« Simple CRUD Web Application with JSF 2.1, PrimeFaces 3.5, Maven and JPA
Part 2 – Securing Web Application GlassFish V3 – JAAS (Authentication and Authorization) »

Part 1 – Creating Entity Classes With POJO + Annotation 16

21 Sep 2012 | Java EE

Most of the data our web project manipulates will be stored in our database. To persist state of our objects within Java, we will use Java Persistence API and its ORM mechanism to map objects to the data saved in our database.

We have a base class for entities. That super class will have the common methods. We annotated it with @MappedSuperclass so when we extend it JPA will not complain about not finding an @Id annotation.

```
1  package com.nz.simplecrud.entity;
2
3  import java.io.Serializable;
4  import javax.persistence.GeneratedValue;
5  import javax.persistence.GenerationType;
6  import javax.persistence.Id;
7  import javax.persistence.MappedSuperclass;
8
9  @MappedSuperclass
10 public abstract class BaseEntity implements Serializable {
11
12     private static final long serialVersionUID = 1L;
13     @Id
14     @GeneratedValue(strategy = GenerationType.IDENTITY)
15     private Integer id;
16
17     public Integer getId() {
18         return id;
19     }
20
21     public void setId(Integer id) {
22         this.id = id;
23     }
24
25     @Override
26     public int hashCode() {
27         int hash = 0;
28         hash += (id != null ? id.hashCode() : 0);
29         return hash;
30     }
31
```

Blogroll

- Arquillian testing tool for Java EE
- PrimeFaces JSF Component Suite

Recent Posts

- Glassfish4 and JSF 2.2 ViewScope support (JSF 2.2, CDI, JPA 2 (ORM) / EclipseLink, JAAS , MySQL, example CRUD web application)
- My Book Review – Instant PrimeFaces Starter- Packt Publishing
- Reviewing Instant PrimeFaces Starter- Packt Publishing
- My Book Review – PrimeFaces Cookbook by Oleg Varaksin and Mert Çalışkan
- Reviewing PrimeFaces Cookbook – Packt Publishing – Oleg Varaksin, Mert Çalışkan

Recent Comments

- simtay on Part 4 – Lazy Loading and Pagination with JSF 2.1 / Primefaces 3.4.2 DataTable and Form validation
- simtay on Simple CRUD Web Application with JSF 2.1, PrimeFaces 3.5, Maven and JPA
- Mohamed on Part 4 – Lazy Loading and Pagination with JSF 2.1 / Primefaces 3.4.2 DataTable and Form validation
- dukgo on Simple CRUD Web Application with JSF 2.1, PrimeFaces 3.5, Maven and JPA
- simtay on Simple CRUD Web Application with JSF 2.1, PrimeFaces 3.5, Maven and JPA

Archives

- November 2013
- August 2013
- April 2013
- March 2013
- January 2013
- September 2012

Categories

```
    @Override
    public boolean equals(Object obj) {

        if (obj == null) {
            return false;
        } else if (!(obj instanceof BaseEntity)) {
            return false;
        } else if (((BaseEntity) obj).id.equals(this.id)) {
            return true;
        } else {
            return false;
        }
    }

    @Override
    public String toString() {
        return "entity." + this.getClass() + "[ id=" + id + " ] ";
    }
}
```

In our web application each user will have an address and User entity has userid as a primary id , and it will also have name, surname, e-mail, password and address.

User Entity class will look like this:
Note that we don't need to add an Id because the base class already has it.

```
package com.nz.simplecrud.entity;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.List;
import javax.persistence.*;


@Entity
public class User extends BaseEntity implements Serializable {

    @Column(nullable = false, length = 50)
    private String username;

    @Column(length = 50)
    private String firstname;

    @Column(length = 50)
    private String lastname;

    @Column(length = 50)
    private String email;

    @Column(length = 64)
    private String password;

    @OneToOne(cascade = {CascadeType.ALL})
    private Address address;

    public User() {
        address = new Address();
    }

    public String getUsername() {
        return this.username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getFirstname() {
        return this.firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return this.lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getEmail() {
        return this.email;
    }

    public void setEmail(String email) {
        this.email = email;
```

› Java EE

Meta

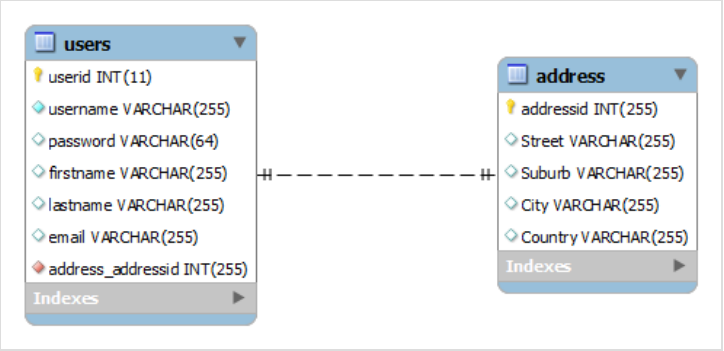
› Log in
› Entries RSS
› Comments RSS
› WordPress.org

```
        this.email = email;
    }

    public String getPassword() {
        return this.password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public Address getAddress() {
        return this.address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }
}
```

I prefer using annotations and I want to briefly mention purposes of these annotations.
First, we have to mark our POJO as persistent entity. @Entity annotation is not optional and has to be declared at the class level.
Each entity has to have at least one field that will serve as a primary key and will be annotated by @Id.
Our objects id will be a generated id, so we need to add sequencing to our entity. That will be accomplished by adding @GenertedValue annotation and its strategy needs to be set to IDENTITY.
With EJB V3 Configuration by Exception is introduced and by default JPA assumes that an entity's name and entity's field names correspond to a database table and this table's columns with the same names. By adding @Column I am overriding this behaviour.
@OneToOne association is defined between user and address because an address object will only be associated with a single user object and user defines the unidirectional relationship.
When a User is persisted, if there is any Address association with this User, then that Address will be persisted too. When the User is removed from database, the Address record will be removed too.

```
package com.nz.simplecrud.entity;

import java.io.Serializable;
import javax.persistence.*;

@Entity
public class Address extends BaseEntity implements Serializable {

    @Column(length = 50)
    private String street;

    @Column(length = 50)
    private String suburb;

    @Column(length = 50)
    private String city;

    @Column(length = 50)
    private String country;

    public Address() {
    }


    public String getStreet() {
        return this.street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getSuburb() {
        return this.suburb;
    }

    public void setSuburb(String suburb) {
        this.suburb = suburb;
    }

    public String getCity() {
        return this.city;
    }

    public void setCity(String city) {
        this.city = city;
    }

```

```
48      public String getCountry() {
49          return this.country;
50      }
51
52      public void setCountry(String country) {
53          this.country = country;
54      }
55  }
```



Download the source code here
Please try live demo (Username: Admin, Password:1234)

16 thoughts on "Part 1 – Creating Entity Classes With POJO + Annotation"

Reply

Oct 6, 2012 4:25 pm

Vinh Truong

Great posts! This's what I'm looking for my project.
Thanks for very nice works!

Reply

Oct 7, 2012 9:28 am

simtay

Hi Vinh,
I am glad you like it. Thanks for your comment.
Cheers
Emre

Reply

Jan 16, 2013 5:25 am

Asme

hi, I've just starting to read your tutorial after week of search about EJBs... I really hope I will undestand it this time. Will tell you later about my progression. wish me good luck.

Reply

Jan 16, 2013 7:32 am

simtay

Good luck and if you have any question feel free to ask here or email me. Cheers

Reply

Feb 16, 2013 5:11 pm

nel

Could you share the database script with me...?
Send it to nelson_mnc44@hotmail.com, please...
thanks for your help!

Reply

Feb 17, 2013 11:58 am

simtay

I have e-mailed you the script. It is also on the website look at

part 2: http://www.simtay.com/part-2-securing-web-application-glassfish-v3-jaas-authentication-and-authorization/ Cheers

Reply

Apr 30, 2013 9:42 pm

royjavelosa

Hi simtay nice tuts very rare will you find this kind of high quality and updated jsf2 tutorial.
I have a question what will I change in the code so that when I click the menu only the middle part will refresh and not the whole page?
if possible can you mail me the updated code
Overall great tuts thanks!

Reply

May 2, 2013 2:15 pm

admin

Hopefully when I have time for that

Reply

Jun 10, 2013 9:12 pm

Anouar

Hi simtay, thanks for this great tutorial
I like your idea to use a "BaseEntity" that will have the common methods. but there is an inconvenient, we lost the possibility to use the inheritance in other context ,specially when it is required in the business logic, (with java we can inherit from just one class, or implement interfaces).
what do you think ?

Reply

Jul 5, 2013 2:20 pm

simtay

You can remove base entity class and add id for each entity class, voila!

Reply

Jul 5, 2013 2:21 pm

simtay

You can remove base entity class and add @ID for each entity class, voila!

Reply

Jun 21, 2013 12:23 pm

Raj

I logged with demo user and open another browser window and hit refresh window it auto logged me with user demo. Is it possible that I can open two separate sessions on same machine so that I can work as different accounts.

Reply

Jul 5, 2013 2:15 pm

simtay

Use a different browser for another session or follow Timmorn's suggestion.

Reply

Aug 14, 2013 12:18 am

Timmorn

Use Chrome and create different accounts. Every account can have its own session then:
https://support.google.com/chrome/answer/2364824?hl=en

Sep 16, 2013 6:59 pm

Halil Karakose

Reply

Hi simtay
I try to create entities by myself via uml diagram. User entity contains a relationship with Role(s) entity. But Role entity does not exist in the data model uml diagram. This is not something critical. It will be better if you update the uml diagram.

Thanks

Reply

Sep 17, 2013 11:36 am

simtay

Sorry Roles aren't meant to be in the code, so removed it. It will be explained in Part 2. Cheers...

Leave a Reply

Author (required)

Email (will not be published)(required)

Website

Just to prove you are a human, please answer the following math challenge either out loud or typing into the text box

× seven = sixty three

b i link b-quote code close tags

Post Comment

« Simple CRUD Web Application with JSF 2.1, PrimeFaces 3.5, Maven and JPA
Part 2 – Securing Web Application GlassFish V3 – JAAS (Authentication and Authorization) »