

# Chapter 17: Making a Simple Web App, Part 2

By Tim Condon

在上一章中，您学习了如何查看类别以及如何创建、编辑和删除缩略词。在本章中，您将学习如何允许用户以友好的方式向缩略词添加类别。

## 创建包含类别的缩略词

Web应用程序的最终实现任务是允许用户管理缩略词上的类别。当使用REST客户端的API时（如iOS应用程序），您能发送多个请求，每个类别一个。但是，使用Web浏览器这是不可行的。

Web应用程序必须接受一个请求中的所有信息，并将请求转换为适当的Fluent操作。此外，在用户可以选择类别之前就去创建它们并不是良好的用户体验。

打开**Category.swift**并在var **acronyms**下面的**extension**底部添加以下函数：

```
static func addCategory(
    _ name: String,
    to acronym: Acronym,
    on req: Request
) throws -> Future<Void> {
    // 1
    return Category.query(on: req)
        .filter(\.name == name)
        .first()
        .flatMap(to: Void.self) { foundCategory in
            if let existingCategory = foundCategory {
                // 2
                return acronym.categories
                    .attach(existingCategory, on: req)
                    .transform(to: ())
            }
        }
}
```

```

    } else {
        // 3
        let category = Category(name: name)
        // 4
        return category.save(on: req)
            .flatMap(to: Void.self) { savedCategory in
                // 5
                return acronym.categories
                    .attach(savedCategory, on: req)
                    .transform(to: ())
            }
    }
}

```

这是新方法的作用：

1. 执行查询以搜索提供名称的类别。
2. 如果类别存在，建立关系并将结果转换为Void。 ()是Void()的简写。
3. 如果该类别不存在，使用提供的名称创建一个新的Category对象。
4. 保存新类别并解包返回的future。
5. 建立关系并将结果转换为Void。

打开**WebsiteController.swift**并在文件底部添加新的Content类型以处理新数据：

```

struct CreateAcronymData: Content {
    let userID: User.ID
    let short: String
    let long: String
    let categories: [String]?
}

```

这将获取缩略词所需的现有信息，并添加一个可选的字符串数组来表示类别。这允许用户提交现有的和新的类别，而不是仅提交现有的类别。

接下来，将createAcronymPostHandler(:)替换为以下内容：

```

// 1
func createAcronymPostHandler(
    _ req: Request,
    data: CreateAcronymData
) throws -> Future<Response> {
    // 2
    let acronym = Acronym(
        short: data.short,
        long: data.long,

```

```
        userID: data.userID)
// 3
return acronym.save(on: req)
    .flatMap(to: Response.self) { acronym in
        guard let id = acronym.id else {
            throw Abort(.internalServerError)
        }

        // 4
        var categorySaves: [Future<Void>] = []
        // 5
        for category in data.categories ?? [] {
            try categorySaves.append(
                Category.addCategory(category, to: acronym, on: req))
        }
        // 6
        let redirect = req.redirect(to: "/acronyms/\(id)")
        return categorySaves.flatten(on: req)
            .transform(to: redirect)
    }
}
```

这是你改变的：

1. 更改路由处理程序的Content类型以接受CreateAcronymData。
2. 创建一个Acronym对象以保存，因为它不再传递到路由中。
3. 调用flatMap(to:)而不是map(to:)，因为你现在在闭包中返回Future <Response>。
4. 定义一个futures数组来存储保存操作。
5. 遍历提供给请求的所有类别，并将Category.addCategory(\_:to:on:)的结果添加到数组中。
6. Flatten数组以完成所有Fluent操作并将结果转换为Response。将页面重定向到新的缩略词页面。

最后，在boot(router:)中，用以下内容替换创建缩略词的POST路由：

```
router.post(
    CreateAcronymData.self,
    at: "acronyms", "create",
    use: createAcronymPostHandler)
```

这会将content类型更改为CreateAcronymData。

您需要允许用户在创建缩略词时指定类别。打开**createAcronym.leaf**，在<button>部分上方添加以下内容：

```
///  
<div class="form-group">  
  ///  
  <label for="categories">Categories</label>  
  ///  
  <select name="categories[]" class="form-control"  
    id="categories" placeholder="Categories" multiple="multiple">  
  </select>  
</div>
```

这是它的作用：

1. 为给类别设置样式使用form-group类定义一个新的<div>。
2. 为输入指定label标签。
3. 定义<select>输入以允许用户指定类别。multiple属性允许用户指定多个选项。name属性categories[]允许表单将类别作为URL-encoded的数组发送。

目前，表单不显示任何类别。使用<select>输入仅允许用户选择预定义类别。为了用户体验更好，您将使用 [Select2 JavaScript library](#)。

打开**base.leaf**并在<link rel = stylesheet ...下面为Bootstrap样式表添加以下内容：

```
#if(title == "Create An Acronym" || title == "Edit Acronym") {  
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/  
select2/4.0.6-rc.0/css/select2.min.css" integrity="sha384-  
RdQbeSCGSeSdSlTMGnUr2oDJZz0uGjJAKQy1MbKMu8fZT5G0qlBajY0n0sY/hKMK"  
crossorigin="anonymous">  
}
```

这会将Select2的样式表添加到创建和编辑缩略词页面。注意复杂的Leaf语句。在**base.leaf**的底部，删除jQuery的第一个<script>标记，并将其替换为以下内容：

```
///  
<script src="https://code.jquery.com/jquery-3.3.1.min.js"  
integrity="sha384-tsQFqpEReu7ZLhBV2VZLAu7zc0V+rXbYlF2cqB8txI/  
8aZajjp4Bqd+V6D5IgvKT" crossorigin="anonymous"></script>  
///  
#if(title == "Create An Acronym" || title == "Edit Acronym") {  
  <script src="https://cdnjs.cloudflare.com/ajax/libs/select2/4.0.3/js/  
select2.min.js"  
integrity="sha384-222hzb8Z8ZKe6pzP18nTSltQM3PdcAwxWKzG0K0IF+Y3bR0r5n9zdQ  
8yTRHgQkQ" crossorigin="anonymous"></script>
```

```
    /// 3
    <script src="/scripts/createAcronym.js"></script>
  }
```

这是它的作用：

1. 包括完整的jQuery库。Bootstrap仅需要**slim**版本，但Select2需要slim版本中不包含的功能，因此需要完整的库。
2. 如果页面是创建或编辑缩略词页面，包含Select2的JavaScript。
3. 还包括本地**createAcronym.js**。

在终端中，输入以下命令以创建本地JavaScript文件。

```
mkdir Public/scripts
touch Public/scripts/createAcronym.js
```

打开文件并插入以下内容：

```
// 1
$.ajax({
  url: "/api/categories/",
  type: "GET",
  contentType: "application/json; charset=utf-8"
}).then(function (response) {
  var dataToReturn = [];
  // 2
  for (var i=0; i < response.length; i++) {
    var tagToTransform = response[i];
    var newTag = {
      id: tagToTransform["name"],
      text: tagToTransform["name"]
    };
    dataToReturn.push(newTag);
  }
  // 3
  $("#categories").select2({
    // 4
    placeholder: "Select Categories for the Acronym",
    // 5
    tags: true,
    // 6
    tokenSeparators: [' '],
    // 7
    data: dataToReturn
  });
});
```

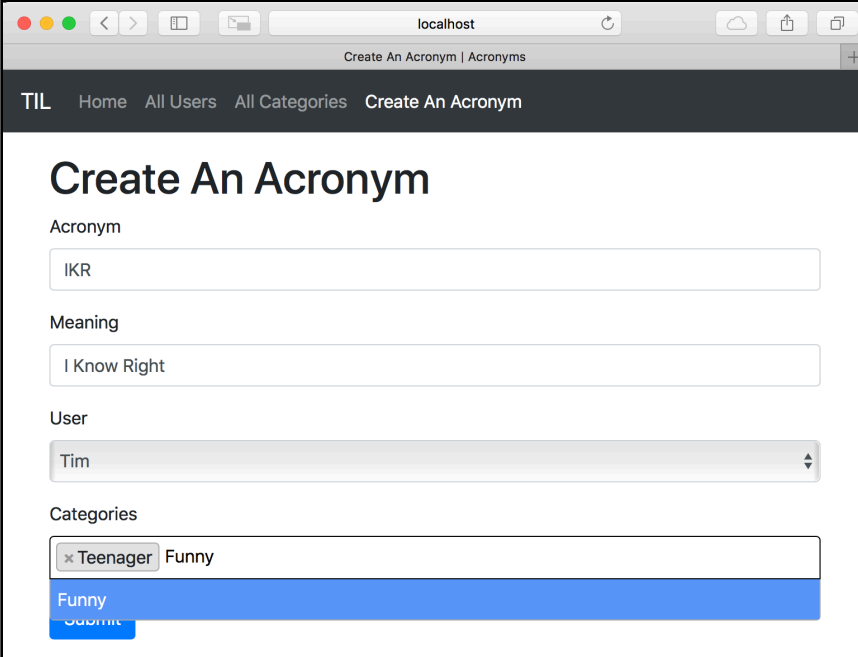
这是脚本的作用：

1. 在页面加载时，向/**api/categories**发送GET请求。这将获得TIL应用程序中的所有类别。
2. 遍历每个返回的类别并将其转换为JSON对象并将其添加到dataToReturn。JSON对象如下所示：

```
{
  "id": <name of the category>,
  "text": <name of the category>
}
```

3. 获取带有ID categories的HTML元素，并在其上调用select2()。这样可以在表单中的<select>上启用Select2。
4. 在Select2输入上设置占位符文本。
5. 在Select2中启用tags。这允许用户动态创建输入中不存在的新类别。
6. 为Select2设置分隔符。当用户键入时，Select2会根据输入的文本创建新类别。这允许用户使用空格分类。
7. 将数据（用户可以选择的选项）设置为现有类别。

保存文件，然后在Xcode中构建并运行应用程序。导航到“**Create An Acronym**”页面。类别列表允许您输入现有类别或创建新类别。该列表还允许您以友好的方式添加和删除“tags”：



The screenshot shows a web browser window with the URL 'localhost' and the page title 'Create An Acronym | Acronyms'. The navigation bar includes links for 'TIL', 'Home', 'All Users', 'All Categories', and 'Create An Acronym'. The main content area is titled 'Create An Acronym' and contains four input fields: 'Acronym' (with 'IKR' entered), 'Meaning' (with 'I Know Right' entered), 'User' (a dropdown menu with 'Tim' selected), and 'Categories' (a multi-select box with 'Teenager' and 'Funny' selected). Below the 'Categories' box is a blue 'Submit' button.

## 显示类别

现在，打开**acronym.leaf**。在“Created By”段落下添加以下内容：

```
/// 1
#if(count(categories) > 0) {
  /// 2
  <h3>Categories</h3>
  <ul>
    /// 3
    #for(category in categories) {
      <li>
        <a href="/categories/#{category.id}">
          #(category.name)
        </a>
      </li>
    }
  </ul>
}
```

这是它的作用：

1. 检查模板上下文是否有任何类别。
2. 如果有，创建标题和<ul>列表。
3. 遍历提供的类别并为每个类别添加一个链接。

保存文件并打开**WebsiteController.swift**。在AcronymContext底部为类别添加新属性：

```
let categories: Future<[Category]>
```

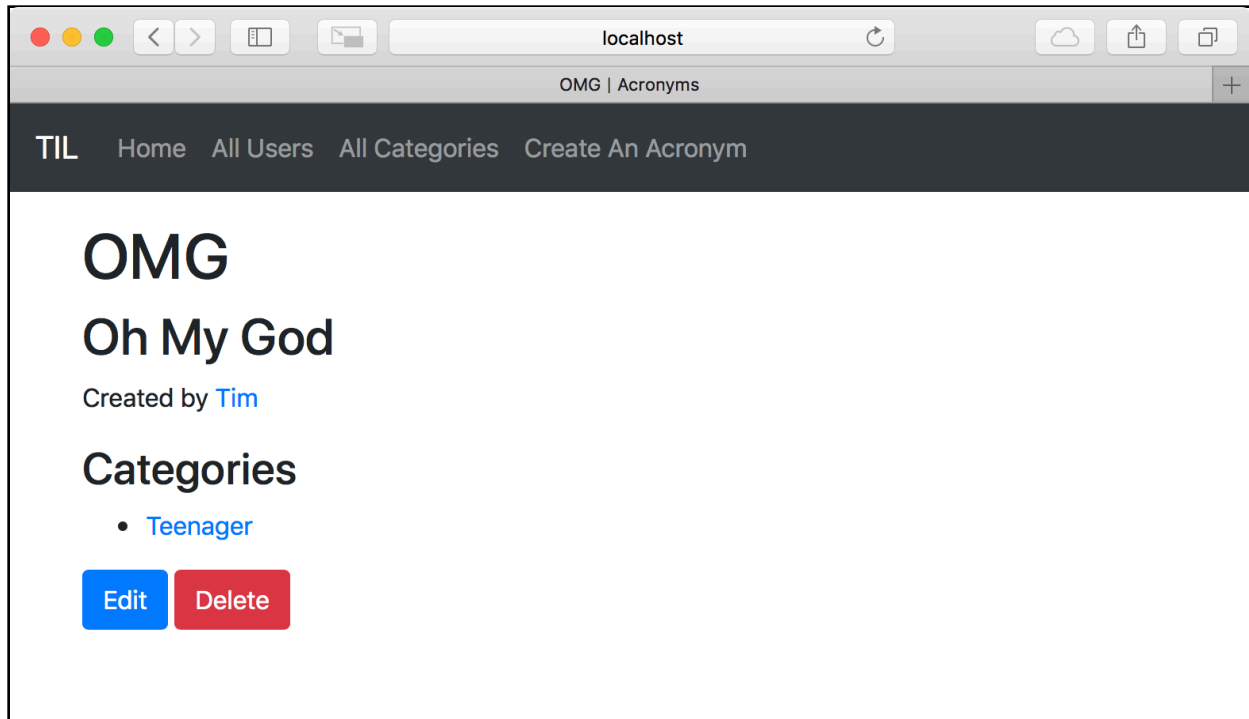
在acronymHandler(:)中，替换：

```
let context = AcronymContext(
  title: acronym.short,
  acronym: acronym,
  user: user)
```

用以下内容：

```
let categories = try acronym.categories.query(on: req).all()
let context = AcronymContext(
  title: acronym.short,
  acronym: acronym,
  user: user,
  categories: categories)
```

再次，这将Future传递给Leaf，它在需要时处理。构建并运行，然后在浏览器中打开创建缩略词页面。在浏览器中创建一个包含类别的缩略词，并转到缩略词的页面。您将在页面上看到缩略词的类别：



## 编辑缩略词

要在编辑缩略词时添加和编辑类别，请打开**createAcronym.leaf**。在类别<div>中，在<select>和</ select>标签之间添加以下内容：

```
#if(editing) {  
  /// 1  
  #for(category in categories) {  
    /// 2  
    <option value="#(category.name)" selected="selected">  
      #(category.name)  
    </option>  
  }  
}
```



这是它的作用：

1. 如果设置了editing标志，则遍历提供的类别数组。
2. 使用selected属性将每个类别添加为<option>。这允许在编辑表单时预先填充类别标签。

保存文件。打开**WebsiteController.swift**并在EditAcronymContext的底部添加一个新属性：

```
let categories: Future<[Category]>
```

在editAcronymHandler(:)中替换：

```
let context = EditAcronymContext(
    acronym: acronym,
    users: User.query(on: req).all())
```

用以下内容：

```
let users = User.query(on: req).all()
let categories = try acronym.categories.query(on: req).all()
let context = EditAcronymContext(
    acronym: acronym,
    users: users,
    categories: categories)
```

这正确地构造了您的新EditAcronymContext。最后，使用以下内容替换editAcronymPostHandler(:)：

```
func editAcronymPostHandler(_ req: Request) throws
-> Future<Response> {
    // 1
    return try flatMap(
        to: Response.self,
        req.parameters.next(Acronym.self),
        req.content
            .decode(CreateAcronymData.self)) { acronym, data in
        acronym.short = data.short
        acronym.long = data.long
        acronym.userID = data.userID

        guard let id = acronym.id else {
            throw Abort(.internalServerError)
        }

        // 2
        return acronym.save(on: req)
            .flatMap(to: [Category].self) { _ in
                // 3
                try acronym.categories.query(on: req).all()
            }.flatMap(to: Response.self) { existingCategories in
                // 4
```

```

    let existingStringArray = existingCategories.map {
        $0.name
    }

    // 5
    let existingSet = Set<String>(existingStringArray)
    let newSet = Set<String>(data.categories ?? [])

    // 6
    let categoriesToAdd = newSet.subtracting(existingSet)
    let categoriesToRemove = existingSet
        .subtracting(newSet)

    // 7
    var categoryResults: [Future<Void>] = []
    // 8
    for newCategory in categoriesToAdd {
        categoryResults.append(
            try Category.addCategory(
                newCategory,
                to: acronym,
                on: req))
    }

    // 9
    for categoryNameToRemove in categoriesToRemove {
        // 10
        let categoryToRemove = existingCategories.first {
            $0.name == categoryNameToRemove
        }
        // 11
        if let category = categoryToRemove {
            categoryResults.append(
                acronym.categories.detach(category, on: req))
        }
    }

    let redirect = req.redirect(to: "/acronyms/\(id)")
    // 12
    return categoryResults.flatten(on: req)
        .transform(to: redirect)
    }
}

```

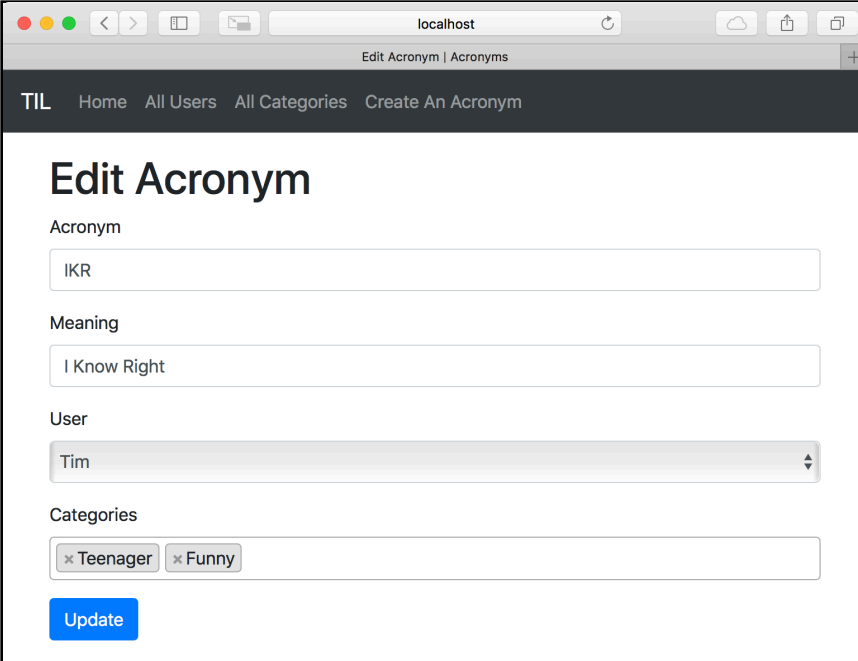
这个新版本的重点是：

1. 请求内容类型更改解码为CreateAcronymData。
2. 在save(on:)上使用flatMap(to:)但返回所有缩略词的类别。注意futures的链接而不是嵌套它们。这有助于提高代码的可读性。
3. 从数据库中获取所有类别。
4. 使用数据库中的类别创建类别名称数组。

5. 为数据库中的类别创建一个Set，为请求提供的类别创建另一个。
6. 计算要添加到缩略词的类别和要删除的类别。
7. 创建一个类别操作结果数组。
8. 遍历所有类别以添加并调用`Category.addCategory(_:to:on:)`来建立关系。将每个结果添加到结果数组中。
9. 循环遍历所有类别名称以从缩略词中删除。
10. 从要删除的类别的名称中获取`Category`对象。
11. 如果`Category`对象存在，请使用`detach(_:on:)`删除关系并删除pivot。
12. Flatten所有类别的future结果。转换结果以重定向到更新的缩略词的页面。

构建并运行，然后在浏览器中打开一个缩略词页面。

单击**Edit**，您将看到填充了现有类别的表单：



The screenshot shows a web browser window at localhost. The page title is 'Edit Acronym | Acronyms'. The navigation bar includes 'TIL', 'Home', 'All Users', 'All Categories', and 'Create An Acronym'. The main content area is titled 'Edit Acronym' and contains the following form fields:

- Acronym:** A text input field containing 'IKR'.
- Meaning:** A text input field containing 'I Know Right'.
- User:** A dropdown menu with 'Tim' selected.
- Categories:** A container with two tags: 'Teenager' and 'Funny', each preceded by an 'x' icon for removal.
- Update:** A blue button at the bottom of the form.

添加新类别，然后单击“**Update**”。页面重定向到缩略词的页面，显示已更新的缩略词。现在尝试从缩略词中删除一个类别。

## 然后去哪儿？

在本节中，您学习了如何创建与iOS应用程序相同功能的全功能Web应用程序。您学习了如何使用Leaf显示不同类型的数据并使用futures。您还学习了如何从Web表单接受数据并为处理数据提供良好的用户体验。

TIL应用程序包含API和Web应用程序。这适用于小型应用程序，但对于非常大的应用程序，您可以考虑将它们拆分为各自的应用程序。然后，Web应用程序会像任何其他客户端那样与API进行通信，例如iOS应用程序。这允许您分别扩展不同的程序部分。大型应用程序甚至可能由不同的团队开发。拆分它们可以让应用程序发展变化，而不依赖于其他团队。

在本书的下一部分中，您将学习如何将身份认证应用于您的应用程序。目前，任何人都可以在iOS应用程序和Web应用程序中创建任何缩略词。这是不可取的，特别是对于大型系统。接下来的章节将向您展示如何使用身份认证保护API和Web应用程序。