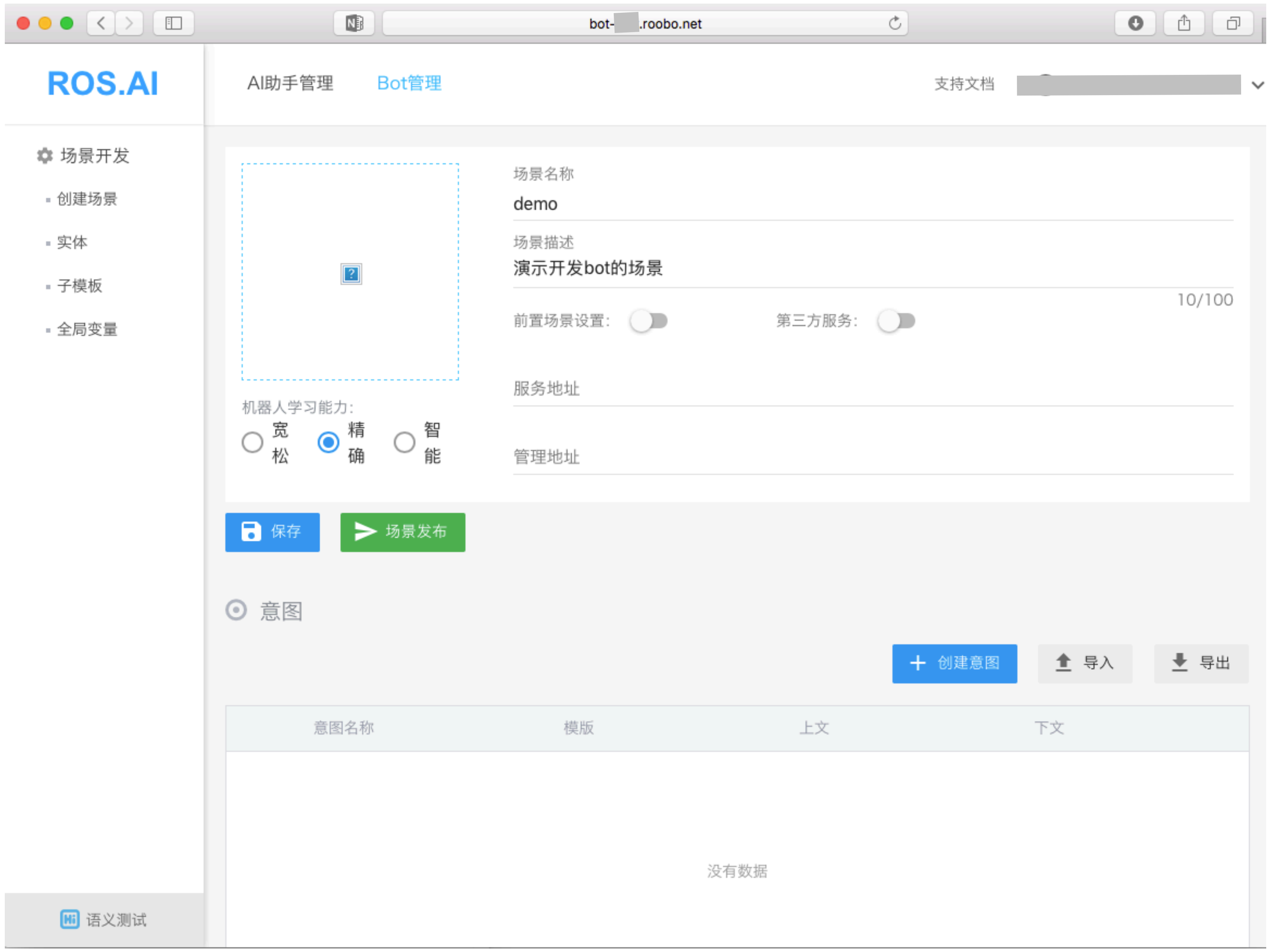
Bot管理 -> 创建场景



创建Bot

3. 创建Bot

填写场景名称，保存，场景就建好了。


bot-[redacted].roobo.net

ROS.AI

AI助手管理 Bot管理 支持文档 [redacted]

场景开发
- 创建场景
- 实体
- 子模板
- 全局变量

场景名称
demo

场景描述
演示开发bot的场景
10/100

前置场景设置： 第三方服务：

服务地址

管理地址

机器人学习能力： 宽松 [✓ 精确] 智能

保存 场景发布

意图

创建意图 导入 导出

意图名称	模版	上文	下文

没有数据

语义测试

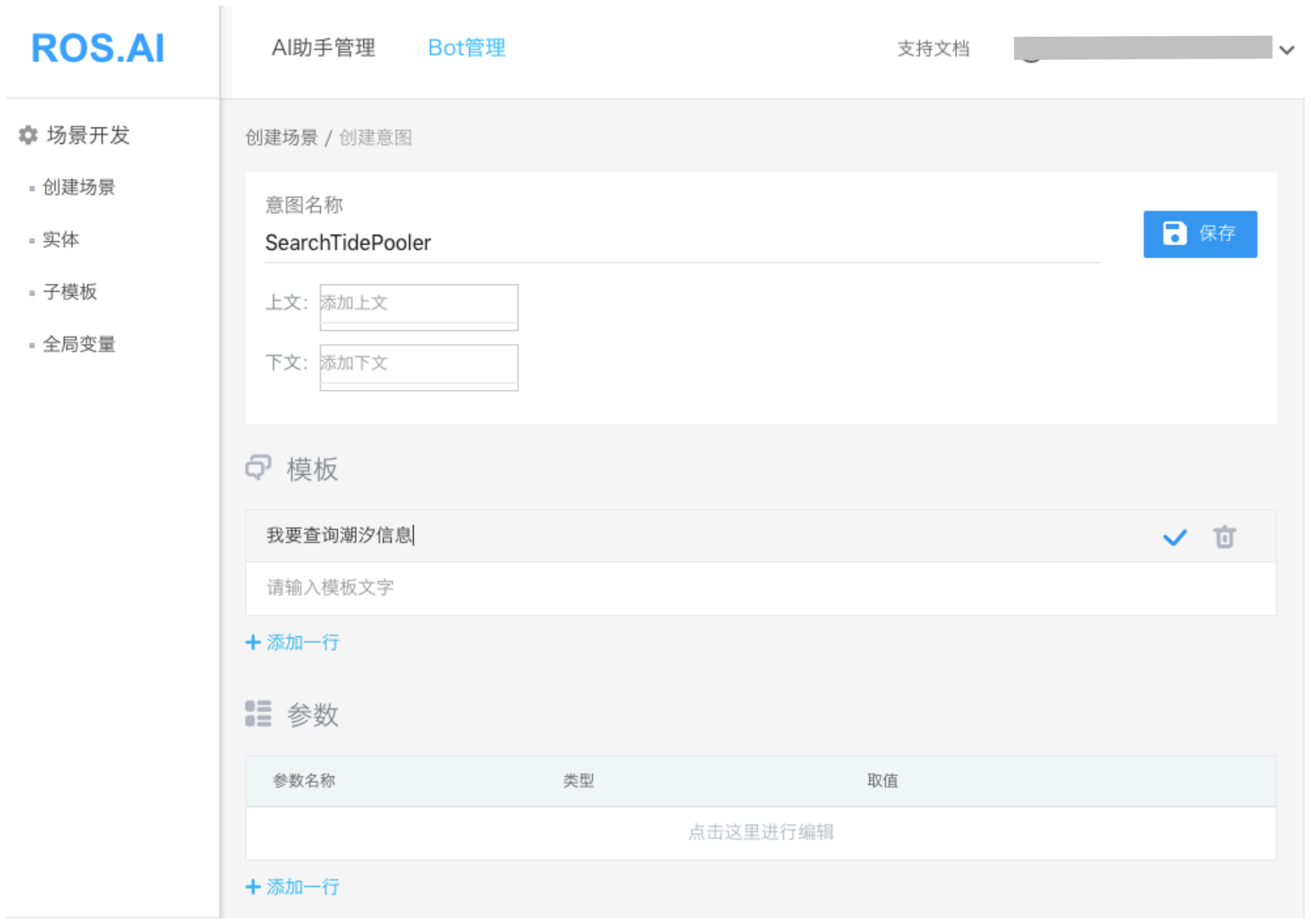
配置Bot

1. 创建意图

点击创建意图，进入创建意图界面。填写意图名，如“SearchTidePooler”，保存。意图就创建好了。

2. 配置意图模版

在模版配置区域，填入该意图需要匹配的句式。如“我要查询潮汐信息”，点击右侧“V”保存模版。



ROS.AI　AI助手管理　Bot管理　支持文档　[redacted]

场景开发
- 创建场景
- 实体
- 子模板
- 全局变量

创建场景 / 创建意图

意图名称
SearchTidePooler
保存

上文: 添加上文
下文: 添加下文

模板

我要查询潮汐信息 ✓
请输入模板文字
+ 添加一行

参数

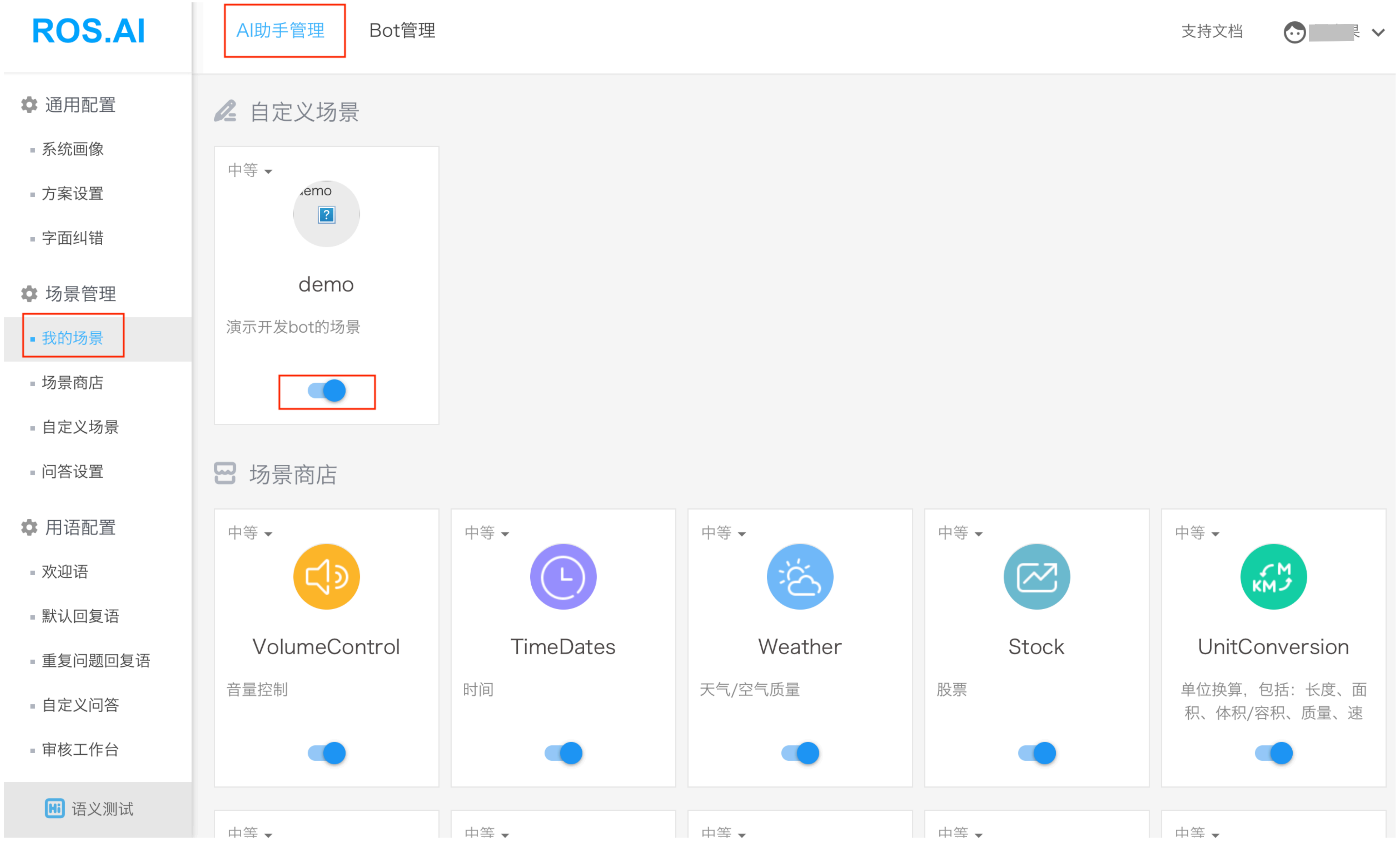
参数名称	类型	取值
点击这里进行编辑		

+ 添加一行

配置Bot

3. 测试意图模版

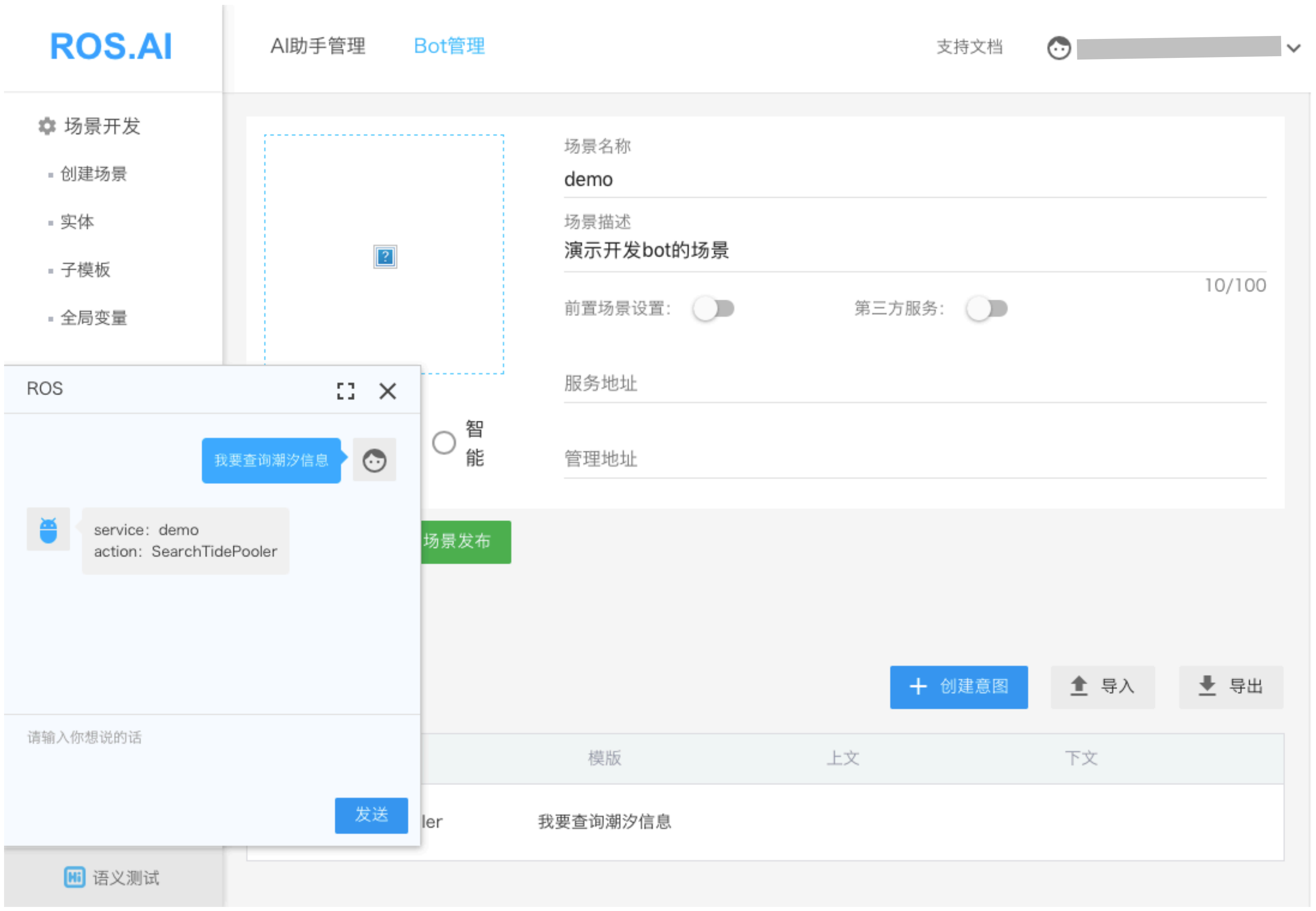
- 在测试语义前需要使能场景(技能)
- 操作：点击上方的“AI助手管理”->点击应用->点击左侧“我的场景”->在右侧页面找到场景并使能



配置Bot

3. 测试意图模版

- 点击左下方的“语义测试”，输入刚才填写的模版“我要查询潮汐信息”，点“发送”，rosai会返回测试结果。
- 如果配置正确的话，会看到匹配上的场景sevice和意图action，如下图。



场景名称
demo
场景描述
演示开发bot的场景
10/100
前置场景设置：
第三方服务：
服务地址
管理地址

我要查询潮汐信息

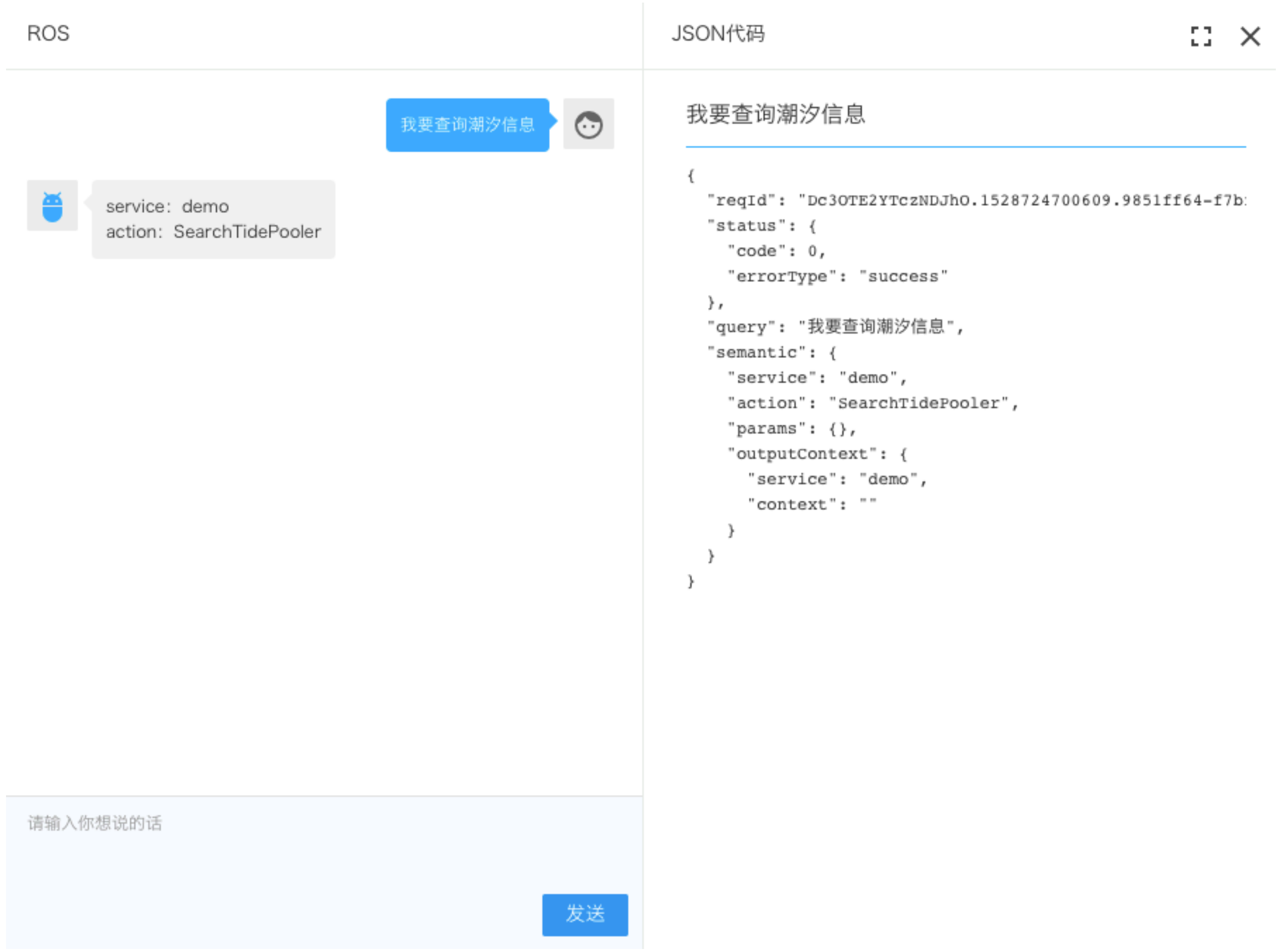
service: demo
action: SearchTidePooler

ler 我要查询潮汐信息

配置Bot

3. 测试意图模版

- 点击最大化按钮，能看到详细的语义解析信息。


ROS

我要查询潮汐信息

service：demo
action：SearchTidePooler

请输入你想说的话

发送

JSON代码

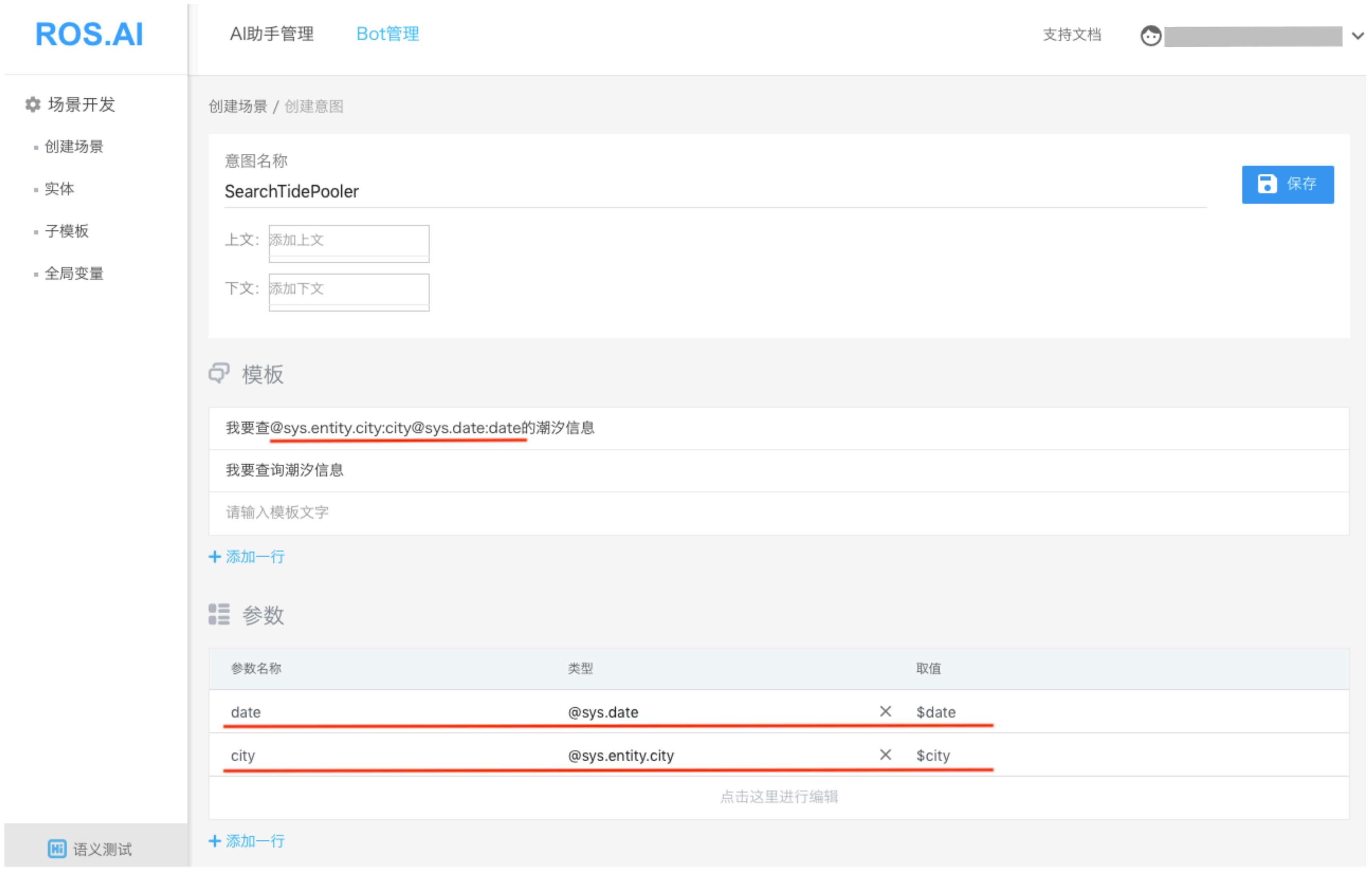
我要查询潮汐信息

```
{
  "reqId": "Dc3OTE2YTczNDJhO.1528724700609.9851ff64-f7b:
  "status": {
    "code": 0,
    "errorType": "success"
  },
  "query": "我要查询潮汐信息",
  "semantic": {
    "service": "demo",
    "action": "SearchTidePooler",
    "params": {},
    "outputContext": {
      "service": "demo",
      "context": ""
    }
  }
}
```

配置Bot

4. 添加槽位

- 在创建意图页面(Bot管理/选择场景/选择意图)，按照如下格式新添加一个模版句式和参数(槽位)。



意图名称

SearchTidePooler

上文: 添加上文

下文: 添加下文

模板

我要查@sys.entity.city:city@sys.date:date的潮汐信息

我要查询潮汐信息

请输入模板文字

+ 添加一行

参数

参数名称	类型	取值
date	@sys.date	$date
city	@sys.entity.city	$city

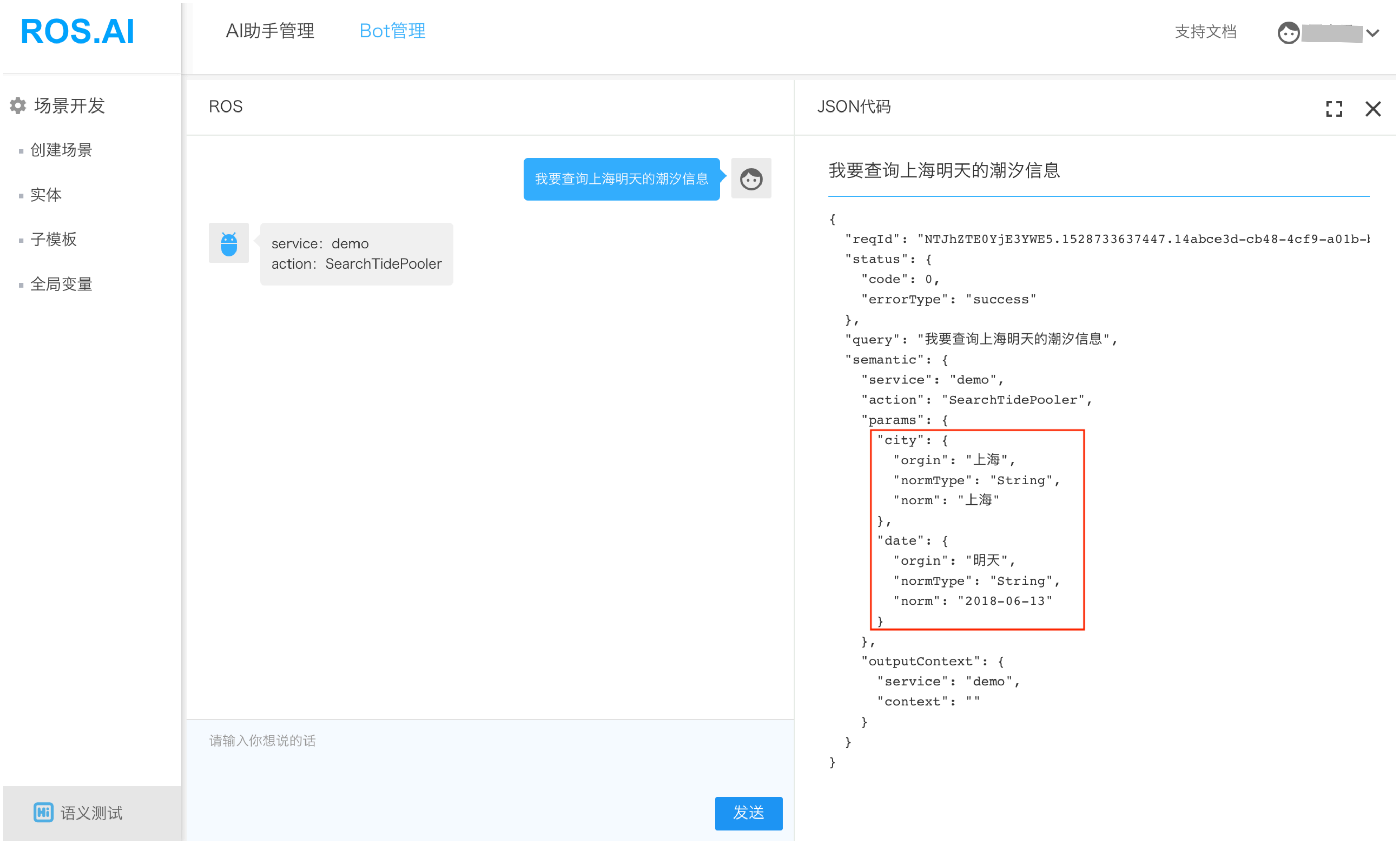
点击这里进行编辑

+ 添加一行

配置Bot

5. 测试带槽位的意图模版

- 输入“我要查询上海明天的潮汐信息”，查看详细的语义解析信息，如下图。



我要查询上海明天的潮汐信息

service：demo
action：SearchTidePooler

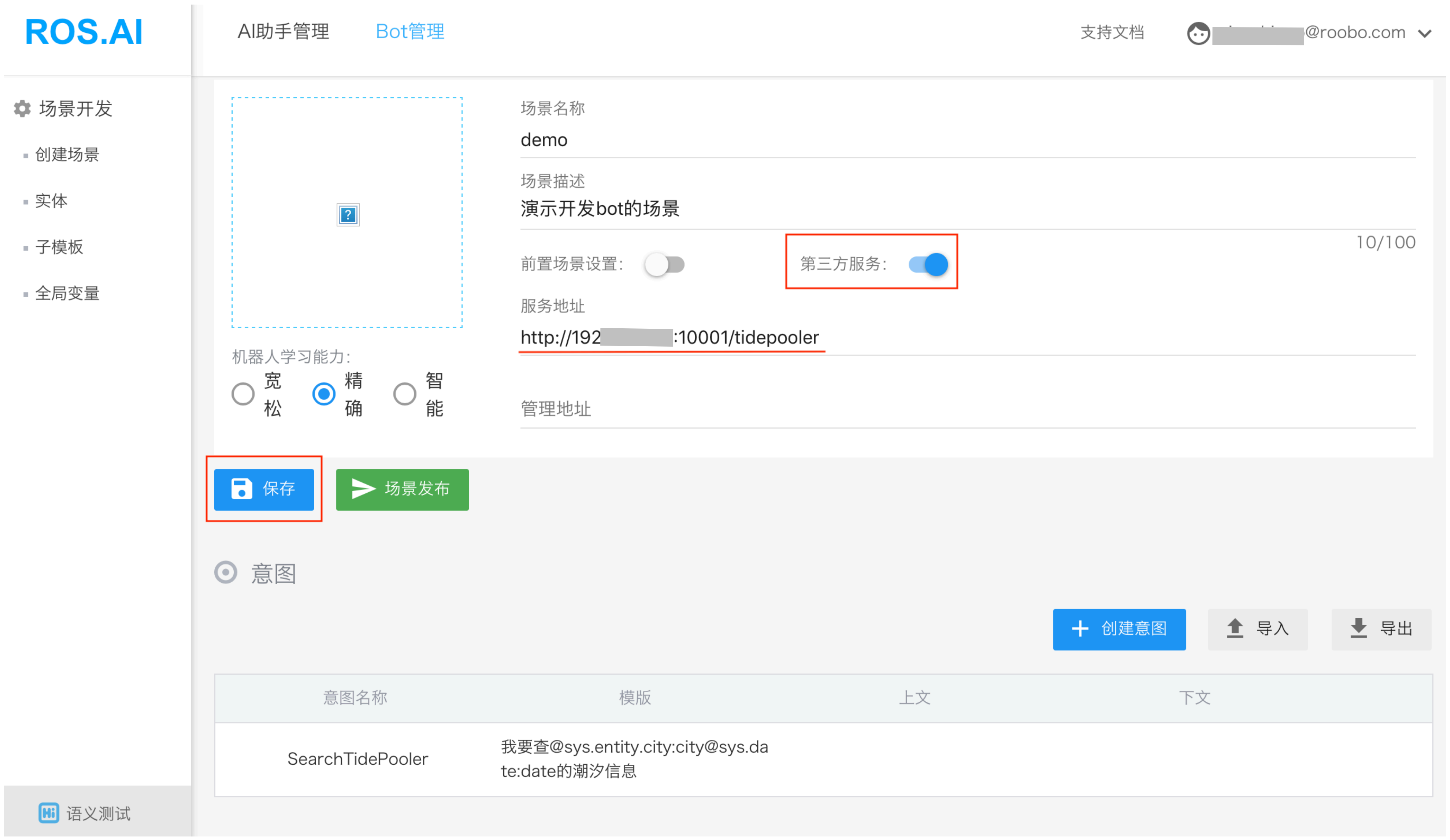
我要查询上海明天的潮汐信息

```
{
  "reqId": "NTJhZTE0YjE3YWE5.1528733637447.14abce3d-cb48-4cf9-a01b-
  "status": {
    "code": 0,
    "errorType": "success"
  },
  "query": "我要查询上海明天的潮汐信息",
  "semantic": {
    "service": "demo",
    "action": "SearchTidePooler",
    "params": {
      "city": {
        "orgin": "上海",
        "normType": "String",
        "norm": "上海"
      },
      "date": {
        "orgin": "明天",
        "normType": "String",
        "norm": "2018-06-13"
      }
    },
    "outputContext": {
      "service": "demo",
      "context": ""
    }
  }
}
```

开发server

1. 配置服务

- 在场景主界面(Bot管理/选中场景)，使能第三方服务，在服务地址栏输入服务api url，如“http://192.168.1.41:10001/tidepooler”。



场景名称
demo

场景描述
演示开发bot的场景

10/100

前置场景设置: ○ 第三方服务: ●

服务地址
http://192[redacted]:10001/tidepooler

管理地址

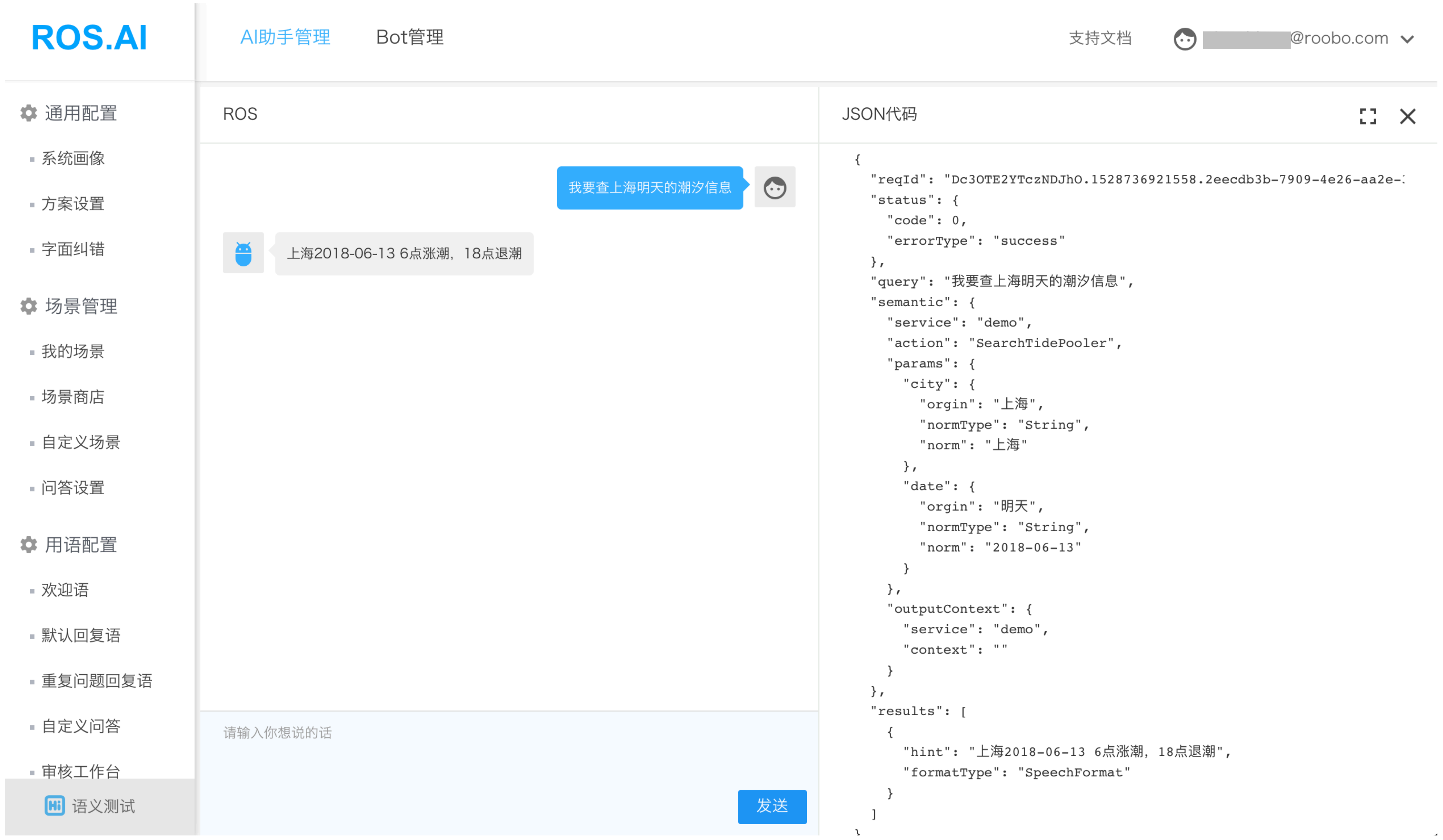
机器人学习能力: 宽松 [✓ 精确] 智能

意图名称	模版	上文	下文
SearchTidePooler	我要查@sys.entity.city:city@sys.date:date的潮汐信息		

开发server

2. 测试服务

- 在应用界面下(AI助手/使能该场景的应用界面)，打开左下方语义测试窗口，输入语句“我要查上海明天的潮汐信息”



我要查上海明天的潮汐信息

上海2018-06-13 6点涨潮，18点退潮

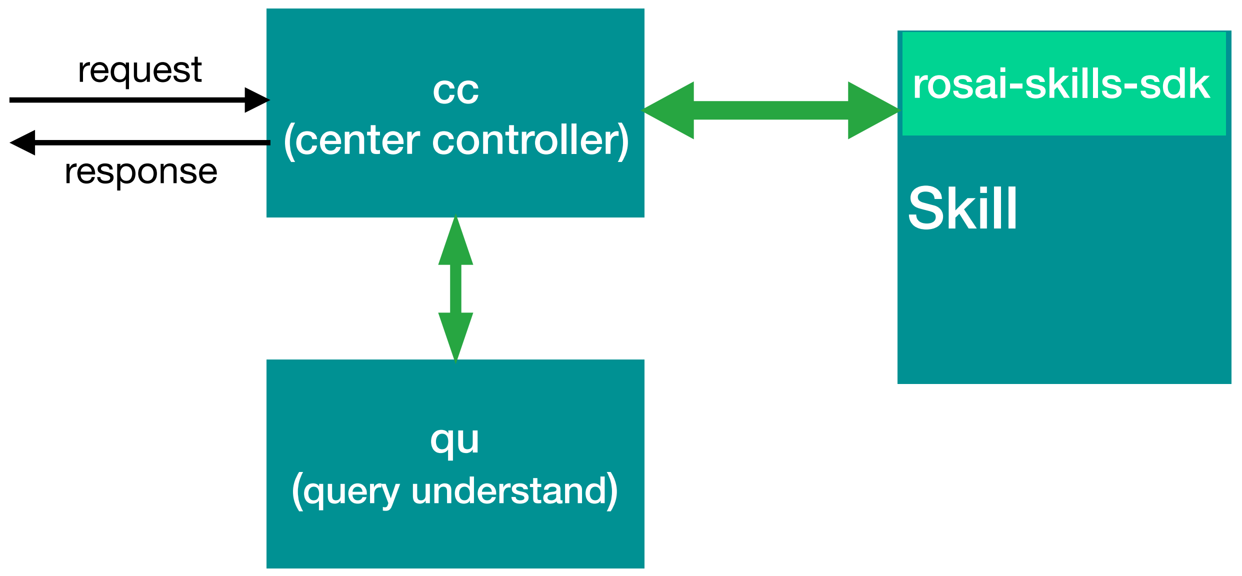
```
{
  "reqId": "Dc3OTE2YTczNDJhO.1528736921558.2eecdb3b-7909-4e26-aa2e-
  "status": {
    "code": 0,
    "errorType": "success"
  },
  "query": "我要查上海明天的潮汐信息",
  "semantic": {
    "service": "demo",
    "action": "SearchTidePooler",
    "params": {
      "city": {
        "orgin": "上海",
        "normType": "String",
        "norm": "上海"
      },
      "date": {
        "orgin": "明天",
        "normType": "String",
        "norm": "2018-06-13"
      }
    },
    "outputContext": {
      "service": "demo",
      "context": ""
    }
  },
  "results": [
    {
      "hint": "上海2018-06-13 6点涨潮，18点退潮",
      "formatType": "SpeechFormat"
    }
  ]
}
```

开发server

3. 开发服务

- 根据交互协议自定义开发(接口协议可参考rosai-skills-v2.md)
- 基于rosai-skills-sdk开发(目前支持go)

rosai-skills-sdk实现了一套根据rosai消息传输协议的自定义技能框架，
该框架让自定义技能开发者无需关注协议细节，消息解析和响应等繁琐的实现，
只需关注业务逻辑的实现。



开发server

3. 开发服务

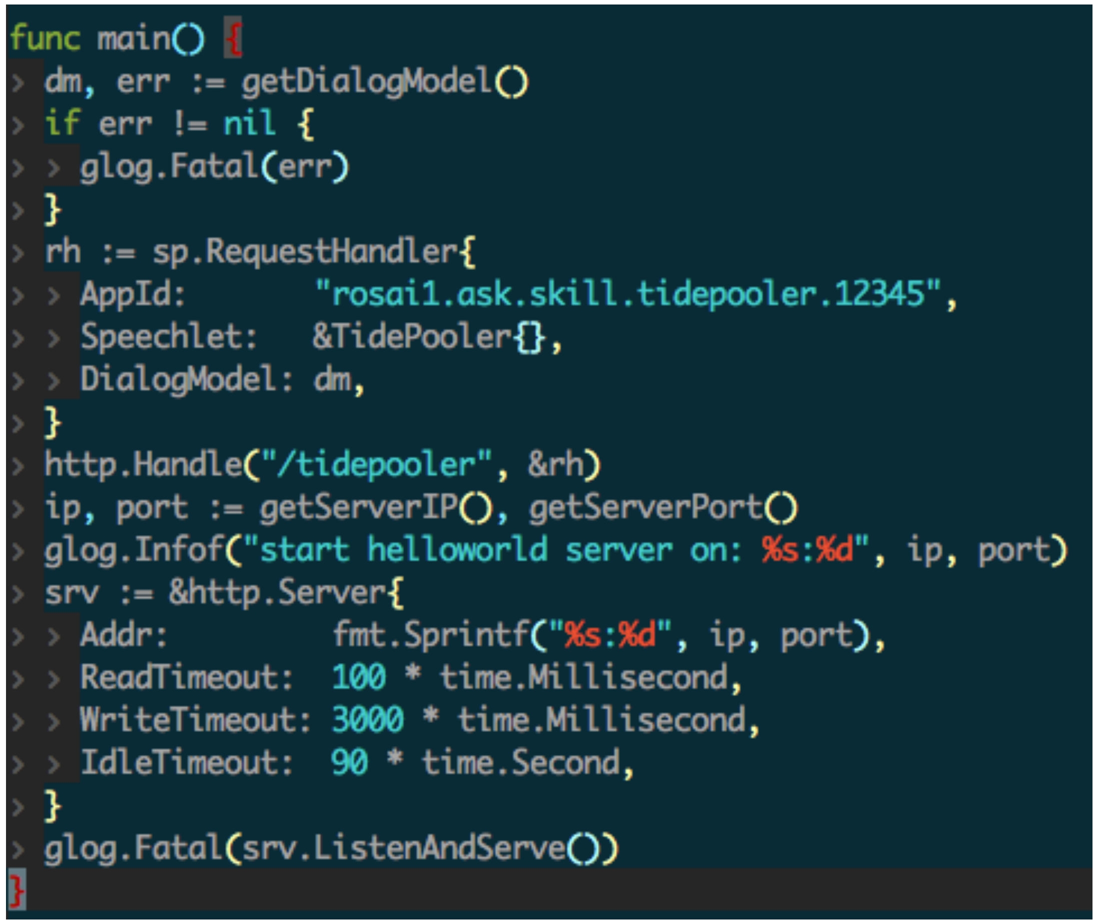
该示例基于rosai-skills-sdk开发

A. 没有槽位引导和确认代理的server开发步骤：

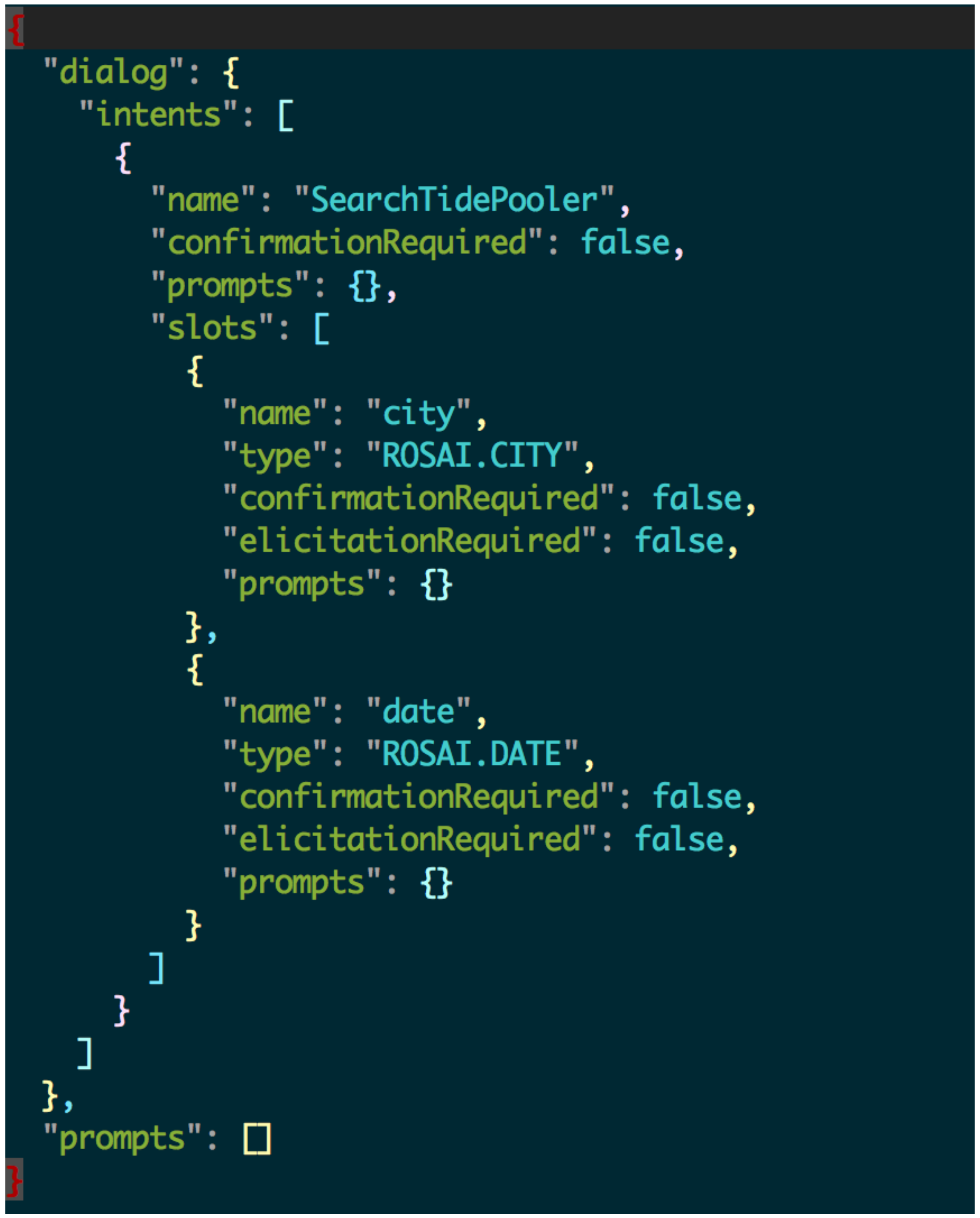
1) 写对话模型，如右图：

2) 注册回调函数，如下图：

3) 实现OnSessionStarted, OnLaunch, OnIntent,
OnSessionEnded 四个方法，如下页图：



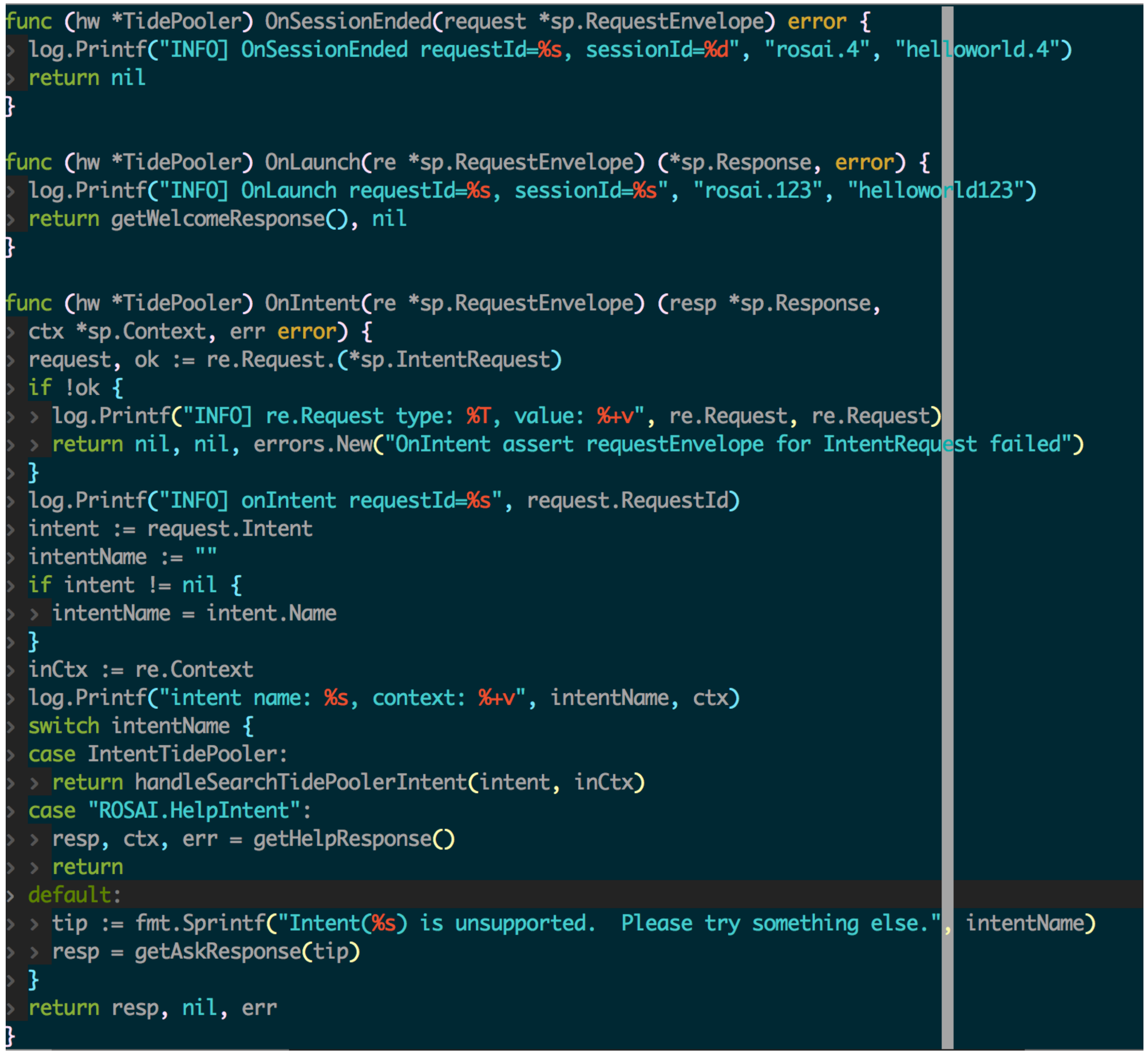
```go
func main() {
	dm, err := getDialogModel()
	if err != nil {
		glog.Fatal(err)
	}
	rh := sp.RequestHandler{
		AppId:       "rosai1.ask.skill.tidepooler.12345",
		Speechlet:   &TidePooler{},
		DialogModel: dm,
	}
	http.Handle("/tidepooler", &rh)
	ip, port := getServerIP(), getServerPort()
	glog.Infof("start helloworld server on: %s:%d", ip, port)
	srv := &http.Server{
		Addr:         fmt.Sprintf("%s:%d", ip, port),
		ReadTimeout:  100 * time.Millisecond,
		WriteTimeout: 3000 * time.Millisecond,
		IdleTimeout:  90 * time.Second,
	}
	glog.Fatal(srv.ListenAndServe())
}
```



```json
{
  "dialog": {
    "intents": [
      {
        "name": "SearchTidePooler",
        "confirmationRequired": false,
        "prompts": {},
        "slots": [
          {
            "name": "city",
            "type": "ROSAI.CITY",
            "confirmationRequired": false,
            "elicitationRequired": false,
            "prompts": {}
          },
          {
            "name": "date",
            "type": "ROSAI.DATE",
            "confirmationRequired": false,
            "elicitationRequired": false,
            "prompts": {}
          }
        ]
      }
    ]
  },
  "prompts": []
}
```

开发server

3. 开发服务



```go
func (hw *TidePooler) OnSessionEnded(request *sp.RequestEnvelope) error {
	log.Printf("INFO] OnSessionEnded requestId=%s, sessionId=%d", "rosai.4", "helloworld.4")
	return nil
}

func (hw *TidePooler) OnLaunch(re *sp.RequestEnvelope) (*sp.Response, error) {
	log.Printf("INFO] OnLaunch requestId=%s, sessionId=%s", "rosai.123", "helloworld123")
	return getWelcomeResponse(), nil
}

func (hw *TidePooler) OnIntent(re *sp.RequestEnvelope) (resp *sp.Response,
	ctx *sp.Context, err error) {
	request, ok := re.Request.(*sp.IntentRequest)
	if !ok {
		log.Printf("INFO] re.Request type: %T, value: %+v", re.Request, re.Request)
		return nil, nil, errors.New("OnIntent assert requestEnvelope for IntentRequest failed")
	}
	log.Printf("INFO] onIntent requestId=%s", request.RequestId)
	intent := request.Intent
	intentName := ""
	if intent != nil {
		intentName = intent.Name
	}
	inCtx := re.Context
	log.Printf("intent name: %s, context: %+v", intentName, ctx)
	switch intentName {
	case IntentTidePooler:
		return handleSearchTidePoolerIntent(intent, inCtx)
	case "ROSAI.HelpIntent":
		resp, ctx, err = getHelpResponse()
		return
	default:
		tip := fmt.Sprintf("Intent(%s) is unsupported.  Please try something else.", intentName)
		resp = getAskResponse(tip)
	}
	return resp, nil, err
}
```

开发server

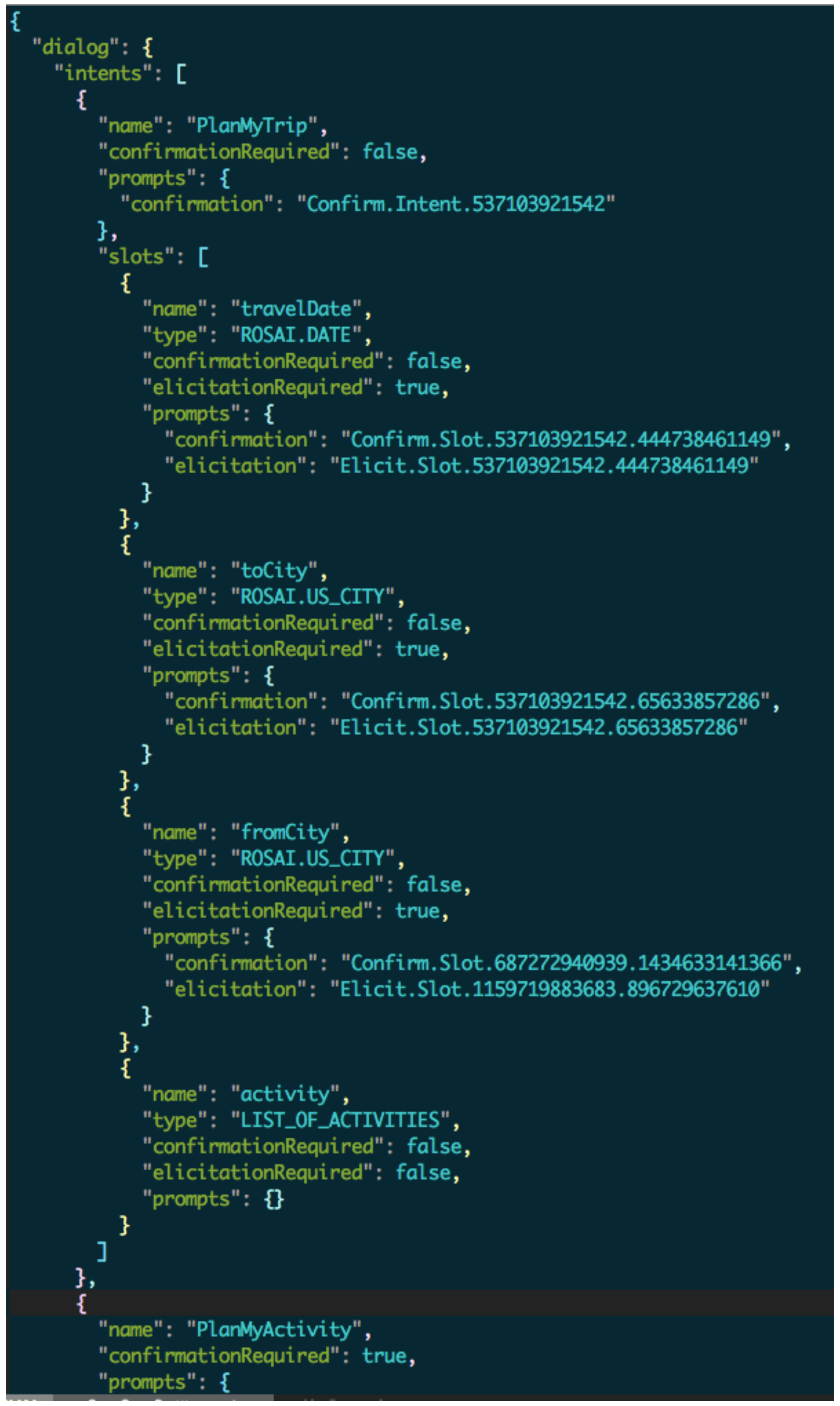
3. 开发服务

B. 槽位引导和确认代理的server开发步骤：

1) 写对话模型，如右图：

2) 注册回调函数，和A一样：

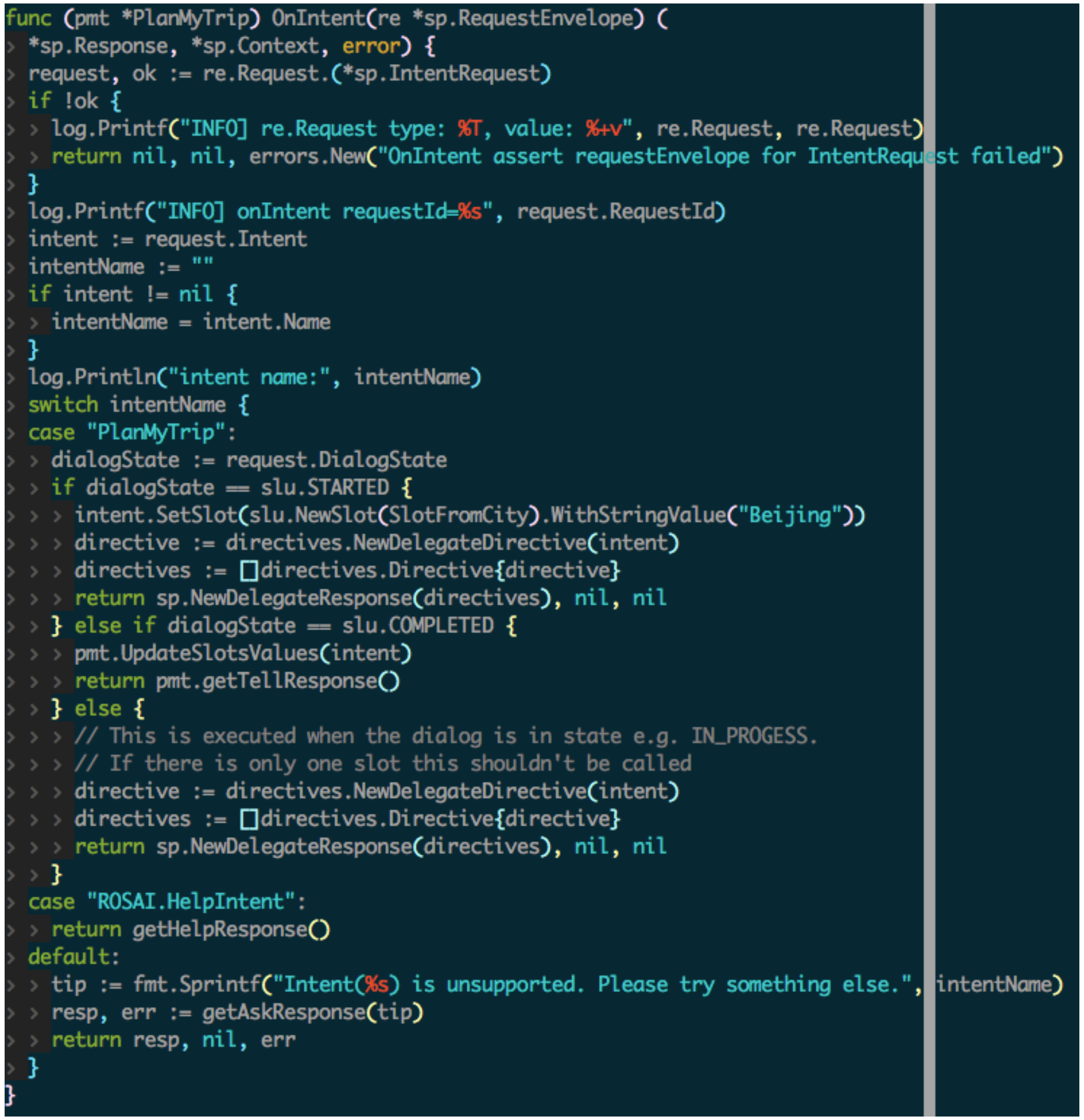
3) 实现OnSessionStarted, OnLaunch, OnIntent, OnSessionEnded 四个方法，如下页图：



```
{
  "dialog": {
    "intents": [
      {
        "name": "PlanMyTrip",
        "confirmationRequired": false,
        "prompts": {
          "confirmation": "Confirm.Intent.537103921542"
        },
        "slots": [
          {
            "name": "travelDate",
            "type": "ROSAI.DATE",
            "confirmationRequired": false,
            "elicitationRequired": true,
            "prompts": {
              "confirmation": "Confirm.Slot.537103921542.444738461149",
              "elicitation": "Elicit.Slot.537103921542.444738461149"
            }
          },
          {
            "name": "toCity",
            "type": "ROSAI.US_CITY",
            "confirmationRequired": false,
            "elicitationRequired": true,
            "prompts": {
              "confirmation": "Confirm.Slot.537103921542.65633857286",
              "elicitation": "Elicit.Slot.537103921542.65633857286"
            }
          },
          {
            "name": "fromCity",
            "type": "ROSAI.US_CITY",
            "confirmationRequired": false,
            "elicitationRequired": true,
            "prompts": {
              "confirmation": "Confirm.Slot.687272940939.1434633141366",
              "elicitation": "Elicit.Slot.1159719883683.896729637610"
            }
          },
          {
            "name": "activity",
            "type": "LIST_OF_ACTIVITIES",
            "confirmationRequired": false,
            "elicitationRequired": false,
            "prompts": {}
          }
        ]
      },
      {
        "name": "PlanMyActivity",
        "confirmationRequired": true,
        "prompts": {
```

开发server

3. 开发服务



```go
func (pmt *PlanMyTrip) OnIntent(re *sp.RequestEnvelope) (
	*sp.Response, *sp.Context, error) {
	request, ok := re.Request.(*sp.IntentRequest)
	if !ok {
		log.Printf("INFO] re.Request type: %T, value: %+v", re.Request, re.Request)
		return nil, nil, errors.New("OnIntent assert requestEnvelope for IntentRequest failed")
	}
	log.Printf("INFO] onIntent requestId=%s", request.RequestId)
	intent := request.Intent
	intentName := ""
	if intent != nil {
		intentName = intent.Name
	}
	log.Println("intent name:", intentName)
	switch intentName {
	case "PlanMyTrip":
		dialogState := request.DialogState
		if dialogState == slu.STARTED {
			intent.SetSlot(slu.NewSlot(SlotFromCity).WithStringValue("Beijing"))
			directive := directives.NewDelegateDirective(intent)
			directives := []directives.Directive{directive}
			return sp.NewDelegateResponse(directives), nil, nil
		} else if dialogState == slu.COMPLETED {
			pmt.UpdateSlotsValues(intent)
			return pmt.getTellResponse()
		} else {
			// This is executed when the dialog is in state e.g. IN_PROGESS.
			// If there is only one slot this shouldn't be called
			directive := directives.NewDelegateDirective(intent)
			directives := []directives.Directive{directive}
			return sp.NewDelegateResponse(directives), nil, nil
		}
	case "ROSAI.HelpIntent":
		return getHelpResponse()
	default:
		tip := fmt.Sprintf("Intent(%s) is unsupported. Please try something else.", intentName)
		resp, err := getAskResponse(tip)
		return resp, nil, err
	}
}
```