一文带你全方位深入国际化开发

邓敏捷 code秘密花园 1周前

> 邓敏捷，字节 Starling 国际化平台核心开发者。

国际化简介

国际化概念

- **国际化** internationalization(I18n)

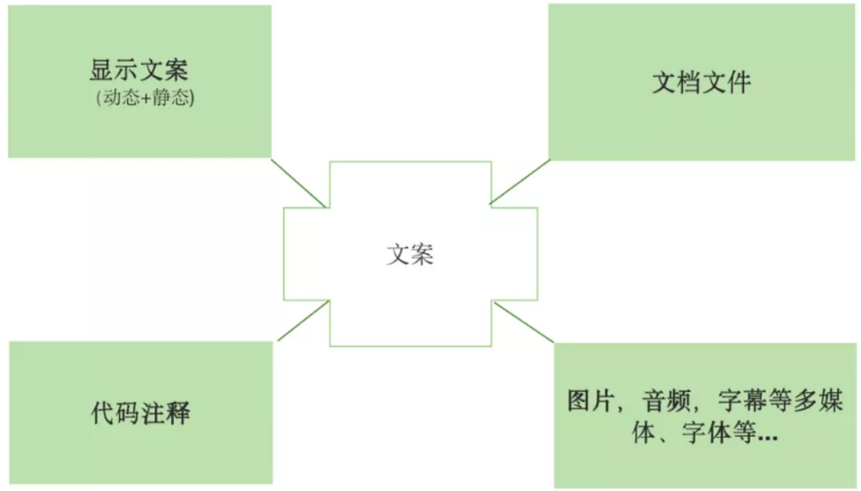
在设计软件，将软件与特定语言及地区脱钩的过程，应用程序的功能和代码设计考虑在不同地区运行的需要，简化不同地区版本的生产

- **本地化** localization (I10n)

移植软件时，加上特定区域信息和翻译文件的过程。如：布局顺序、图片、货币、日期格式...

- **全球化** globalization(g11n)

国际化 + 本地化 = 全球化



国际化对象

网站或者 App 需要进行国际化，大部分需要围绕显示文案、动态文案、代码注释、图片多媒体等四个方面的进行改造：

1. 显示文案（2 种情况）

- 静态文案

> 能够在研发开发的时候规定死的，或者通过某一些规定来推断出的文案

研发依次将这些文案收集起来，通过直接上传到 Starling，或者利用 excel 等中转文件方式后上传到 Starling 平台，等待翻译完成后下载或者 Starling 接口下发文案注入到代码中。

- 动态文案

> 无法通过规律来确定具体的文案(多为用户创造的文案，例如: 评论文案、商品名等)

研发无法在开发时介入对这类文案的改造，可通过机器翻译或者将此类文案在运行时收集起来当做静态文案处理。

2. 代码注释

> 如果开发代码，涉及到多个语种的研发，方便理解代码 or 某些合规需求，那么嵌入到代码注释文案也需纳入改造范围

研发在编码阶段就需要将注释文案处理完成，或者后期排查注释文案，统一进行处理。

3. 相关文档

> 对网站 or App 相关文档(例如使用手册、相关协议等)中的文案

若需求中涉及到文档的翻译改造，那么就需要找专业的翻译员或者将该文档提交 Starling 文档翻译工单。

4. 接入图片,多媒体音频,字体等

> 对改造网站 or App 相关多媒体资源中出现的文案

若需求中涉及到相关图片等的翻译改造，那么就需要找专业的翻译员或者 Staring 官方平台。

国际化流程

理论上，只需要一个可支持 I18n 框架即可。

框架的另外一个含义就是标准，如果没有框架，确实可以自己造一个翻译地区映射机制，但是无法达到统一，明面上实现了国际化，却无法实现本土化(各个国家的单复数，数字，货币等等不同类型的本地化文案)。

可以尝试打开浏览器控制台，输入 Intl 就会出现一个全局 Intl 对象。

国际化发展

> 该部分发展史主要涉及的国际化文案为 **静态显示文案**。

人力阶段

研发需要将项目需求中涉及的文案挨个人工提取以文件形式记录下来(通常为 excel)，上传到 starling 平台或者将文件发送给翻译员，由翻译员上传翻译，最后研发拿到最后的翻译结果文件，以可读取形式(json 等)放入代码仓库。

此阶段涉及到每个需求都要人工介入收集、沟通、等待翻译结果、回填到项目中。如果后期有文案变动需要一次文案变动导致的代码上线。

动态下发

此时期研发依旧需要挨个提取文案，然后上传到 Starling 平台，但是平台侧提供了动态下发文案功能，研发可以在代码里边通过 sdk 的方式动态拉取下来翻译后的文案。此时研发和翻译的动作就开始解耦。

此阶段涉及到每个需求都要人工介入收集、手动上传等机械重复劳动过程。

工具开花

此阶段，产生了丰富的效率工具链生态，研发无需手动收集，也无需手动上传，重复的劳动得到了 Cli 命令化，此外 Starling 平台动态下发能力具备了监控报警功能，也同时提供了浏览器插件 In-Context 设计走查工具、Starling vscode 研发遍历工具。

到达此阶段，基本已经实现了整个流程的自动化操作，但也面临不满意完全自动化结果、工具自动化流程不明确等问题。

有质有效

> 在**I18nOps 标准化**中提及

此阶段将 Starling 发展时期的工具链做了一层平台化的收拢聚合，可视化的展示出整个自动化流程，支持上对自动化结果不满意的二次修改，一次国际化需求流程得到的了标准化，同时实现技术无感化，无论是否有国际化代码开发背景，均可以保证国际化开发质量。

此阶段，完全实现了整个流程的标准化、自动化、可视化，并提供了对整个项目文案的健康管控。

Starling 平台

首先为自己的项目在 Starling 平台上创建项目， 然后一定需要的是为项目建立空间，空间是之后研发拉取文案的集合。

如果需要以产品需求迭代维度管理文案，请建立任务，并为任务匹配对应的同步空间。之后在任务进行的修改/新增/删除文案等操作都会同步到空间中。但是注意的是，空间下直接的修改不会反向同步到任务中。

在任务和空间的页面中就可以进行文案本地化相关的翻译工作。填写/导入不同语种下的翻译文案、截图、注释、最长字符数等信息。

文案编辑正确，并完成审校等工作后，可进行文案的发布：选取发布文案范围，依照自动生成/手动输入的版本号进行发布。发布后研发可拉取本次发布语种的最新文案集合。

非研发关注

PM/UI/QA/

- 产品&运营

关注国际化文案是否符合本地化要求，文案是否全量符合需求节点所要求的状态(如翻译进度、是否上线)...

- 设计

关注不同的国际化文案对于同一份 UI 的不同语种下的展示效果(如长度适配显示)，多语言文案是否可以和翻译平台进行打通...

- 测试

关注在不同语种作用下，是否达到会显示对应不同语种，会不会出现语种错乱的原因...

效率提升

- Starling Figma

为设计稿量身定制 UI 多语种，帮助设计师能够直观感受到一份 UI 在不同语种下的显示，并且为文案本地化前置流程提供可能。

- Starling In-Context

结合上下文环境，给译者提供提供真实的语境，帮助其翻译，提供真实环境，让开发者测试人员能够在线走查的需求产品。

研发关注

开发基准

- **语言地区代码**

“语言地区代码”是指对该地区的语言和文化期望。它使用 BCP 47 中定义的“区域代码”表示。

该代码由连字符（-）分隔的几部分组成。第一部分是表示**语言**的简短字符串。第二个可选部分是表示**地区**的短字符串。另外，可以指定各种扩展名和变体。

通常，Web 应用程序常用到的格式为**语言-地区**。如 en 英语、en-US 说英语的美国地区、en-GB 说英语的英国地区。

更多阅读**语言码 ISO 639 标准**、 **阮一峰解释**（链接见文末）。

- **语言映射关系**

源语言：项目原本的语种，比方说中国地区的开发通常为中文(zh、zh-CN)。

目标 (翻译) 语言：项目运行后需用呈现给用的不同翻译后的语种，比方说通常中台产品会提供中/英语种。

Key: 源语言与目标(翻译)语言的唯一标识，可以解决源语言**一词多义**等问题。

通常研发 RD 在书写代码时候会函数的形式调用，Key 为唯一标识、源语言为兜底文案、目标翻译语言离线存放在本地或者动态线上拉取。

- **开发规则**

ICU 消息格式: 由 **字符串**和 {花括号} 中包含的可变元素**占位符**组成，帮助准确显示本地化(性别、复数形式、数字、时间和日期方面文案)，以表达所有必要的拼写和语法细节。

更多阅读 **ICU Formatting Messages**、 **ICU 在线测试。(链接见文末)**

ECMA-402 Intl 使用规范: ECMAScript 对象的应用程序编程通用接口，它提供语言敏感的字符串比较、数字格式以及日期和时间格式化。

更多阅读 **MDN 解释**。(见文末)

通常，不同的编码语言对映射语种的方案会提供不同的框架，如前端有**starling_intl**、**react-intl**。

动态文案

实际在使用中，避免不了一些动态文案场景的出现(如评论区)，这个时候可以把这些文案节点打上一个动态标识，在运行时调用机器翻译。在可枚举的动态文案也可以由前端和后端约定翻译平台开放接口的使用，达到可枚举文案能够走精确翻译的模式(比方说跨境电商商品名场景)。

效率提升

首先是 VSCode 插件，在研发编写 code，研发在正常书写文案代码 or 代码注释，starling vscode 提供一键上传文案、翻译文案、翻译注释等相关国际化操作。

然后是 CLI 工具，帮助开发者以全局视角来判断项目仓库中国际化的覆盖率，并且也提供快速提升覆盖率的一键式替换等命令行操作。

文案 Key

确定文案 key 的角色会有两类：研发、产品（此处是产品 PM/UX/本地化 PM 的集合）。具体的 key 生成者是由团队本地化与研发节奏合作决定的。

- **研发确定**

由研发确定 key 会相对来说比较符合产品开发的节奏以及 key 本身的需求使用习惯。

key 对于本地化翻译、产品线上显示效果没有直接的影响，更多是为研发服务，能够在代码中将文案与托管在 starling 中的文案进行对齐。

- **产品确定**

由产品侧进行 key 的制定往往是期待整体流程的缩短，能在源文案确定的同时撰写 key，一同提交至 starling 平台，然后让研发与翻译并行。并且 key 由产品/本地化同学书写，能够添加模块/功能等信息，有利于此类角色后期在 starling 平台进行文案管理/检索。

建议

看重**国际化时间**：由产品(如有)制定 key，先建立文案 key 的翻译关系

看重**key 的质量：**由研发同学主导 key 的构建，在代码编写时确定语义 key

翻译模式

- **翻译后置：** 先研发，后本地化的传统国际化流程，适用于国际化时间要求不高/国际化任务量不大/语种数量不过的项目。对于此类用户研发可以在初期在代码中定义 key，当前使用 I18nOps 体系(工具+平台)可完全满足翻译后置的场景~
- **翻译前置：** 研发与本地化翻译工作并行，缩短项目整体时间长度，适合本地化工作体量较大、涉及语种较多的项目，这种项目往往会有较长的多语言译文返回周期，尽量提早开始会降低因语言返回时间较短带来的上线延迟风险。

I18nOps 标准化

参与一次国际化需求中，所涉及到的不同人员可以有不同的工具来提高效率，但是会难免遇到流程不够清晰，数据不够直观展示，国际化需求进度不明显等困扰，造成不必要的时间浪费，以及文案事故。

在通常情况下，我们在做一次需求就会对应一个代码的分支，也就是会存在一个代码仓库，那么就可以精准的关注此次需求的国际化代码质量以及此次需求涉及到的文案的翻译情况。

由此产出 I18nops 平台，以**仓库维度**收敛文案，用户进行授权后，I18nOps 可对代码进行全面的语法分析并提取中静态文案 key，这样就可以得出仓库中的国际化代码是否书写正确，使用到的文案是否翻译、发布等数据。再一个利用平台补充的能力，如补充动态文案 key，过滤不需要国际化的文案，快速翻译未翻译的文案等，可以做到精准的下发兜底使用的文案译文，并且将仓库项目中所有的文案记录到人(研发认领机制)。

与此同时，i18nops 提供一键将仓库分支代码进行国际化工单模式，这对于不同国际化认知的同学统一走工单模式进行国际化需求上线很有帮助，因为这免去了所有国际化的细碎操作(如: 上传、翻译、替换...)。

结合提取的每一次的国际化数据，可以得出代码覆盖率、注释覆盖率、翻译覆盖率等国际化相关的数值，让整体的国际化情况一目了然。

参考资料

[1] 语言码 ISO 639 标准: *https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes*

[2] 阮一峰解释: *http://www.ruanyifeng.com/blog/2008/02/codes_for_language_names.html*

[3] ICU 消息格式: *https://starling.bytedance.net/open/content/9163/111707*

[4] ICU Formatting Messages: *http://userguide.icu-project.org/formatparse/messages*

[5] ICU 在线测试: *https://devpal.co/icu-message-editor/?data=I%20have%20%7Bnum%2C%20plural%2C%20one%7B%7Bnumber%7D%20%23%20apple%7D%20other%7B%23%20apples%7D%7D%2Cbut%20it%27s%20too%20small%0A*

[6] ECMA-402: *https://tc39.es/ecma402/*

[7] Intl 使用规范:: *https://tc39.es/ecma402/*

[8] MDN 解释: *https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Intl*

[9] react-intl: *https://formatjs.io/docs/getting-started/installation/*