15,055,889 名会员

第367

文章 问答 论坛 东西 休息室 ?

Search for articles, questions,

手表

使用 Spring Boot、AngularJS 和 MySQL 设计 UTF-8 编码的 Web 应用程序

韩博孙

2021 年 4 月 15 日 麻省理工学院

评价我: 4.17/5 (3 票)

创建一个简单的 Web 应用程序以支持 UTF-8 编码的字符串

本教程将展示如何使用 Spring Boot、AngularJS 和 MySQL 创建一个可以支持 UTF-8 编码字符串的简单 Web 应用程序。

下载源 - 493.4 KB

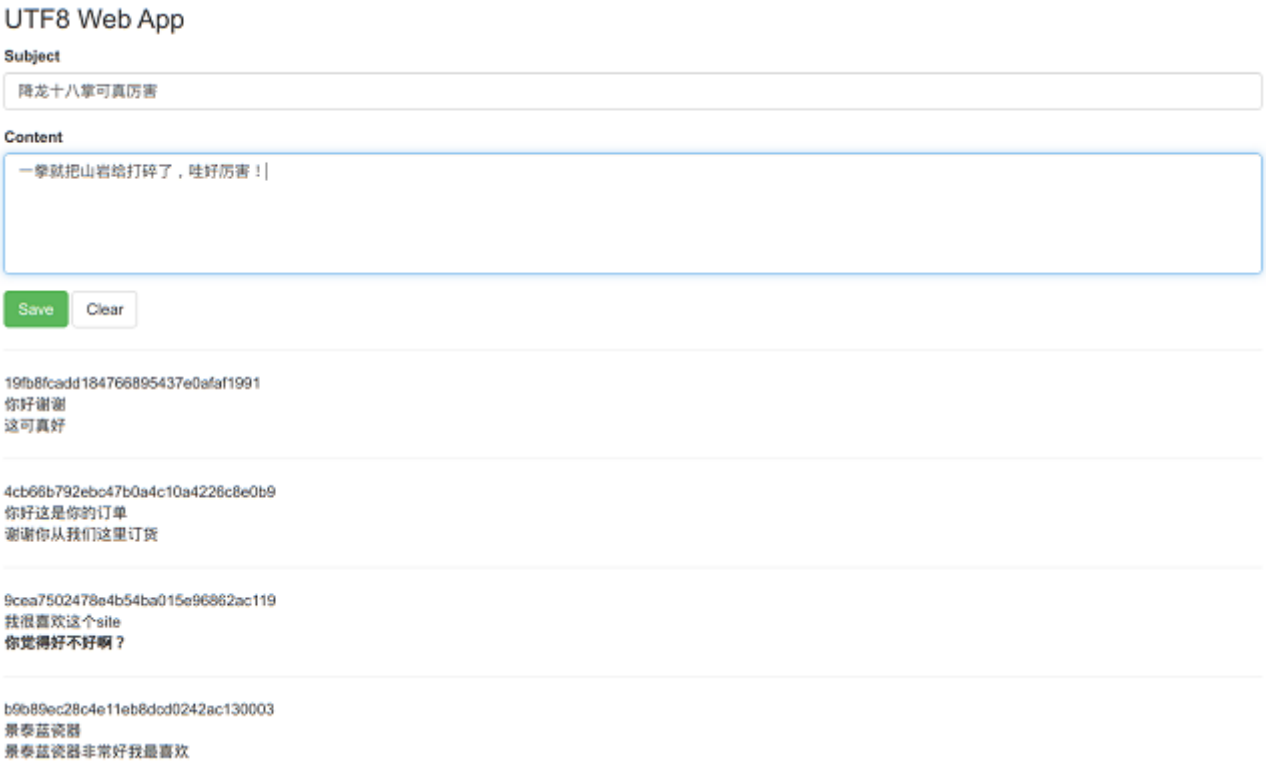
介绍

当我开始编程时，我试图建立非英语网站。它是 Geocity，都是静态页面，我不得不处理代码页（多个字节代表单个字符），特别是中文，例如 GB2312、GB18032 或 BIG5 字符编码。与此同时，国际委员会正在努力制作UNICODE（16位和32位）字符集。后来UTF-8编码问世并成为一种标准，它是一种可变位长字符编码。这甚至比代码页还要糟糕，因为不同的字符可以用不同的字节数来表示它们。如果有人必须为这种字符编码编写解码器，那可能是一场噩梦。这种编码的优点是它是将越来越多的字符添加到字符集中的最灵活的方式。多年来，在它成为标准之后，

这些年来，我不需要处理任何这些。我在美国工作，我必须处理的就是 ASCII。当我忙于自己的生活时，世界仍在继续。技术在发展，处理 UTF-8 编码变得非常容易。十五年前的难题很容易解决。我想如果我忽略这个问题足够长的时间，问题就真的消失了。或者它已经足够愚蠢，以至于可以很麻烦地解决它。

本教程的目的是记录设置可正确处理 UTF-8 编码的基于 Spring Boot 的 Web 应用程序的所有必要步骤。如果我需要做类似的事情，我可以参考本教程。我相信在不久的将来我会将它添加到我的新项目中。把这个放在我能找到的地方会非常有用。

快速免责声明，我创建了本教程及其示例应用程序，而没有进行深入研究。所以我在这方面的知识是有限的。本教程附带的示例应用程序只是一个起点，并不是您需要了解或必须了解的所有内容。会有差距并可能为您带来潜在问题，请自行研究以在出现这些问题时解决。我希望这不会吓到你，如果是这样，请继续阅读。



整体架构

我使用 MySQL 来存储数据。为简单起见，只有一张桌子。数据库及其表设置为使用 UTF-8 字符集。示例应用程序使用 Spring Boot 设计，一个简单的控制器处理来自网页的请求。只有一个页面，AngularJS 应用程序处理用户与页面的交互。

要使所有这些工作，必须完成一些设置/配置：

- 设置数据库和表以处理 UTF8。
- 设置网页，使其能够支持整页显示和使用 UTF8。
- 在网页上，表单必须配置为正确处理 UTF-8 数据（用于请求和响应）。
- 在服务器端，您需要确保 UFT8 字符串可以存储到数据库中并且可以检索回来。

除了这些额外的步骤外，Web 应用程序的配置与不支持 UTF8 的 Web 应用程序相同。在接下来的几节中，我将讨论如何完成这些设置。让我们从数据库和表的创建开始。

MySQL 数据库和表

我做的第一件事就是寻找我必须为 MySQL 数据库和创建表做的配置。那里有很多资源详细解释了如何做到这一点。关键是字符集必须设置为：`utf8mb4`。不能使用 UTF-8 的原因是它与标准 UTF-8 的字符集不同。`charset`MySQL 使用的这个 UTF-8是自定义的字符编码。所以忘记 UTF-8 编码并使用`utf8mb4`. 如果您需要更多信息，请在线查看它们。

这是我创建数据库、访问用户和配置数据库字符集的 SQL 脚本：

SQL 复制代码

```sql
DROP DATABASE IF EXISTS `utf8testdb`;

DROP USER IF EXISTS 'utf8tdbuser'@'localhost';

CREATE DATABASE `utf8testdb`;

CREATE USER 'utf8tdbuser'@'localhost' IDENTIFIED BY '888$Test$888';

GRANT ALL PRIVILEGES ON `utf8testdb`.* TO 'utf8tdbuser'@'localhost';

FLUSH PRIVILEGES;
```

```
ALTER DATABASE `utf8testdb` CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

前两行是删除数据库并删除用户（如果存在）。第三行创建数据库，第四行创建用户并设置密码。第五行将新创建的数据库的所有权限授予用户。第六行将刷新权限，以便用户可以使用它们。最后一行是最重要的一行，它改变了数据库并将数据库使用的字符集设置为utf8mb4，并使用规则utf8mb4_unicode_ci定义了strings的字符序列和排序规则。

创建存储数据的表非常简单，这里是：

SQL 复制代码

```
USE `utf8testdb`;

DROP TABLE IF EXISTS `testcontent`;

CREATE TABLE `testcontent` (
   `id` VARCHAR(34) NOT NULL PRIMARY KEY,
   `subject` VARCHAR(512) NOT NULL,
   `content` MEDIUMTEXT NOT NULL
);
```

由于整个数据库都设置为使用正确的 UTF-8 字符集，因此无需在表级别设置 UTF-8 特定配置。如果不想将整个数据库配置为使用正确的 UTF-8 字符集，则可以在表级别设置 UTF-8。我相信这是可能的。

创建表后，为了测试它，我可以创建一个简单的insert语句并在其中插入一些虚拟数据并查看它是否有效。然后使用一个简单的查询来检索它并查看该值是否正确显示。这是测试insert和查询语句的屏幕截图：

```
1  USE `utf8testdb`;
2
3  INSERT INTO `testcontent` (`id`,
4      `subject`,
5      `content`) VALUES (
6     'b9b89ec28c4e11eb8dcd0242ac130003',
7     '景泰蓝瓷器',
8     '景泰蓝瓷器非常好我最喜欢'
9  );
10
11 SELECT * FROM `testcontent`;
```

接下来，我将讨论如何创建支持 UTF8 的 HTML 页面。

用于 UTF-8 数据处理的 HTML 页面

出于演示目的，Web 应用程序很简单。它有一个表单，允许用户输入消息主题和消息内容，然后单击按钮。主题和内容将发送到后端进行存储。存储完成后，服务器端将进行查询并获取所有存储消息的列表。为了支持 UTF-8 字符处理，我必须对网页做两件事。一种是声明页面使用UTF-8字符集进行显示；另一种是设置charset使用UTF8进行数据交换。他们很简单。这是 Web 应用程序整个页面的源代码：

HTML 缩小▲ 复制代码

```
<!DOCTYPE html>
<html>
   <head>
      <title>AngularJS HTML Editor Directive - by Han</title>
      <meta http-equiv="Content-Type" content="text/html;
      charset=utf-8">
      <link href="/assets/bootstrap/css/bootstrap.min.css" rel="stylesheet">
      <link href="/assets/styles/index.css" rel="stylesheet">
```

```
    </head>
    <body>
       <div class="container" ng-app="testSampleApp"
       ng-controller="testSampleController as vm">

          <div class="row">
             <div class="col-xs-12">
                <h3>UTF8 Web App</h3>
             </div>
          </div>
          <div class="row">
             <div class="col-xs-12">
                <form accept-charset="utf-8">
                   <div class="form-group">
                      <label for="subjectLine">Subject</label>
                      <input id="subjectLine" class="form-control"
                      maxlength="100" ng-model="vm.subjectLine">
                   </div>
                   <div class="form-group">
                      <label for="subjectContent">Content</label>
                      <textarea id="subjectContent"
                                class="form-control fixed-area"
                                maxlength="32767"
                                ng-model="vm.subjectContent"
                                rows="5"></textarea>
                   </div>
                </form>
             </div>
             <div class="col-xs-6">
                <button class="btn btn-success"
                ng-click="vm.saveUtf8Content()">Save</button>
                <button class="btn btn-default"
                ng-click="vm.clearContent()">Clear</button>
             </div>
          </div>

          <div ng-if="vm.postsList && vm.postsList.length > 0">
              <hr>
              <div class="row" ng-repeat="post in vm.postsList">
                 <div class="col-xs-12">
                    <span>{{post.postId}}</span><br>
                    <span>{{post.title}}</span><br>
                   <div ng-bind-html="post.safeHtmlContent"></div>
                   <hr>
                 </div>
              </div>
          </div>

          <footer class="footer">
             <p>© 2021, Han Bo Sun.</p>
          </footer>

       </div>

       <script type="text/javascript"
       src="/assets/jquery/js/jquery.min.js"></script>
       <script type="text/javascript"
       src="/assets/bootstrap/js/bootstrap.min.js"></script>
       <script type="text/javascript"
       src="/assets/angularjs/1.7.5/angular.min.js"></script>
       <script type="text/javascript"
       src="/assets/angularjs/1.7.5/angular-resource.min.js"></script>
       <script type="text/javascript"
       src="/assets/app/js/app.js"></script>
       <script type="text/javascript"
       src="/assets/app/js/testSampleService.js"></script>
     </body>
</html>
```

要声明此网页使用 UTF-8 进行显示，我必须head在页面的部分中添加一个元标记，如下所示：

HTML 复制代码

```
<html>
   <head>
      ...
      <meta http-equiv="Content-Type"
      content="text/html; charset=utf-8">
      ...
   </head>
   ...
</html>
```

这是我定义用于处理用户输入并将用户数据发送到后端的表单的部分，我突出显示了表单的开头，我必须在其中添加属性“accept-charset”。这将强制将accept-charset被添加到将用户数据发送到后端的请求的标头中。这将确保字符串在请求中正确表示。由于数据是用 发送的ngResource，我不确定这是否有必要。但我把它放在表格中以防万一：

HTML 复制代码

```
<div class="row">
   <div class="col-xs-12">
      <form accept-charset="utf-8">
         <div class="form-group">
            <label for="subjectLine">Subject</label>
            <input id="subjectLine" class="form-control"
             maxlength="100" ng-model="vm.subjectLine">
         </div>
         <div class="form-group">
            <label for="subjectContent">Content</label>
            <textarea id="subjectContent"
                        class="form-control fixed-area"
                        maxlength="32767"
                        ng-model="vm.subjectContent"
                        rows="5"></textarea>
         </div>
      </form>
   </div>
   <div class="col-xs-6">
      <button class="btn btn-success"
      ng-click="vm.saveUtf8Content()">Save</button>
      <button class="btn btn-default"
      ng-click="vm.clearContent()">Clear</button>
   </div>
</div>
```

如上所示，“ subjectLine”输入字段绑定到作用域变量vm.subjectLine，“ subjectContent”文本区域绑定到作用域变量vm.subjectContent。没有什么花哨的。将用户请求发送到后端服务器的按钮定义如下：

HTML 复制代码

```
<button class="btn btn-success"
ng-click="vm.saveUtf8Content()">Save</button>
```

还有一个地方是我把数据库中存储的所有消息都显示出来的地方，我只是用一个——列出来ngRepeat，像这样：

HTML 复制代码

```
<div ng-if="vm.postsList &&
vm.postsList.length > 0">
   <hr>
   <div class="row"
   ng-repeat="post in vm.postsList">
      <div class="col-xs-12">
         <span>{{post.postId}}</span><br>
         <span>{{post.title}}</span><br>
         <div ng-bind-html="post.safeHtmlContent"></div>
```

```
            <hr>
        </div>
    </div>
</div>
```

无论如何，这就是 HTML 代码的全部内容。接下来，我将讨论处理数据交换的 AngularJS 代码。

AngularJS 代码

这个 Web 应用程序前端最重要的部分是 AngularJS 代码。我把它分成两个文件。一个称为*app.js*，另一个称为*testSampleService.js*。第一个文件是前端应用程序的主要入口。另一个是向服务器发送数据和从服务器接收数据的服务对象。

这是文件*app.js*的源代码：

JavaScript 缩小▲ 复制代码

```
(function () {
   "use strict";
   var mod = angular.module("testSampleApp",
   [ "testSampleServiceModule" ]);
   mod.controller("testSampleController",
   [ "$scope", "$sce",
   "testSampleService", function ($scope, $sce, testSampleService) {
      var vm = this;

      vm.subjectLine = null;
      vm.subjectContent = null;

      vm.postsList = null;

      loadAllPosts();

      vm.saveUtf8Content = function () {
         var obj = {
            title: vm.subjectLine,
            content: vm.subjectContent
         };

         testSampleService.sendPost(obj).then(function (result) {
            if (result && result.status === true) {
               loadAllPosts();
            }
         }, function (error) {
            if (error) {
               console.log(error);
            }
         });
      };

      vm.clearContent = function () {
         vm.subjectLine = null;
         vm.subjectContent = null;
      };

      function loadAllPosts() {
         testSampleService.listAllPosts().then(function(results) {
            if (results && results.length > 0) {
               vm.postsList = results;

               angular.forEach(vm.postsList, function (post) {
                  if (post) {
                     post.safeHtmlContent = $sce.trustAsHtml(post.content);
                  }
               });
            };
         }, function (error) {
            if (error) {
```

```
                console.log(error);
            }
            vm.postsList = null;
        });
    }
}]);
})();
```

如图所示，为了支持基于 UTF-8 的数据交换，我没有什么特别的事情要做。有一件事是肯定的，AngularJS 是一个现代框架，它支持字符编码而无需太多配置。我相信所有其他框架也会很好地支持这一点。

在上面的代码中，调用了从后端加载所有主题和内容的函数`loadAllPosts()`。这里是：

JavaScript 复制代码

```
function loadAllPosts() {
   testSampleService.listAllPosts().then(function(results) {
      if (results && results.length > 0) {
         vm.postsList = results;

         angular.forEach(vm.postsList, function (post) {
            if (post) {
               post.safeHtmlContent = $sce.trustAsHtml(post.content);
            }
         });
      };
   }, function (error) {
      if (error) {
         console.log(error);
      }
      vm.postsList = null;
   });
}
```

在函数中，我调用了 service `testSampleService`' 方法`listAllPosts()`从后端获取所有消息内容对象。然后我必须将对象的内容转换为安全/可显示的 HTML 字符串。那是 的呼唤`$sce.trustAsHtml()`。

这个*app.js*的另一个重要部分是将消息对象保存到后端的方法。它被称为`vm.saveUtf8Content()`。

JavaScript 复制代码

```
vm.saveUtf8Content = function () {
   var obj = {
      title: vm.subjectLine,
      content: vm.subjectContent
   };

   testSampleService.sendPost(obj).then(function (result) {
      if (result && result.status === true) {
         loadAllPosts();
      }
   }, function (error) {
      if (error) {
         console.log(error);
      }
   });
};
```

服务*testSampleService.js*是为处理与后端的数据交换而创建的。这是整个文件：

JavaScript 缩小▲ 复制代码

```
(function () {
   "use strict";
   var mod = angular.module("testSampleServiceModule",
   [ "ngResource" ]);
   mod.factory("testSampleService", [ "$resource",
      function ($resource) {
```

```
        var svc = {
           sendPost: sendPost,
           listAllPosts: listAllPosts
        };

        var apiRes = $resource(null, null, {
           sendPost: {
              url: "/processPost",
              method: "post",
              isArray: false
           },
           listAllPosts: {
              url: "/listAllPosts",
              method: "get",
              isArray: true
           }
        });

        function sendPost(postInfo) {
           return apiRes.sendPost(postInfo).$promise;
        }

        function listAllPosts() {
           return apiRes.listAllPosts().$promise;
        }

        return svc;
     }
   ]);
})();
```

同样，处理数据交换没有什么特别的需要。后端服务是 RESTFul 服务，所以`ngResource`可以工作。如果我需要任何其他标头信息，我可以将其添加到 API 调用中。但我没有。

下一个。我将讨论后端服务设计。这是我有点不知道的部分。如果您不知道，我在 90 年代开始使用 C/C++。那时的字符都是用字节来表示的，那么就有了双字节的UNICODE字符。那么多字节的字符代码称为代码页，也就是双字节。然后是 4 字节的超级 UNICODE。然后 UTF-8 作为可变长度出现，一个字符可以是 1 个字节、2 个字节或 4 个字节。Java 和 C# 都使用 UNICODE 作为字符。这个应用程序使用 Java，所以我有点不确定 UTF-8 字符是如何存储的。这就是我承认我不太了解编码的地方。问题是我是否需要使用任何编码转换来在 UNICODE 和 UTF-8 之间进行转换。

后端 RESTFul 服务 API

事实证明，我也不需要做任何特别的事情。带有 UTF-8 字符的请求`string`在传递给后端服务端时，`string`将被表示为 UTF-8 并存储在一个 Java`String`对象中。它不是作为双字节 UNICODE`string`或其他编码。这当然非常方便。无论如何，在我进入后端服务的核心之前。我将从头开始。几乎所有这些都是对我过去所做的事情的回顾。所以没有新的惊喜。

让我从后端应用程序的入口点开始。它看起来像这样：

爪哇 复制代码

```
package org.hanbo.boot.rest;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class App
{
   public static void main(String[] args)
   {
      SpringApplication.run(App.class, args);
   }
}
```

这个启动条目是我从过去 3 年的许多过去教程之一中复制和粘贴的内容。我已经解释了这是如何工作的，这里不再赘述。

接下来，我需要一个可以提供 MySQL 数据库连接的配置类。这是从我过去的一个教程中再次复制和粘贴的。我`JdbcTemplate`为这个项目使用了普通的 SQL 查询和更新。这是此配置对象的完整源代码：

爪哇 缩小▲ 复制代码

```
package org.hanbo.boot.rest;

import javax.sql.DataSource;

import org.apache.commons.dbcp2.BasicDataSource;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.datasource.DataSourceTransactionManager;

@Configuration
public class DataAccessConfiguration
{
   @Value("${db.jdbc.driver}")
   private String dbJdbcDriver;

   @Value("${db.conn.string}")
   private String dbConnString;

   @Value("${db.access.username}")
   private String dbAccessUserName;

   @Value("${db.access.password}")
   private String dbAccessPassword;

   @Value("${db.access.validity.query}")
   private String dbAccessValityQuery;

   @Bean
   public DataSource dataSource()
   {
      BasicDataSource dataSource = new BasicDataSource();

      dataSource.setDriverClassName(dbJdbcDriver);
      dataSource.setUrl(dbConnString);
      dataSource.setUsername(dbAccessUserName);
      dataSource.setPassword(dbAccessPassword);
      dataSource.setMaxIdle(4);
      dataSource.setMaxTotal(20);
      dataSource.setInitialSize(4);
      dataSource.setMaxWaitMillis(900000);
      dataSource.setTestOnBorrow(true);
      dataSource.setValidationQuery(dbAccessValityQuery);

      return dataSource;
   }

   @Bean
   public NamedParameterJdbcTemplate namedParameterJdbcTemplate()
   {
      NamedParameterJdbcTemplate retVal
          = new NamedParameterJdbcTemplate(dataSource());
       return retVal;
   }

   @Bean
   public DataSourceTransactionManager txnManager()
   {
      DataSourceTransactionManager txnManager
         = new DataSourceTransactionManager(dataSource());
      return txnManager;
   }
}
```

这比入门类要复杂一些。我要重复我自己，所以你不必查找我以前的教程。首先，我需要从名为*application.properties*的应用程序配置文件加载一些数据。这就是我的做法：

爪哇 复制代码

```
@Value("${db.jdbc.driver}")
private String dbJdbcDriver;

@Value("${db.conn.string}")
private String dbConnString;

@Value("${db.access.username}")
private String dbAccessUserName;

@Value("${db.access.password}")
private String dbAccessPassword;

@Value("${db.access.validity.query}")
private String dbAccessValityQuery;
```

这些行通过键加载配置属性。例如，这一行：

爪哇 复制代码

```
@Value("${db.jdbc.driver}")
private String dbJdbcDriver;
```

它将从*application.properties*文件中读取这一行：

复制代码

```
db.jdbc.driver=com.mysql.cj.jdbc.Driver
...
...
```

接下来，我需要定义一个名为`dataSource`. 这个 Java bean 将指定应用程序的 MySQL 连接信息。

爪哇 复制代码

```
@Bean
public DataSource dataSource()
{
   BasicDataSource dataSource = new BasicDataSource();

   dataSource.setDriverClassName(dbJdbcDriver);
   dataSource.setUrl(dbConnString);
   dataSource.setUsername(dbAccessUserName);
   dataSource.setPassword(dbAccessPassword);
   dataSource.setMaxIdle(4);
   dataSource.setMaxTotal(20);
   dataSource.setInitialSize(4);
   dataSource.setMaxWaitMillis(900000);
   dataSource.setTestOnBorrow(true);
   dataSource.setValidationQuery(dbAccessValityQuery);

   return dataSource;
}
```

接下来，我需要一个始终为我提供`JdbcTemplate`对象的`dataSource`bean ，这将使用之前定义的bean：

爪哇 复制代码

```
@Bean
public NamedParameterJdbcTemplate namedParameterJdbcTemplate()
{
   NamedParameterJdbcTemplate retVal
      = new NamedParameterJdbcTemplate(dataSource());
```

```
   return retVal;
}
```

最后，我需要一个事务管理器。我需要它，因为我可以`@Transactional`对方法使用注释。这允许我根据单个方法中数据库操作的成功提交或回滚更改。这就是我定义这个事务管理器的方式：

爪哇 复制代码

```
@Bean
public DataSourceTransactionManager txnManager()
{
   DataSourceTransactionManager txnManager
      = new DataSourceTransactionManager(dataSource());
   return txnManager;
}
```

一旦我有了应用程序和这个`DataAccessConfiguration`类的入口，我就可以创建 API 控制器和服务类。该`Service`班是使用了一个`JdbcTemplate`执行CRUD操作。API 控制器将处理来自用户的 HTTP 请求。

这是 RESTFul API 控制器的完整源代码：

爪哇 缩小▲ 复制代码

```
package org.hanbo.boot.rest.controllers;

import java.util.ArrayList;
import java.util.List;

import org.hanbo.boot.rest.models.PostDataModel;
import org.hanbo.boot.rest.models.StatusModel;
import org.hanbo.boot.rest.services.PostDataService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class SampleFormController
{
   @Autowired
   private PostDataService postDataSvc;

   @RequestMapping(value="/processPost",
                   method=RequestMethod.POST,
                   consumes = "application/json",
                   produces = "application/json")
   public ResponseEntity<StatusModel>
   processPost(@RequestBody PostDataModel postContent)
   {
      StatusModel resp = new StatusModel();
      if (postContent != null)
      {
         System.out.println
         (String.format("Title: [%s]", postContent.getTitle()));
         System.out.println
         (String.format("Content: [%s]", postContent.getContent()));

         String postId = postDataSvc.savePostData(postContent);
         if (postId != null && !postId.isEmpty())
         {
            resp.setStatus(true);
            resp.setMessage("Post has been processed successfully. postID: " + postId);
         }
         else
         {
            resp.setStatus(true);
```

```
            resp.setMessage("Unable to save post entity. Unknown error.");
         }
      }
      else
      {
         resp.setStatus(true);
         resp.setMessage("Unable to save post entity. No valid post object available.");
      }

      ResponseEntity<StatusModel> retVal = ResponseEntity.ok(resp);

      return retVal;
   }

   @RequestMapping(value="/listAllPosts",
                   method=RequestMethod.GET,
                   produces = "application/json")
   public ResponseEntity<List<PostDataModel>> getAllPosts()
   {
      List<PostDataModel> retList = postDataSvc.getAllPostData();
      if (retList == null)
      {
         retList = new ArrayList<PostDataModel>();
      }

      ResponseEntity<List<PostDataModel>>
      retVal = ResponseEntity.ok(retList);

      return retVal;
   }
}
```

将对象保存到数据库的方法称为`processPost`。映射到此方法的 URL 是“ */processPost* ”。在这个方法中，当我得到请求对象时，我会使用将请求`System.out.println()`的主题和内容属性打印到控制台。下面是数据输出到控制台时的截图：

控制器将繁重的工作交给了名为 的服务对象`postDataSvc`。它的类型为`PostDataService`。这是这个的完整源代码`PostDataService`：

爪哇 缩小▲ 复制代码

```
package org.hanbo.boot.rest.services;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.UUID;

import org.hanbo.boot.rest.models.PostDataModel;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Service;
```

```
import org.springframework.transaction.annotation.Transactional;

@Service
public class PostDataServiceImpl
   implements PostDataService
{
   private final String sql_insertPostData =
   "INSERT INTO `testcontent` (id, subject, content)
   VALUES (:id, :title, :content);";

   private final String sql_queryAllPosts =
   "SELECT id, subject, content FROM `testcontent` LIMIT 1000;";

   @Autowired
   private NamedParameterJdbcTemplate sqlDao;

   @Override
   @Transactional
   public String savePostData(PostDataModel dataToSave)
   {
      if (dataToSave != null)
      {
         String title = dataToSave.getTitle();
         if (title == null || title.isEmpty())
         {
            throw new RuntimeException("Title is NULL or empty");
         }

         String content = dataToSave.getContent();
         if (content == null || content.isEmpty())
         {
            throw new RuntimeException("Content is NULL or empty");
         }

         Map<String, Object> parameters = new HashMap<String, Object>();

         String postId = generateId();
         parameters.put("id", postId);
         parameters.put("title", dataToSave.getTitle());
         parameters.put("content", dataToSave.getContent());

         int updateCount = sqlDao.update(sql_insertPostData, parameters);
         if (updateCount > 0)
         {
            return postId;
         }
      }

      return "";
   }

   @Override
   public List<PostDataModel> getAllPostData()
   {
      List<PostDataModel> retVal = new ArrayList<PostDataModel>();

      retVal = sqlDao.query(sql_queryAllPosts,
            (MapSqlParameterSource)null,
            (rs) -> {
               List<PostDataModel> foundObjs = new ArrayList<PostDataModel>();
               if (rs != null)
               {
                  while (rs.next())
                  {
                     PostDataModel postToAdd = new PostDataModel();
                     postToAdd.setPostId(rs.getString("id"));
                     postToAdd.setTitle(rs.getString("subject"));
                     postToAdd.setContent(rs.getString("content"));
```

```
                    foundObjs.add(postToAdd);
                }
            }

            return foundObjs;
        });

    return retVal;
  }

  private static String generateId()
  {
    UUID uuid = UUID.randomUUID();
    String retVal = uuid.toString().replaceAll("-", "");

    return retVal;
  }
}
```

我将服务拆分为一个接口和一个实现类。上面的代码是实现类。在其中，我使用了`JdbcTemplate`来处理数据访问代码。如果您查看上面的代码，就没有什么特别的了。如果您有兴趣，这里是将记录插入数据库的代码：

爪哇 复制代码

```
Map<String, Object> parameters = new HashMap<String, Object>();

String postId = generateId();
parameters.put("id", postId);
parameters.put("title", dataToSave.getTitle());
parameters.put("content", dataToSave.getContent());

int updateCount = sqlDao.update(sql_insertPostData, parameters);
if (updateCount > 0)
{
  return postId;
}
```

这是从表中查询所有已保存行的代码：

JavaScript 复制代码

```
retVal = sqlDao.query(sql_queryAllPosts,
      (MapSqlParameterSource)null,
      (rs) -> {
        List<PostDataModel> foundObjs = new ArrayList<PostDataModel>();
        if (rs != null)
        {
          while (rs.next())
          {
            PostDataModel postToAdd = new PostDataModel();
            postToAdd.setPostId(rs.getString("id"));
            postToAdd.setTitle(rs.getString("subject"));
            postToAdd.setContent(rs.getString("content"));

            foundObjs.add(postToAdd);
          }
        }

        return foundObjs;
      });
```

这里的技巧是，当请求数据进来时，它会以正确的字符集编码保存到数据库中。并且数据库是为数据正确配置的，所以在检索行时，数据和原来一样。因此，当检索到数据时，它们可以像在原始请求中一样返回。

POM 文件

我想提到的最后一件事是这个项目的 Maven POM 文件。我想指出这个文件中只有一个属性，它将源文件编码指定为 UTF-8。如果源文件或静态文件使用 UTF-8 字符集，这很有用。它还删除了构建警告：“`[WARNING] Using platform encoding (CP1251 actually) to copy filtered resources, i.e. build is platform dependent!`”

这是我指定此类配置的方式：

XML 复制代码

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    ...

    <properties>
      ...
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    ...
</project>
```

这就是这个示例应用程序的全部内容。现在已经讨论了这个应用程序的所有内外部，是时候测试它了。

如何运行示例应用程序

下载 zip 格式的源文件后，首先应将所有*.sj文件重命名为*.js文件。

然后使用终端（或命令提示符）转到解压缩源代码的基本文件夹。然后输入以下命令：

复制代码

```
mvn clean install
```

成功构建项目后，在同一终端/命令提示符下，运行以下命令：

复制代码

```
java -jar target/hanbo-agular-utf8webapp-1.0.1.jar
```

命令成功运行后，使用浏览器并导航到：

复制代码

```
http://locahost:8080/
```

当网页出现时，您可以在主题行和内容文本区域输入一些内容，包括英语、中文、日语或韩语。您还可以尝试一些其他语言，如希伯来语、阿拉伯语或印地语。然后单击名为“**保存**”的绿色按钮。保存的请求数据将显示在底部：

9cea7502478e4b54ba015e96862ac119
我很喜欢这个site
你觉得好不好啊？

---

b4ac347e7bf84976b8fa9bdd625e5ead
前田夕暮
冬が来た。白い樹樹の光を 体のうちに蓄積しておいて、夜深く眠る

---

b9b89ec28c4e11eb8dcd0242ac130003
景泰蓝瓷器
景泰蓝瓷器非常好我最喜欢

---

d16da06ef86b46a19a9e7f1b90bb7621
한국어 사랑고백 시
모래알 하나를 보고도 너를 생각했지 너를 생각하는 것이 나의 일생이었지 풀잎 하나를 보고도 너를 생각했지

如图所示，我尝试过中文、日文和韩文。保存到数据库后，所有这些都被正确检索和显示。当我在 MySQL Workbench 中运行查询时，我可以在数据库表中看到这些：

```
11 • USE `utf8testdb`;
12 • SELECT * FROM `testcontent`;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

#	id	subject	content
1	19fb8fcadd184766895437e0afaf1991	你好谢谢	这可真好
2	4cb66b792ebc47b0a4c10a4226c8e0b9	你好这是你的订单	谢谢你从我们这里订货
3	71fa1f71600f4877aaa303485da4d9be	降龙十八掌可真...	一拳就把山岩给打碎了，哇好厉害！
4	9cea7502478e4b54ba015e96862ac119	我很喜欢这个site	你觉得好不好啊？
5	b4ac347e7bf84976b8fa9bdd625e5ead	前田夕暮	冬が来た。白い樹樹の光を 体のうちに蓄積しておいて、夜深く眠る
6	b9b89ec28c4e11eb8dcd0242ac130003	景泰蓝瓷器	景泰蓝瓷器非常好我最喜欢
7	d16da06ef86b46a19a9e7f1b90bb7621	한국어 사랑고백 시	모래알 하나를 보고도 너를 생각했지 너를 생각하는 것이 나의 일생이었지 풀잎 하나를 보고도 너를 생각했지
*	NULL	NULL	NULL

两者（页面显示和数据库查询）都应验证端到端应用程序执行是否正确。但是，正如我在免责声明中所说的那样，我不是在 Web 应用程序中使用 UTF-8 字符集的专家，所以可能有一些我没有做的事情，应用程序可能会崩溃。我希望这不会发生。无论如何，请按原样使用示例项目。

概括

使用 UTF-8 作为基本字符集编写基于 Web 的应用程序是我一直想做的事情。但直到现在，我才迈出了这第一步。我创建了这个教程，以便我可以参考，以防我将来必须做类似的事情。我以为这可能很难。原来这并不难。我想以我的经验和技能水平，Web 应用程序设计将不再困难。

最大的挑战是数据库配置。MySQL 的诀窍是使用 utf8mb4 编码而不是 UTF-8 编码。接下来，我必须确保网页使用 UTF-8 作为字符编码，表单使用 UTF-8 作为字符集。Java端没有变化，因为Java字符串对象可以存储UTF-8，不需要字符串转换。只要字符串数据正确编码（以 UTF-8 格式），示例应用程序的端到端测试运行就会正常运行。

创建这个示例应用程序真的很有趣。当然，我学到了一些有价值的东西。希望对各位读者有用。如果你看到任何奇怪的东西，请写在本教程的评论部分。这对其他人和我都有帮助。谢谢您阅读此篇。

历史

- 2021 年 4 月12日 - 初稿

执照

本文以及任何相关的源代码和文件均在MIT 许可下获得许可

分享

关于作者

韩博孙

组长 The Judge Group

美国

手表该会员

没有提供传记

评论和讨论

添加评论或问题

电子邮件提醒

Search Comments

第一 页上一页 下一页

我的一票

MoPHL 21-Apr-21 0:17

刷新 1

一般 新闻 建议 问题 错误 答案 笑话 赞美 咆哮 管理员

使用Ctrl+Left/Right 切换消息，Ctrl+Up/Down 切换主题，Ctrl+Shift+Left/Right 切换页面。

文章 Copyright 2021 by Han Bo Sun
所有其他 版权所有 © CodeProject ,
1999-2021 Web01 2.8.20210930.1