Linguagem de Programação Orientada a Objetos I

Flask

Prof. Tales Bitelo Viegas
https://fb.com/ProfessorTalesViegas

Na prática 0.0.12 – Gravatar

- O serviço Gravatar é uma maneira simples de incluir avatares de usuário
- Através de um hash/md5 do e-mail do usuário, podemos utilizar a imagem que o usuário colocou no serviço através da URL:
  - http://www.gravatar.com/avatar/HASH
  - https://secure.gravatar.com/avatar/HASH

Gravatar

- Encapsulamos a geração da URL do avatar no modelo User, conforme abaixo

```
    def __init__(self, **kwargs):
        super(User, self).__init__(**kwargs)
        if (self.email is not None) and (self.avatar_hash is None):
            self.avatar_hash = hashlib.md5(self.email.encode('utf-8')).hexdigest()
```

```
def gravatar(self, size=100, default='identicon', rating='g'):
    if request.is_secure:
        url = 'https://secure.gravatar.com/avatar'
    else:
        url = 'http://www.gravatar.com/avatar'
    hash = self.avatar_hash or hashlib.md5(self.email.encode('utf-8')).hexdigest()
    return "{url}/{hash}?s={size}&d={default}&r={rating}".format(url=url, hash=hash, size=size, default=default,
```

Gravatar

- Podemos requisitar a URL do usuário diretamente no template

```
{% block page_content %}
    <div class="page-header user-profile">
        <div class="user-avatar"><img src="{{ user.gravatar(128) }}" /></div>
        <h1>{{ user.username }}</h1>
    </div>
```

Na prática 0.0.13 – Rota do Perfil

- A classe ProfileForm define 3 campos editáveis do perfil
- O decorator login_required previne o acesso de usuários que não estão logados

```
class ProfileForm(Form):
    name = StringField('Name', validators=[Optional(), Length(1, 64)])
    location = StringField('Location', validators=[Optional(), Length(1, 64)])
    bio = TextAreaField('Bio')
    submit = SubmitField('Submit')
```

```
@talks.route('/profile', methods=['GET', 'POST'])
@login_required
def profile():
    form = ProfileForm()
    if form.validate_on_submit():
        current_user.name = form.name.data
        current_user.location = form.location.data
        current_user.bio = form.bio.data
        db.session.add(current_user._get_current_object())
        db.session.commit()
        flash('Your profile has been updated.')
        return redirect(url_for('talks.user', username=current_user.username))
    form.name.data = current_user.name
    form.location.data = current_user.location
    form.bio.data = current_user.bio
    return render_template('talks/profile.html', form=form)
```

Na prática 0.0.14 – Adicionando Talks

```
class Talk(db.Model):
    __tablename__ = 'talks'
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(128), nullable=False)
    description = db.Column(db.Text)
    slides = db.Column(db.Text())
    video = db.Column(db.Text())
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'))
    venue = db.Column(db.String(128))
    venue_url = db.Column(db.String(128))
    date = db.Column(db.DateTime())
```

Adicionando Talks

```
class TalkForm(Form):
    title = StringField('Title', validators=[DataRequired(), Length(1, 128)])
    description = TextAreaField('Description')
    slides = StringField('Slides Embed Code (450 pixels wide)')
    video = StringField('Video Embed Code (450 pixels wide)')
    venue = StringField('Venue',
                        validators=[DataRequired(), Length(1, 128)])
    venue_url = StringField('Venue URL',
                            validators=[Optional(), Length(1, 128), URL()])
    date = DateField('Date')
    submit = SubmitField('Submit')
```

v.0.0.14 – Database Queries

```
@talks.route('/')
def index():
    talk_list = Talk.query.order_by(Talk.date.desc()).all()
    return render_template('talks/index.html', talks=talk_list)

@talks.route('/user/<username>')
def user(username):
    user = User.query.filter_by(username=username).first_or_404()
    talk_list = user.talks.order_by(Talk.date.desc()).all()
    return render_template('talks/user.html', user=user, talks = talk_list)
```