UNICORN COLLEGE

Zaměstnanec (Jukin Tomáš)
Webové technologie (verze 2014) Zima 2016 ▸ 04 Domácí úkoly / práce

UCL Exercise Assignment
UCL-BT:WEB14A16W.CZ/XA03_PT

**WEB 3. Kontrolní bod - Backend /PT**

## Kontext

Semestrální práce z předmětu WEB má 3 části (dva kontrolní body a finální odevzdání). Ke každé části se váže jeden artefakt (jedno zadání).

1. V první části si ověříte svoji schopnost práce s HTML a CSS (někteří i JS). Vaším cílem bude vytvořit landing page pro váš produkt. Odevzdáním této části si rovněž zvolíte téma své semestrální práce.
2. Ve druhé části již budete pracovat v Ruby on Rails a vytvoříte tzv. modely a frontend vašeho produktu (pokud budete dělat například blog, tak v této části vytvoříte tu jeho část, kterou vidí vaši čtenáři). Poznámka: frontend má jiný (leč možno související) vzhled, tedy design než-li landing page!
3. Ve třetí a poslední části (to jest tato) pak vytvoříte tzv. backend pro váš produkt, tedy administrační rozhraní pomocí kterého můžete svůj produkt spravovat (tedy pro náš příklad blogu můžete v administraci přidávat a upravovat např. články, kategorie a tagy vašich článků na blogu).

Každá webová aplikace potřebuje backend - tedy tu část, kterou vidí pouze administrátoři a která umožňuje správu obsahu, odtud se této části také říká administrace. Proto ji bude mít i vaše semestrální práce. V tomto kontrolním bodu budeme opět používat framework Ruby on Rails.

**Upozornění!**

Tento úkol opravdu nenaprogramujete za jeden večer za 2 hodiny, tak ho nepodceňte :-). Přednášejícímu to zabralo 3 hodiny, protože 2 z nich psal dokumentaci a ví co dělá, tudíž ti nerychlejší z vás to budou mít za 3... Ale když se v tom zamotáte může to být třeba na

dva dny až na dva týdny (= doporučuji se proto nezamotávat a nebo začít včas a v případě zamotání se dorazit na seminář, kde vás cvičící rád rozmotá).

Hodně štěstí!

## Zadání

Úkolem tohoto kontrolního bodu je vytvoření backendu, neboli administrace pro vaši semestrální práci.

Bez ohledu na rozsah vaší práce je třeba vytvořit administraci, která bude mít správu všech modelů vaší aplikace (tyto máte vytvořeny z předchozího kontrolního bodu). Mezi modely musí být (již z minula) alespoň jedna 1:N a alespoň jedna M:N asociace.

Jaké modely budete spravovat záleží na tématu vaší semestrální práce.

Je třeba, aby vaše administrace pro každý model, který spravuje umožňovala:

- Zobrazovat seznam (akce index)
- Upravovat jej (akce edit a update)
- Zobrazovat jeho detail (akce show)
- Mazat jeho položky po jedné (akce delete)
- Přidávat nové položky (akce new a create)
- 1:N asociace může být spravována pouze z jedné strany (např. článek má kategorie, tak stačí, aby u článku šlo vybrat kategorii, nemusí jít u kategorie vybrat, které články ji mají)
- M:N asociace může být spravována pouze z jedné strany (např. článek má tagy, tak stačí, aby u článku šlo vybrat tagy, nemusí jít u tagu vybrat, které články ho mají)

U kontrolních bodů prosím používejte POUZE sqlite databázi! Jiné databáze s dovodu snadnosti kontroly nebudou akceptovány.

Dále musí u vaší administrace jít: **(tohleto jsou věci, které vám scaffold už neudělá)**

- 1:N asociace musí jít spravovat pomocí select boxu (např. u článku si selectboxem vzberu jeho kategorii)
- M:N asociace musí jít spravovat pomocí bandy tagů (např. u článku si zaškrtám, jaké tagy chci aby měl)

- U formulářů pro výběr asociací (např. ty kategorie a tagy u článku) musí být odkaz na přidání nové položky příslušného typu (např. u políčka pro výběr kategorie v článku bude odkaz na přidání nové kategorie) - *pro účely tohoto kontrolního bodu stačí, když tento odkaz normálně přepne stránku na přidání příslušné položky - tedy na přidávací formulář, bylo by možná slušné, kdyby to udělal v novém tabu (= přidat attribut target="_blank" do a tagu) - ano pak je stejně na původní stránce třeba dát refresh, což vymaže formulář - to nám teď nevadí, řešení tohoto problému je lehce netriviální a proto ho po vás teď nechci ;) - kdy chce přijmout výzvu, může tento problém vyřešit jako bonusový úkol :-)*
- Musí existovat jakýsi "Dashboard", kde uvidím odkazy na všechny "administrovatelné věci". Dashboard = odkazy na administrace jednotlivých věcí. Stačí formou nav tagu, ve kterém jsou ul-li tagy s odkazy, fajnšmekři mohou udělat nějaké dlaždice jako bonus
- Na dashboardu by mělo být vidět, kolik kterých věcí v systému mám
- Dashboard by musí být homepage tohoto kontrolního bodu (tj. měl by na něj vést root path)
- Vaše administrace musí být RESTful (toto bude defaultně, pokud neuděláte něco špatně) - viz. přednášky co byly
- V detailech položek musí být vidět všechny jejich attributy! - včetně created_at a updated_at a včetně associací
- V seznamových výpisech musí být vidět všechny položky včetně relací a created_at a updated_at s vyjímkou toho, že tam není třeba vidět ty relace, ke kterým položka nemá formuláře - a logicky, když mám např kategorie u článků, tak chci vidět v seznamu název té kategorie a nebo hash, co tam ruby dává defaultně!

Také je třeba vyřešit:

- Vzhled administrace - netřeba žádný zázrak, ale administrtace mu být použitelná (varianta s frameworky předpokládá plné nasazení Twitter Bootstrap - např. v kombinaci se simple_form)
- Přihlašování (stačí základní HTTP Basic Auth)
- Validace
- Výchozí hodnoty

Níže jsou jednotlivé úkoly více rozepsány. Ale v podstatě je třeba mít nyní na paměti, že bychom tímto kontrolním bodem měli "dokončit administraci", abychom získali kompletní aplikaci.

**Vzhled administrace**

Vaším úkolem je administraci nastylovat. Její vzhled nemusí ladit se vzhledem landing page z prvního úkolu, ani vzhled frontendu ze druhého úkolu. Nemusí to být umělecké dílo, ale musí to **být použitelné**. To znamená, že ti z vás, kdo nepoužíváte frameworky si dejte pozor zejména na kontrast barvy písma a pozadí (kupodivu černá na černé není vidět, ale ani šedivá na černé není vidět atd. - po pravdě nalezení správného barevného schématu je docela věda, grafici to umí z hlavy, ale co my programátoři? Inu my si umíme

naprogramovat pomocníky :-) - jeden povedený, který vřele doporučuji všem, kdo nemáte, nebo nejste grafiky a neděláte verzi s frameworky, které to řeší za vás, je k dispozici zde: http://colorschemedesigner.com/ - přepněte si v něm výběr na komplementární barvy).

Vzhled udělejte pomocí CSS souborů, které do administrace připojíte (jako tomu bylo u prvního i druhého kontrolního bodu). Ne že CSS budete psát přímo do HTML!

Jinak jedna věc pro variantu s frameworky - ne že použijete výchozí barevné schéma vašeho frameworku! Buďto si k němu vyrobte vlastní schéma a nebo použijte nějaké už existující, ale ne to výchozí!

**Validace**

V tomto kontrolním bodě je nutné také všem vašim modelům přidat validaci. To znamená že:

1. Všechny attributy, které jsou povinné musí být povinné (validujeme pomocí validates_presence) - jak se pozná, že je attribut povinný? Tak že když je nil, nebo není zadaný, tak aplikace začne padat. Jinými slovy, až bude cvičící vaši práci kontrolovat, tak zkusí pro každou z vašich položek vytvořit nový model a vyplní jen povinné položky (attributy). Když pak aplikace spadne, body dolů. Tedy vaše aplikace musí být udělána tak, že musí počítat s tím že poviné attributy uživatel nezadá - musí se u jejich výpisu u nich například dělat IF podmínka aby to nespadlo atd.
2. Pokud je nějaký attribut číslo, tak musí mít na číslo validaci.
3. Pokud je nějaký attribut email, nebo něco podobného (URL, atd.) tak na to musí mít validaci
4. Boolean attributy musí mít validace (inclusion in true/false)
5. Pokud má model asociace, měl by zvalidovat asociované modely (pozor na zacyklení)
6. Pokud je model enumerace (výčet hodnot, které jsou v kódu "natvrdo"), pak je třeba jej zvalidovat pomocí validate_inclusion.

***POZOR!***

V každém formuláři musí být poznat, který attribut (pole) je povinný a který není! Pokud tomu tak nebude, body dolů.

Všechny dostupné druhy validací a jak je udělat najdete zde:
http://guides.rubyonrails.org/active_record_validations_callbacks.html#validations-overview

**Výchozí hodnoty**

Do alespoň jednoho modelu přidejte alespoň k jednomu attributu výchozí hodnotu. Pokud vám to dává smysl můžete to udělat u více modelů a více attributů, ale u jednoho to být musí.

Existuje více způsobů jak přidat výchozí hodnotu. Lze to pomocí migrace, což je OK, ale je problém když ji chcete změnit - pak je třeba udělat migraci novou. Tudíž to není příliš flexibilní. V modelu se to nastavuje problematicky. Proto na to existuje gem.

Pro kterou z variant se rozhodnete, zda migraci, v modelu, nebo gem je na vás. Já doporučuji gem.

Jak na výchozí hodnoty v Rails: http://stackoverflow.com/questions/1550688/how-do-i-create-a-default-value-for-attributes-in-rails-activerecords-model

Gem pro výchozí hodnoty: https://github.com/FooBarWidget/default_value_for

Jak tuto administraci uděláte - zda napřed použijete scaffold a do něj to pak dopíšete, nebo zda to napíšete "z ničeho" bez scaffold generátoru, jen s použitím dílčích generátorů pro modely, kontroléry a migrace je na vás. Obě cesty vedou k cíli.

**Přihlašování do administrace**

Administrace by neměla být dostupná všem a proto je nutné do ní implementovat alespoň základní formu autentizace.

Ta je realizována pomocí jednoho jména a hesla, která jsou natvrdo napsána do kódu. Říká se jí HTTP Basic Auth.

Vaším úkolem je do aplikace takovouto autorizaci přidat. Návod na to, jak to udělat najdete tady: http://guides.rubyonrails.org/action_controller_overview.html#http-basic-authentication

Pokud děláte bonusové úkoly, a integrovali jste, nebo chcete v tomto úkolu integrovat, bonusovou autentizaci pomocí Devise pak nemusíte tento krok dělat. Více viz. bonusové úkoly.

**Tipy:**

- upravujte stejnmou rails aplikaci z minulého kontrolního bodu, nevytvářejte novou
- doporučujeme použít gem simple_form (není však nutnou podmínkou, pouze usnadní práci)
- vyhněte se použití rout match, místo toho používejte collection a member v resource, případně zanořené resource (viz. guides stránka "Rails routing from the outside in").
- migrace nikdy nepište sami, VŽDY je generujte - už kvůli názvu souboru (ano, může se vám stát že to nevygeneruje obsah té migrace, to nevadí, pak ho musíte dopsat - pokud to jde, chcete použít jen metodu change místo up & down)

- migrace nikdy neupravujte po spuštění rake db:migrate. Raději udělejte migraci novou. Pokud si nejste jisti zda ji upravit můžete nebo ne, tak je neupravujte nikdy.
- **NEPODCEŇTE TENTO KONTROLNÍ BOD! Ze všech kontrolních bodů je tento nejnáchylnější k tomu, aby se časově protáhl...**
- pokud někde chcete použít enumeraci - to je, když máte např. nějaký model a ten má u něčeho napevno jeden z několika stavů (více než dvou, u dvou lze místo toho použít boolean attribut) a tyto stavy máte v aplikaci napevno (tedy jejich seznam nelze měnit v administraci), pak je to přesně tzv výčet, neboli enum. Tak triviální použití enumerace je, že máte attribut typu string a v modelu validate_inclusion v poli. A ve formuláři máte select helper a dáváte mu pole těch možností. Takto to určitě lze udělat, ale mnohem pohodlnější je enumerize gem. Doporučujeme.
- controllery ve vaší aplikaci mají fungovat jen jako jakési lepidlo, tedy mít v sobě co nejméně kódu, zato model ho bude mít hodně. JE doporučení místo volání Rails Active Record API přímo v controlleru ho místo toho volat pomocí metody modelu. Tedy např. místo toho abych v akci kontroléru pro nalezení a zobrazení příspěvku podle username autora udělal Post.where(:username => params[:username]).first, tak je lepší naučit model Post statickou metodu self.find_by_username(username), ve které by byl ten kód z kontroléru - BTW pro takto triviální věci to netřeba protože zrovna toto umí Rails automaticky, ale pro dlouhá a složitá volání je určitě lepší to mít v modelu.

**Poznámky:**

- Pro tento kontrolní bod NELZE používat gemy jako jsou rails_admin, active_admin, atd. - **použití těchto gemů = 0 bodů, nebo nutnost přepracovat úkol!**
- Je NUTNÉ používat Rails verze 4 a vyšší.
- Je NUTNÉ dodržovat téma vaší semestrální práce!

V prvním kontrolním bodě jste se rohodli pro jednu ze dvou variant své semestrální práce. Níže jsou další podmínky zadání dle této varianty. Až po tento řádek se zadání týkalo obou variant.

Omezení pro variantu ze zelené louky

- Vaše administrace musí být validní HTML, dle validátoru z minulého kontrolního bodu
- Na CSS vaší administrace se vztahují všechna omezení pro variantu ze zelené louky z 1. kontrolního bodu, kromě povinných tagů.

Omezení pro variantu s frameworky

- Vaše administrace musí být validní HTML5 včetně správného použití HTML5 formulářových prvků

- Vaše administrace musí být responsivní
- Na HTML, CSS a JS vaší administrace se vztahují všechna omezení z 1. kontrolního bodu (ve smyslu musí být validní, atd. Není zde omezené co se týče povinných tagů, je zda však omezení ohledně toho co je vhodné použít - např. tag nav na navigaci, atd.)
- Co se týče vzhledu, tak stejně jako u landing page nemůžete použít výchozí vzhled vašeho frameworku! Pro twitter bootstrap existuje jednak možnost jeho customizace a nebo celá řada už existujících barevných témat. Tak koukejte nějaké najít a použít! Kdo odevzdá výchozí modrý bootstrap, dostane bodový postih!

**Bonusové úkoly:**

Pokud máte pocit, že je tento kontrolní bod příliš triviální, můžete dobrovolně zpracovat následující:

- zkuste do administrace místo HTTP Basic Auth přidat autentizaci pomocí gemu devise (pokud používáte devise, tu HTTP tam mít nemusíte)
- zkuste do administrace přidat autorizaci pomocí gemu cancancan
- přidejte ke všem modelům do výpisu položek kromě akcí pro úpravu a smazání i akci "duplikovat", která umožní vložit novou položku jako kopii nějaké už existující položky - v podstatě "předvyplní formulář pro novou položku všemi daty té zvoléné položky **kromě ID**"
- přidat do aplikace podporu pro nahrávání obrázků (zajímá vás gem carrierwave - carrierwave je lepší než paperclip a dragonfly)
- přidat do aplikace podporu pro verzování obsahu - takový malý "time machine" (zajímá vás gem papertrail)
- přidat do aplikace WYSIWYG editor - chcete použít tinyMCE, nechá se k němu najít i "tinyMCE twitter bootstrap skin" (googlíte gem tinyMCE rails)
- přidat do aplikace podporu pro Markdown WYSIWYM (zajímá vás gem redcarpet)
- přidat do aplikace podporu pro takovou geeky textareu - dobře se do ní píše HTML (jmenuje se to codemirror a zajímá vás gem codemirror-rails)

## Řešení

Odeslat řešení domácího úkolu

Vaše řešení MUSÍ jít spustit "z ničeho" následujícím postupem (předpokládejte, že cvičícímu už funguje ruby a bundler). Tento postup bude kontrolující cvičící aplikovat a pokud mu neprojde, tak vaše práce bude ohodnocena 0 body!

Postup kontroly:

1. Stáhnutí .zip a jeho rozbalení, výstupem tohoto rozbalení by měl být 1 adresář
2. Přepnutí do tohoto adresáře
3. bundle install
4. rake db:migrate
5. rake db:seed
6. rails server
7. Pak se cvičící podívá na 0.0.0.0:3000, kde by mělo něco být (windowsáři se dívají na 127.0.0.1:3000)

Ve vlastním zájmu si, se svým zipem připraveným k odevzdání, tento postup vyzkoušejte provést! A až teprve potom co vám to PROJDE své řešení odevzdávejte!

I když úlohu odevzdáte, je možné za ni získat 0 bodů, toto nastane když:

- A) Obecné podmínky - 0 bodů máte když:
  - Odevzdáte méně než 50% řešení
  - Zjistíme, že jste úkol opsali (0 bodů dostane ten kdo řešení opsal i ten kdo řešení poskytl) - ANO, děláme diff - http://cs.wikipedia.org/wiki/Diff
  - Odevzdáte úkol po DEADLINE, 1 minuta po DEADLINE je také po DEADLINE
  - Odevzdáte úkol ve špatném formátu, se špatnou přílohou, atd. sice včas, ale logicky nebude akceptován. A pak nestihnete odevzdat opravenou verzi úkolu všas (do deadline). Toto NENÍ chyba cvičícího, že vám řešení "nestihl" opravit, ale VAŠE neboť jste nezvládli řešení odevzdat ve správném formátu napoprvé, nebo včas navzdory tomu, že je zde detailně popsán.

- B) Specifické podmínky tomuto předmětu či úkolu - 0 bodů máte když:
  - Zjistíme, že jste úkol opsali stylem copy paste z jiné webové stránky (zjistit toto je ještě snažší než diff)
  - Použijete tag <marquee>
  - Použijete font Comic Sans MS

- Odevzdáte aplikaci, u které selže bundle install, rake db:migrate a nebo se aplikace nespustí (selže rails server).

**Odevzdání JE BEZPODMÍNEČNĚ NUTNÉ provést přes systém UU** (Unicorn Universe), a to NÁSLEDOVNĚ:

Během realizace tohoto úkolu by měl vzniknout adresář s Ruby on Rails aplikací, adresář pojmenujete:

ucl_web_03_novak_jan

ten zazipujete a vznikne vám jeden .zip (zazipujete CELÝ ADRESÁŘ, ne jen jeno obsah!). Tento zip pojmenujete "ucl_web_03_novak_jan.zip". Pokud se soubor jmenuje jinak, přejmenujete ho.

Místo novak_jan je pochopitelně VAŠE jméno BEZ diakritiky. ANO, ty divné čáry v tom názvu NEJSOU mezery, ale znaky podtržítka - "_"

Tento SPRÁVNĚ POJMENOVANÝ .zip odevzdáte do UU jako přílohu řešení. A POTÉ napíšete do POPISU řešení na UU:

Odevzdávám ucl_web_03_novak_jan.zip

Místo novak_jan je pochopitelně VAŠE jméno BEZ diakritiky.

Tedy správné odevzdání = správně pojmenovaná příloha + správně napsaný text popisu. Pokud chcete cvičítímu něco vzkázat, připíšete to do popisu úkolu, za "Odevzdávám ucl_web_03_novak_jan.zip", přičemž mezi "Odevzdávám ucl_web_03_novak_jan.zip" a vaším textem bude jeden prázdný řádek.

**Pokud to uděláte jinak, VAŠE ŘEŠENÍ NEBUDE AKCEPTOVÁNO a budete to muset udělat znovu.**

Pokud máte k úkolu, jeho zpracování, či odevzdání nějaké dotazy, kontaktujte VČAS, ne naposlední chvíli, svého cvičícího komunikačním kanálem, který preferuje a který vám pro tyto účely definoval.

# Zdroje

- Jak na HTTP Basic Autentizaci v Rails - http://guides.rubyonrails.org/action_controller_overview.html#http-basic-authentication
- Jak na validace v Rails - http://guides.rubyonrails.org/active_record_validations_callbacks.html#validations-overview
- Jak na výchozí hodnoty v Rails - http://stackoverflow.com/questions/1550688/how-do-i-create-a-default-value-for-attributes-in-rails-activerecords-model
- Generátor barevných schémat - http://colorschemedesigner.com/
- Gem pro nastavení výchozích hodnot - https://github.com/FooBarWidget/default_value_for

Když nevíte, jak to implementovat, mrkněte na:

- Přednášky, přesně to, co se po vás chce, přednášející ukazoval na přednášce
- Web http://guides.rubyonrails.org - tam, JE vše co se po vás chce napsáno. Pokud ste to nepochopili z přednášek, pak si přečtěte tento web. Když si ho přečtete, tak to z něj nelze nepochopit. Pokud si ho přečtete a přesto nepochopíte, tak to znamená, že jste ho nepřečetli.