

# Maharishi International University

Enterprise Architecture (EA)  
CS544

## Slang Dictionary

TEAM – 6

Bleard Rexhaj, ID: 611019

Erdenesaikhan Tserendavga, ID: 611104

Enkhjargal Gansukh, ID: 611031

*Professor: Joe Bruen*

*August-2020*

[Home](#) [Categories](#) [Login!](#) [Register!](#)



### Random Slang Words

Search through thousands of slang words.

**hello**

a very offensive curse word



2017-01-14 18:00:00.0 by [i am multi coloured](#)

 632  85

**Hello**

The only word on this site that has nothing to do with sex or drugs!



2007-02-21 18:00:00.0 by [pirates rule!!!](#)

 3242  594

**hello**

It's a form of salutation of greeting in the English language, aslo used at telephone.



2017-07-26 19:00:00.0 by [RetardWolf](#)

 131  20

**MVC**

Acronym for Model View Controller, a software architecture pattern/[framework]/paradigm related to graphical user interfaces.

2013-10-12 19:00:00.0 by [CanonCat](#)

 13  0

#### About

Slang dictionary will be a project where users could simply browse new slang words every day. The web application ?SLANG DICTIONARY? will provide an easy and convenient way for users to register and add slang words, the normal guest users can simply browse through and read about the words they are interested in. Registration will require a valid Email address for identity verification, guests will be able to add comments and to vote regarding the words, Word publishers will be able to edit their slang added words.

#### Archives

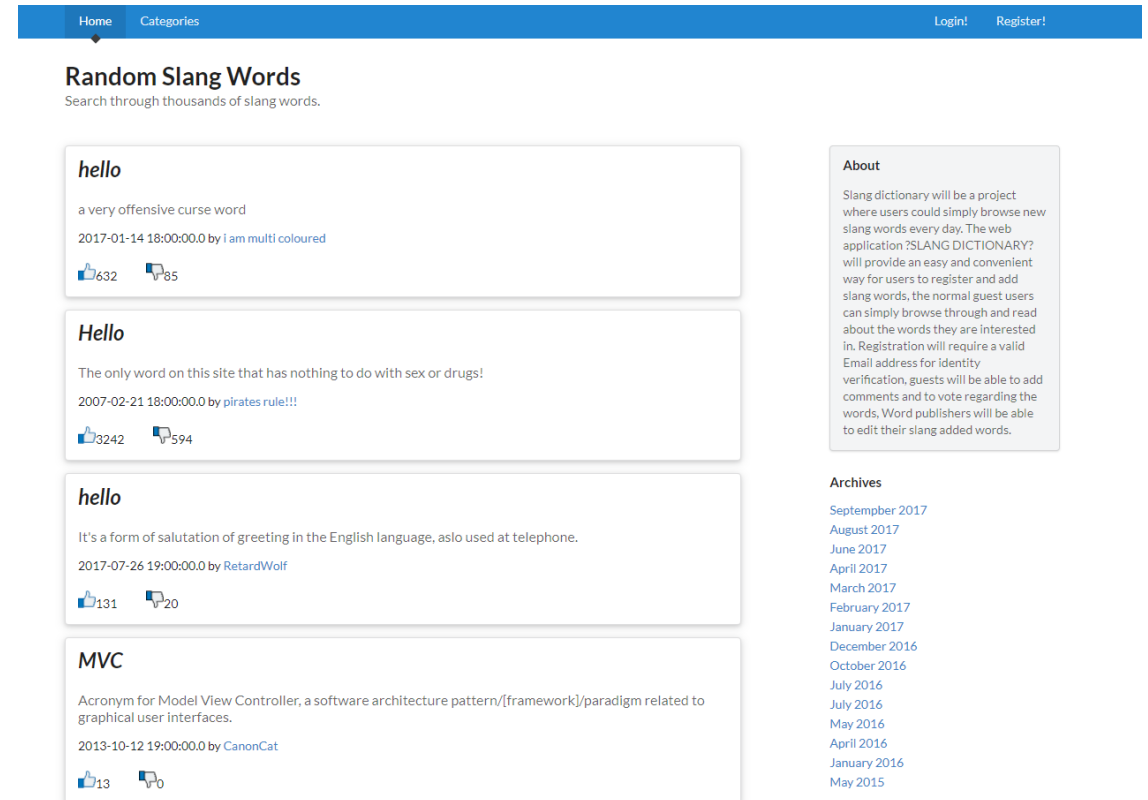
- [Septempber 2017](#)
- [August 2017](#)
- [June 2017](#)
- [April 2017](#)
- [March 2017](#)
- [February 2017](#)
- [January 2017](#)
- [December 2016](#)
- [October 2016](#)
- [July 2016](#)
- [July 2016](#)
- [May 2016](#)
- [April 2016](#)
- [January 2016](#)
- [May 2015](#)

# Contents

- Introduction
- Purpose
- Technologies and tools used
- Modules
- App Screenshots
- Future Extension

# Introduction

Slang dictionary is a project where users can simply browse new slang words every day. The web application “SLANG DICTIONARY” will provide an easy and convenient way for users to register and add slang words, the normal guest users can simply browse through and read about the words they are interested in. Registration will require a valid Email address for identity verification, guests will be able to add comments and to vote regarding the words, Word publishers will be able to edit their slang added words.



# Purpose

- To learn and implement some of the Main Technologies that have been covered in the course lectures and to broaden our knowledge enhancing the capability of interpreting theory into practice.
- To be able to understand and design better software applications using combination of different tools and technologies .
- To solve problems by applying pattern programming.

# Technologies and tools used

- Spring boot
- Hibernate JPA
- Email
- Spring Security
- AOP
- Testing
- MySQL
- GitHub
- Hibernate Performance
- Validation



# User Module

- The implementation of a back to front module was made, A user will be able to register / login, The technologies used on this module are Validation, Aspect Oriented Programing, Hibernate JPA, Security, Database ORM, Spring Email.
- Validation – Validation was made by spring validation, main validation annotation used are @NotEmpty, @Email and custom validations like @EmptyOrSize, @FieldMatch that will check whether the password and confirm password will match.
- Spring Email - Spring Email was used in order to complete the implementation of the user, so whenever a user is registered it will be in a disabled state, an email will be generated and sent out and when the user clicks the generated link that includes the token his data will be enabled and ready for use.
- AOP – Aspect oriented programing was used in the User Module with the @Before and @After annotations, there was the need to encrypt the password and add a default role before the user was being persisted to the database and also to send an confirmation email to the user in order for him to confirm his account, and aop was the perfect tool to solve this cross-cutting concern.
- Security - Spring security was user to authorize the user based on his authorities.
- ORM/Hibernate - Hibernate was used to persist ORM java objects.

# User Module Examples - Validation

Country should not be empty  
Password should not be empty  
Street should not be empty  
Password should be between 6 and 60  
First Name must not be empty and should be between 3 and 20  
Last Name must not be empty and should be between 3 and 20  
Zip Code must be between 5 and 9  
Email should not be empty  
City should not be empty



```
User.name= {0} must not be empty and should be between {2} and {1}  
User.email = {0} must be an valid email address  
User.address.zip = {0} must be between 5 and 9  
User.email.Empty = {0} should not be empty  
User.address = {0} should not be empty  
User.password={0} must match {1}  
Password.NotEmpty= {0} should not be empty  
Password.Size = {0} should be between {2} and {1}
```

```
firstName = First Name  
lastName = Last Name  
email = Email
```

```
singleAddress.zipCode = Zip Code  
singleAddress.street = Street  
singleAddress.city = City  
singleAddress.country = Country
```

```
userCredentials.password=Password  
password = Password  
verifyPassword = Verify Password
```

# User Module – Spring Email

```
EmailConfirmationToken confirmationToken = new EmailConfirmationToken(user);

emailService.save(confirmationToken);

SimpleMailMessage mailMessage = new SimpleMailMessage();
mailMessage.setTo(user.getEmail());
mailMessage.setSubject("Complete Registration!");
mailMessage.setFrom("{spring.mail.username}");
mailMessage.setText("To confirm your account, please click here : "+
    "http://localhost:8080/confirm-account?token="+confirmationToken.getConfirmationToken());

emailService.sendEmail(mailMessage);
```

```
@Autowired
EmailService emailService;
```

```
public interface EmailService {
    void save(EmailConfirmationToken emailConfirmationToken);

    @Async
    void sendEmail(SimpleMailMessage email);

    EmailConfirmationToken findByConfirmationToken(String confirmationToken);
}
```



# User Module – AOP

```
@Before("addAddress() && addArgs(user)")
public void beforeExecution(JoinPoint joinPoint, User user){
    //Assigning the address to the user.
    Set<Address> addresses = new HashSet<>();
    addresses.add(user.getSingleAddress());
    user.setAddress(addresses);

    //Adding a Role User to user by default.
    UserCredentials userCredentials = new UserCredentials();
    userCredentials.setEnabled(false);
    userCredentials.setUserName(user.getFirstName());
    BCryptPasswordEncoder bCryptPasswordEncoder = new BCryptPasswordEncoder();
    userCredentials.setPassword(bCryptPasswordEncoder.encode(user.getUserCredentials().getPassword()));
    userCredentials.setVerifyPassword(userCredentials.getPassword());

    //Setting up Roles for this specific user.
    List<Authority> authorityList = new ArrayList<>();
    Authority authority = new Authority();
    authority.setAuthority("USER");
    authority.setUsername(user.getFirstName());
    authorityList.add(authority);

    //Assigning relationships
    userCredentials.setAuthority(authorityList);
    userCredentials.setUser(user);
    user.setUserCredentials(userCredentials);
}
```

# User Module – AOP

```
@After("addAddress() && addArgs(user)")
public void afterExecution(JoinPoint joinPoint, User user){
    EmailConfirmationToken confirmationToken = new EmailConfirmationToken(user);

    emailService.save(confirmationToken);

    SimpleMailMessage mailMessage = new SimpleMailMessage();
    mailMessage.setTo(user.getEmail());
    mailMessage.setSubject("Complete Registration!");
    mailMessage.setFrom("{spring.mail.username}");
    mailMessage.setText("To confirm your account, please click here : "+
        "http://localhost:8080/confirm-account?token="+confirmationToken.getConfirmationToken());

    emailService.sendEmail(mailMessage);
}
```

# User Module – Spring Security

```
@Autowired
DataSource dataSource;

@Autowired
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.jdbcAuthentication()
        .dataSource(dataSource)
        .usersByUsernameQuery("select user,password,enabled from authentication where user = ?")
        .authoritiesByUsernameQuery("select username, authority from authorities where username = ?");
}

@Override
protected void configure(HttpSecurity httpSecurity) throws Exception {
    httpSecurity.authorizeRequests()
        .antMatchers(...antPatterns: "/myterms").hasAuthority("USER")
        .antMatchers(...antPatterns: "/").permitAll()
        .and().formLogin();
}

@Bean
public BCryptPasswordEncoder getPasswordEncoder() { return new BCryptPasswordEncoder(); }
```

# Category Module - Controller

```
@Controller
@RequestMapping(value = "/category")
public class CategoryController {
    @Autowired
    CategoryService categoryService;

    @RequestMapping(value = "")
    public String getCategoryList(Model model){
        System.out.println("===== category list =====");
        List<Category> categoryList = categoryService.findAll();
        for (Category t: categoryList) {
            System.out.println(t.getName());
        }
        model.addAttribute("categoryList", categoryList);
        return "categoryList";
    }
}
```

```
@RequestMapping(value =("/{category}", method = RequestMethod.GET)
public String terms(@PathVariable("category") String categoryName, Model model){
    System.out.println(categoryName);
    List<Category> categoryList = categoryService.findAll();
    for(Category cat : categoryList){
        if(cat.getName().equals(categoryName)){
            model.addAttribute("termList", cat.getTerms());
            return "categoryTerms";
        }
    }
    return "";
}
}
```

# Category Module – Service Implement

```
@Service
@Transactional
public class CategoryServiceImpl implements CategoryService {
    @Autowired
    private CategoryDao categoryDao;

    public List<Category> findAll(){
        return categoryDao.findAll();
    }
    public Category findByName(String name){
        return categoryDao.findByName(name);
    }
    public void addCategory(Category category){
        categoryDao.save(category);
    }
}
```

# Category Module – DAO implement

```
@Repository
public class CategoryDaoImpl extends GenericDaoImpl<Category> implements CategoryDao {
    public CategoryDaoImpl(){
        super.setDaoType(Category.class);
    }

    public List<Category> findAll(){
        return super.findAll();
    }

    public void save(Category t){
        super.save(t);
    }
}
```

# Category Module - Model

```
@Entity
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    // declaration of category name
    private String name;

    // declaration of category description
    private String description;

    @Fetch(FetchMode.SUBSELECT)
    @OneToMany(cascade = CascadeType.ALL)
    @JoinColumn(name = "category_id")
    private List<Term> terms = new ArrayList<Term>();
}
```

# Term Module

```
@Controller
@RequestMapping(value = "/term")
public class TermController {

    @Autowired
    TermService termService;

    @Autowired
    UserService userService;

    @Autowired
    CategoryService categoryService;

    @RequestMapping(value = {"/", "/list"})
    public String getTermList(Model model) {
        List<Term> termList = termService.findAll();
        model.addAttribute(s: "termList", termList);
        return "termList";
    }

    @RequestMapping(value= "/search")
    public String searchTerm(@RequestParam String input, Model model){
        System.out.println(input);
        List<Term> list = termService.findAll().stream().filter((term -> term.getWord().equals(input)));
        model.addAttribute(s: "termList", list);
        return "termList";
    }

    @RequestMapping(value = "/add")
    public String showAddView(Model model, @ModelAttribute("term") Term term){
        model.addAttribute(s: "categoryList", categoryService.findAll());
        return "addterm";
    }
}
```



# RestController

```
@org.springframework.web.bind.annotation.RestController
public class RestController {
    @Autowired
    TermDao repository;

    @GetMapping("/api")
    public @ResponseBody List<Term> getTermList() { return repository.findAll(); }

    @GetMapping("/api/find")
    public @ResponseBody List<Term> getTermByWord(@RequestParam(value = "word") String word){
        System.out.println(word);
        List<Term> list = new ArrayList<>();
        for(Term t : repository.findAll()){
            if(t.getWord().equals(word)) list.add(t);
        }
        return list;
    }

    @PostMapping("/api")
    @ResponseStatus(value = HttpStatus.NO_CONTENT)
    public void addTerm(@RequestBody Term term) {
        System.out.println("/api/add");
        repository.save(term);
    }
}
```