TurboModule 搭建

如何创建 TurboModule?

创建一个 `TurboModule` 分为以下步骤：

1. 编写 ArkTS 原生实现代码
2. 编写 JSI 层 C++ 代码
3. 编写 RN 调用 TurboModule 的代码

本文档以示例工程中的 `SampleTurboModule` 为例，介绍了 `TurboModule` 自定义方法的实现步骤。

1. 编写 ArkTS 原生实现代码

`RNPakcage` 提供了 `TurboModulesFactory` 的创建方法， `TurboModulesFactory` 根据名字提供对应的 `TurboModule` 。

原生自定义方法实现

创建名为 `SampleTurboModule.ets` 文件，在文件内创建名为 `SampleTurboModule` 的类，继承自 `TurboModule`，添加自定义方法 pushStringToHarmony(...)：

```
import { TurboModule } from 'rnoh/ts';

export class SampleTurboModule extends TurboModule {
  pushStringToHarmony(arg: string, id?: number): string {
    ...
    return arg;
  }
}
```

TurboModulesFactory 实现

`TurboModulesFactory` 是 `TurboModule` 的工厂类，根据 `name` 创建 `TurboModule`：

```
import {RNPackage, TurboModulesFactory} from 'rnoh/ts';
import type {TurboModule, TurboModuleContext} from 'rnoh/ts';
import {SampleTurboModule} from './SampleTurboModule';

class SampleTurboModulesFactory extends TurboModulesFactory {
  createTurboModule(name: string): TurboModule | null {
    if (name === 'SampleTurboModule') {
      return new SampleTurboModule(this.ctx);
    }
    return null;
  }

  hasTurboModule(name: string): boolean {
    return name === 'SampleTurboModule';
  }
}
```

RNPackage 实现

提供创建 `TurboModulesFactory` 的方法：

```
export class SampleTurboModulePackage extends RNPackage {
  createTurboModulesFactory(ctx: TurboModuleContext): TurboModulesFactory {
    return new SampleTurboModulesFactory(ctx);
  }
}
```

createRNPackages

新建 `RNPackagesFactory.ets` 文件用于提供 `RNPackage`，在 `createRNPackages(...)` 方法中返回需要 `RNPackage`：

```
export function createRNPackages(ctx: RNPackageContext): RNPackage[] {
  return [
    new SampleTurboModulePackage(ctx),
  ];
}
```

将 `createRNPackages` 作为创建 `RNInstance` 时的构造参数传递：

```
const cpInstance: RNInstance = await
this.rnAbility.createAndRegisterRNInstance({ createRNPackages });
```

2. 编写 JSI 层 C++ 代码

NativeSampleTurboModuleSpecJSI

创建名为 `SampleTurboModuleSpec.h` 的声明文件，导出 `NativeSampleTurboModuleSpecJSI`：

```
#include <ReactCommon/TurboModule.h>
#include "RNOH/ArkTSTurboModule.h"

namespace rnoh {
    class JSI_EXPORT NativeSampleTurboModuleSpecJSI : public ArkTSTurboModule {
    public:
        NativeSampleTurboModuleSpecJSI(const ArkTSTurboModule::Context ctx,
const std::string name);
    };
}
```

`SampleTurboModuleSpec.cpp` 实现，注意 `MethodMetadata` 的第一个参数是自定义方法的参数个数：

```
#include "SampleTurboModuleSpec.h"

using namespace rnoh;
using namespace facebook;

static jsi::Value
__hostFunction_NativeSampleTurboCxxModuleSpecJSI_pushStringToHarmony(jsi::Runtim
e &rt, react::TurboModule &turboModule, const jsi::Value *args, size_t count) {
    return jsi::Value(static_cast<ArkTSTurboModule &>(turboModule).call(rt,
"pushStringToHarmony", args, count));
}

NativeSampleTurboModuleSpecJSI::NativeSampleTurboModuleSpecJSI(const
ArkTSTurboModule::Context ctx,
const std::string name): ArkTSTurboModule(ctx, name) {
    methodMap_["pushStringToHarmony"] = MethodMetadata{2,
__hostFunction_NativeSampleTurboCxxModuleSpecJSI_pushStringToHarmony};
}
```

SampleTurboModulePackage.h

提供 `createTurboModuleFactoryDelegate`：

```
#include "RNOH/Package.h"

namespace rnoh {
class SampleTurboModulePackage : public Package {
    public:
        SampleTurboModulePackage(Package::Context ctx) : Package(ctx) {}

        std::unique_ptr<TurboModuleFactoryDelegate>
createTurboModuleFactoryDelegate() override;
    };
} // namespace rnoh
```

SampleTurboModulePackage.cpp

实现 C++ 层的 `createTurboModuleFactoryDelegate` 的 `createTurboModule` 方法，在 `createTurboModule` 方法中需要引用 `NativeSampleTurboModuleSpecJSI`：

```
#include "./TurboModules/SampleTurboModuleSpec.h"

using namespace rnoh;
using namespace facebook;

class SampleTurboModuleFactoryDelegate : public TurboModuleFactoryDelegate {
    public:
    SharedTurboModule createTurboModule(Context ctx, const std::string &name)
const override {
        if (name == "SampleTurboModule") {
            return std::make_shared<NativeSampleTurboModuleSpecJSI>(ctx, name);
        }
        return nullptr;
    };
};

std::unique_ptr<TurboModuleFactoryDelegate>
SampleTurboModulePackage::createTurboModuleFactoryDelegate() {
    return std::make_unique<SampleTurboModuleFactoryDelegate>();
```

```
}
```

PackageProvider.cpp

在 `PackageProvider` 的 `getPackages` 方法实现中添加 `SampleTurboModulePackage`：

```
#include "RNOH/PackageProvider.h"
#include "SampleTurboModulePackage.h"

using namespace rnoh;
std::vector<std::shared_ptr<Package>>
PackageProvider::getPackages(Package::Context ctx) {
    return {
        std::make_shared<SampleTurboModulePackage>(ctx),
    };
}
```

CMakeLists.txt

在 `CMakeLists` 中导入 C++ 文件链接：

```
add_library(rnoh_app SHARED
    "./PackageProvider.cpp"
    "./TurboModules/SampleTurboModuleSpec.cpp"
    "./SampleTurboModulePackage.cpp"
    "${RNOH_CPP_DIR}/RNOHAppNapiBridge.cpp"
)
```

3. 编写 RN 调用 TurboModule 的代码

```
import type {TurboModule} from 'react-native';

interface SpecSampleTurboModule extends TurboModule {
    pushStringToHarmony(arg: string, id?: number): string;
}

const SampleTurboModule =
TurboModuleRegistry.getEnforcing<SpecSampleTurboModule>('SampleTurboModule');
...
```

```
// 调用原生方法
SampleTurboModule.pushStringToHarmony('pages/Details', 1);
```