目录一：ctx-registers

```
app.use(ctx ⇒ {
  // can use
  ctx.$config       //  「只读」 配置对象
  ctx.$data         //  「可修改」 供模板使用的数据：可以在中间件中修改`$data`对象
  ctx.$routes       //  「只读」 server-routes 服务端设定的Url路由
  ctx.$ds           //  「只读」 datasources 数据源
  ctx.$middlewares  //  「只读」 middlewares 中间件列表
  ctx.$tpls         //  「只读」 templates 可以采用的渲染模板
  ctx.$bundles      //  「只读」 bundles 能被渲染的页面(前端源码构建)
  ctx.$logger       //  「只读」 内置logger对象
})
```

- 给ctx扩展新对象 & 且仅且在Node服务启动时进行初始化 ➙ 避免在Koa 请求上下文中重复初始化

```
// 传统方式
import Http form '@…/…/http'
app.use(ctx ⇒ {
  // 每次请求时都会新建一个 Http实例(耗资源)
  const http = new Http()
})
// 扩展Context方式
app.use(ctx ⇒ {
  // 整个Koa应用中，每次请求都复用同一个Http实例
  const http = ctx.http
})
```

举例：doT 模板初始化 & 将相关功能挂载在ctx 上下文对象

```
import doT from ‘dot'
// http://olado.github.io/doT/index.html
export default class DoT {
  constructor() {
    this.partials = {}
    const defs = {
      partials: partialPath ⇒ {
        if (typeof this.partials[partialPath] === 'undefined') {
          throw new Error(`illegal partial path: '${partialPath}'`)
        }
        return this.partials[partialPath]
      },
    }

    glob('**/*.def', { cwd: tplPath, dot: false, sync: true })
      .forEach(file ⇒ this.partials[file.replace(/\.\w+$/, '')] = fs.readFileSync(path.resolve(tplPath, file)))

    glob('**/*.jst', { cwd: tplPath, dot: false, sync: true })
      .forEach(async file ⇒ {
        this[camelCase(file.replace(/\//g, ' ').replace(/\.\w+$/, ''))] = doT.template(
          await fs.readFile(path.resolve(tplPath, file)), null, defs)
      })
  }
}

// 直接使用 ctx.doT.notFound()
app.use( (ctx) ⇒ this.body = ctx.doT.notFound({}))
```