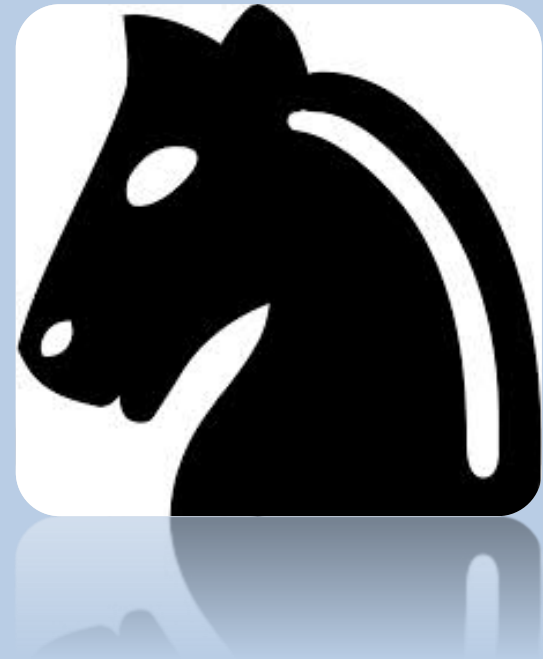




Ruby Business Programming

UE, 188.519, 2 h, 3 ECTS
Horst Eidenberger
Since 2018





FRONTMATTER

LECTURE
PROJECT
BACKMATTER

- Contents & Goals
- Lecture
- Lab Projects
- Assessment

Synopsis – What is to come?

- Implementing a simple business task with I/O and data management
- An opportunity for learning Ruby as well as the Rails framework (mostly in self-study) while gaining ECTS credits for it
- Gaining experience in a typical small-scale business programming scenario



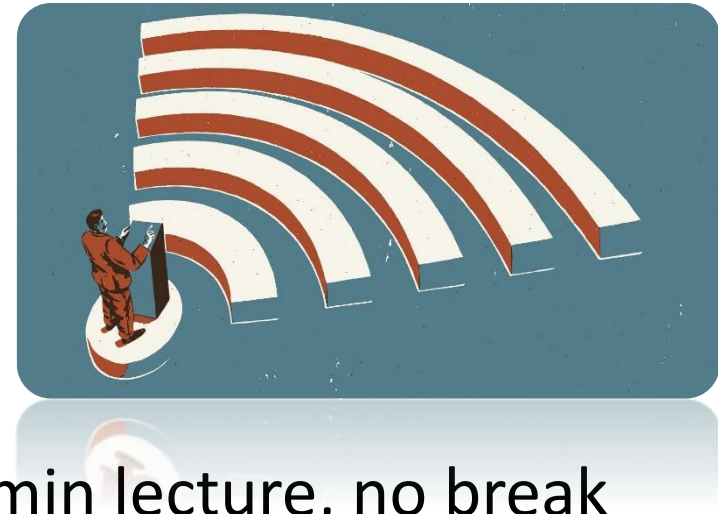
Educational Goals

1. Students learn to implement and **maintain a simple Ruby on Rails server**
2. Students learn to **implement Ruby on Rails applications** based on a given implementation process and defined requirements.
3. Students learn to **manage external data sources** with REST and JSON.
4. Students learn to **present business data** on mobile devices with optimized usability.



Lecture

- Initial lecture: today, here, now
- Contents:
 - Lab course topic
 - Ruby on Rails concepts
 - Administrative information
- Duration: 45min admin stuff + 45min lecture, no break
- No exam (lab course), but relevant for lab project
- **Attendance is mandatory!**
- **Registration in TISS is mandatory!**



Lab Projects

- Topic: Implement a mobile interface for TU Wien teaching data based on a Ruby on Rails server
- Aspects:
 - Search interfaces for people, courses, projects and theses
 - Presentation of results in lists and detail pages
 - CRUD functions for personal lists
 - Optimized interface for mobile applications
- Students work in groups of two participants
- Supervisor acts as product owner
- Repeated submissions are possible



Assessment

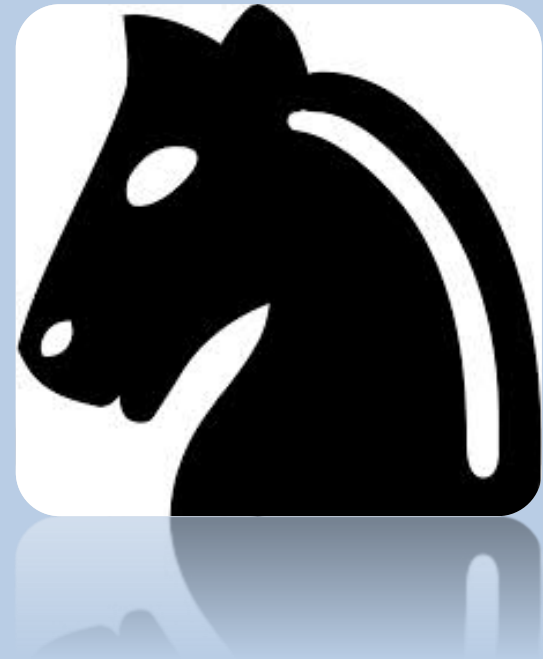
- Exclusively based on the lab project.
- Early hand-ins are possible.
- There is no exam but a personal meeting with all group members where the results and findings are discussed (“Abgabegespräch”).
- Active participation in the TUWEL forum is considered a bonus in the grading.
- Be brave and make your own decisions. Such behavior is almost ever rewarded.



Summary

1. Learn Ruby on Rails
2. Implement the given requirements (below)
3. Submit on time.
4. **Do not cheat!**





FRONTMATTER

LECTURE

PROJECT

BACKMATTER

- Idea & Concepts
- Conventions
- Assets
- Examples

Idea of Ruby on Rails

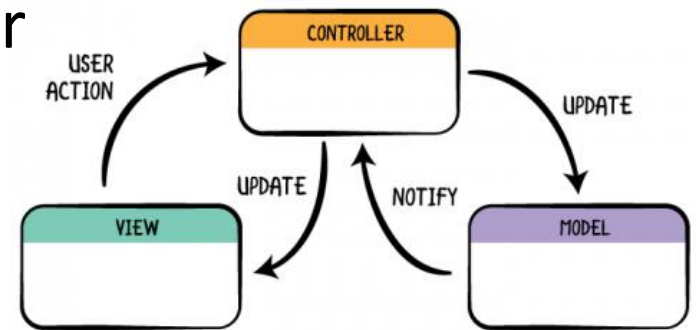
1. Targeted towards business applications
2. Strict implementation of event-based systems
3. Transparent persistence layer
4. Greedy implementation by conventions
5. Powerful wizards for creation of resources
6. Simple command line tools for DevOps

- ⇒ Rapid evolutionary prototyping
- ⇒ Tailor-made for agile processes
- ⇒ Solution-focused developer community



Course-Relevant Concepts

- RoR Application Design
 - MVC = Model, View, Controller
 - CRUD = Create, Read, Update, Delete
- RoR Programming
 - OOP = Object Oriented Programming
 - ORM = Object Relational Mapping
- Data Management
 - REST = Representational State Transfer



RoR Technologies and Assets

- Technologies

- ERB = Embedded Ruby: user interface classes
- ActiveRecord:
 - Object-relational mapper (does mapping, migrations, etc.)
 - “ActiveRecord::Base” implements CRUD functions of model classes
- “rake”: Ruby make utility

- Assets

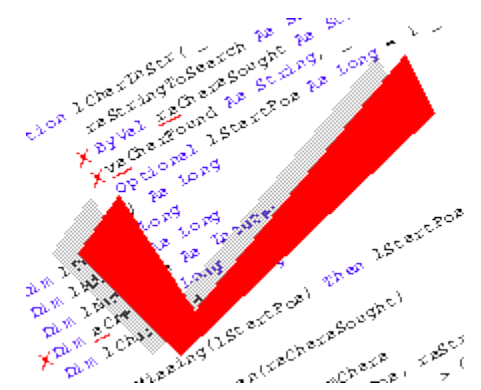
- “config/routes.rb”: Project configuration file:
 - Manages routes of views to controllers and of controllers to models
 - Manages entity relationships; e.g.:

```
resources :articles do          # 1:n relationship
  resources :comments
end
```
- Directories “app/{controllers, models, views}” contain applications resources, “db” contains database model



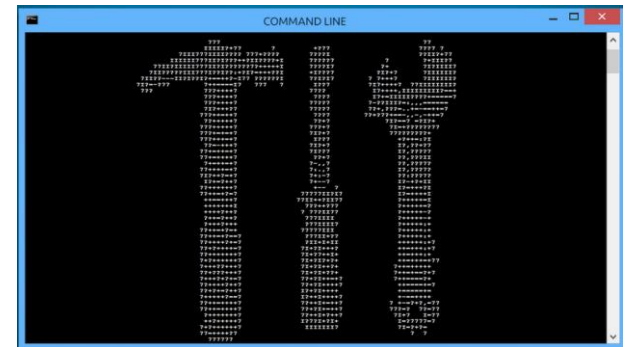
Some RoR Conventions

- Classes
 - Class names start with capital letters
 - Controller file names:
`lc(<controller_class_name>)+"_controller.rb"`
 - Models in Rails use a singular name, and their corresponding database tables use a plural name
- Methods
 - Implementation of a public method (no server restart required):
 1. View available methods by command "bin/rails routes"
 2. Create method in the controller class
 3. Create view template "<method>.html.erb" in directory "app/views"
 - Place CRUD actions in each controller in the following order: *index, show, new, edit, create, update, destroy*
- Variables:
 - Instance variables have prefix "@", parameters have prefix ":"
 - Array "params" contains parameters of the calling HTTP request



Important DevOps Commands

- Dev:
 - Create new project: “bin/rails new <project_name> “
 - Create new controller:
“bin/rails generate controller <name> [<action(s)>]”
 - Create new entity: “bin/rails generate model <entity_name> [attribute:type ...]”
- Ops:
 - Start web server: “bin/rails server”
 - Perform database migration:
“bin/rails db:migrate”
 - Show all routes from views to controllers: “bin/rails routes” (HTTP requests-> Ruby methods)



REST and JSON in Ruby

- JSON: „{ field : value }“ (alternative to complex XML formats)
- Packages („Gems“ in Ruby-Speak):
 - Reading from HTTP interfaces: `require 'net/http'`
 - Dealing with JSON formats: `require 'json'`
- Calls:
 - Get request to a REST interface (i.e. a HTTP request):

```
uri = URI('https://tiss.tuwien.ac.at/api/search/  
projectFullSearch/v1.0/projects?searchterm=nano')  
response = Net::HTTP.get(uri)
```
 - Parsing the response:

```
my_hash = JSON.parse(response)  
for i in 0..my_hash["results"].count-1  
  puts my_hash_1["results"][i]["title"]  
end
```
- See useful resources (backmatter) for links to libraries

Examples: Frequent Ruby Instructions

- Model classes:
 - Field validation in model classes:

```
validates :title, presence: true,  
length: { minimum: 5 }
```
 - Create entity relationships:
 - n:1 direction: `belongs_to :article`
 - 1:n direction: `has_many :comments`
- Controller classes:
 - Read parameters (with necessary and allowed parameters):

```
@article = Article.new(  
  params.require(:article)  
    .permit(:title, :text) )
```
 - Index data:

```
@article = Article.find(params[:id])
```
 - Save object through ORM:

```
@article.save
```
 - Webpage redirect:

```
redirect_to @article
```



Examples: Frequent ERB Markup

- Input forms:
 - Create web form:

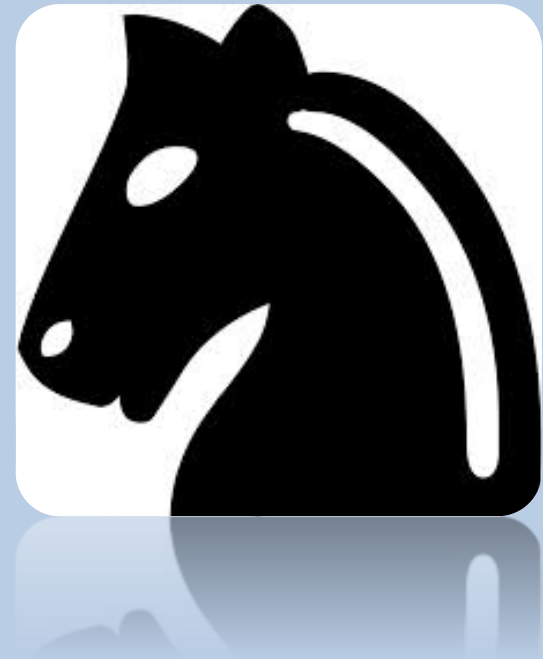
```
<%= form_with scope:
:article, url:
articles_path, local:
true do |form| %>
```
 - Define fields:
 - Label: `<%= form.label :title %>`
 - Input field: `<%= form.text_field :title %>`
 - Submit button: `<%= form.submit %>`
- Output data:
 - Write item: `<td><%= article.title %></td>`
 - Iterate over data: `<% @articles.each do |article| %>`
- References:
 - Web links: `<%= link_to 'Back', articles_path %>`
 - Validation: `<% if @article.errors.any? %>`



Summary

1. Ruby on Rails is a non-compromising approach to fast business programming.
2. Conventions require high fitness of programmers.
3. Operating is performed on the command line.





FRONTMATTER
LECTURE

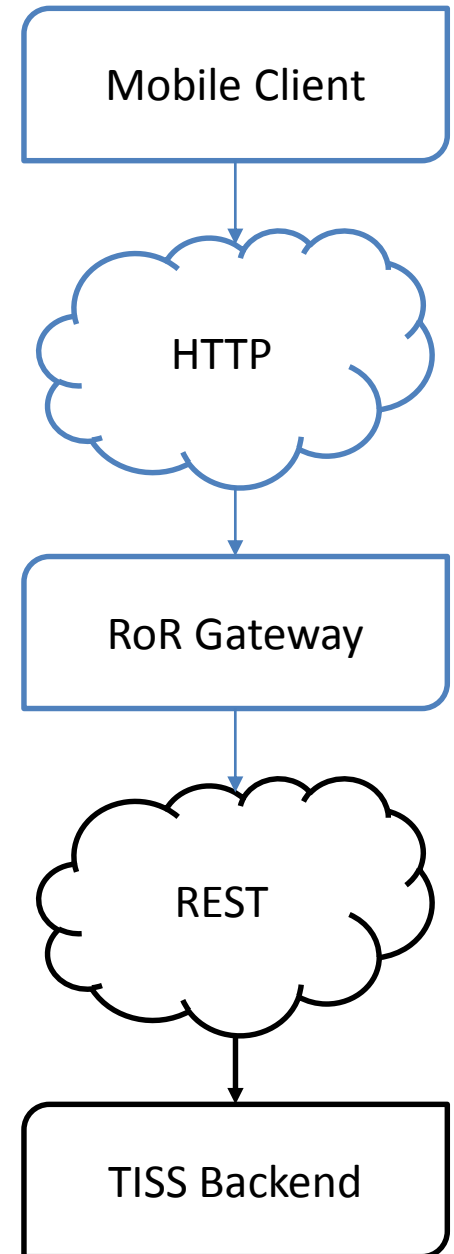
PROJECT

BACKMATTER

- System overview
- Requirements
- Approach
- REST Services Usage

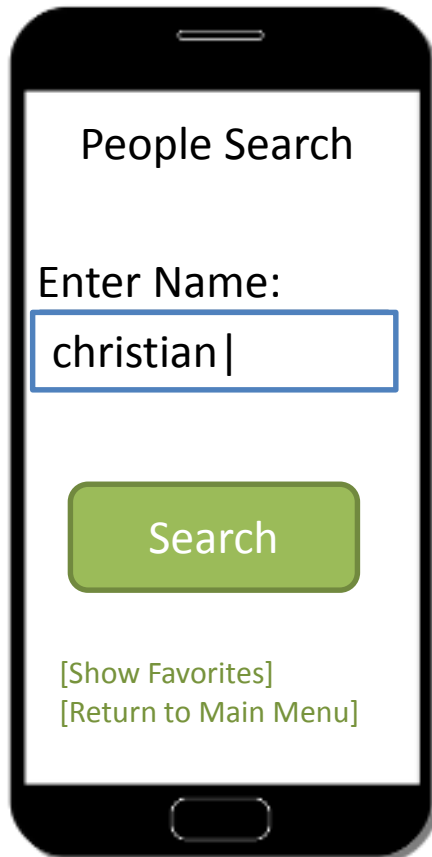
Overview

1. Implement a Ruby on Rails gateway for the TISS information systems (blue components)
2. Make use of existing public REST interfaces for data acquisition
3. Implement CRUD functions for favorites and all other requirements
4. Follow the regulations for hand-ins

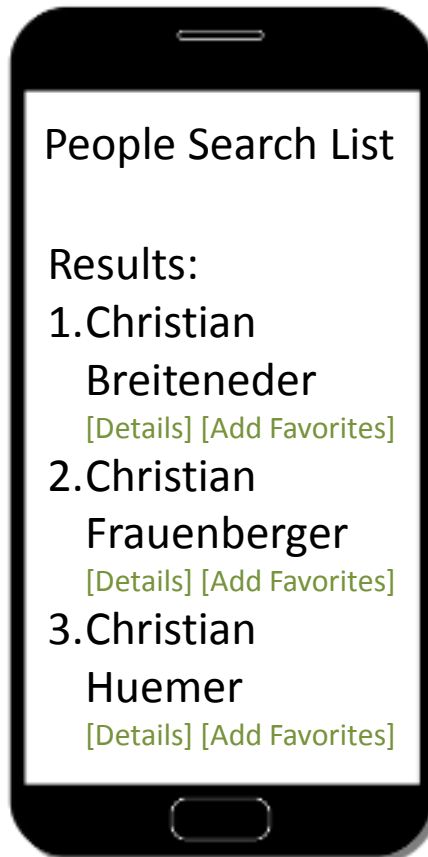


User Interface Examples

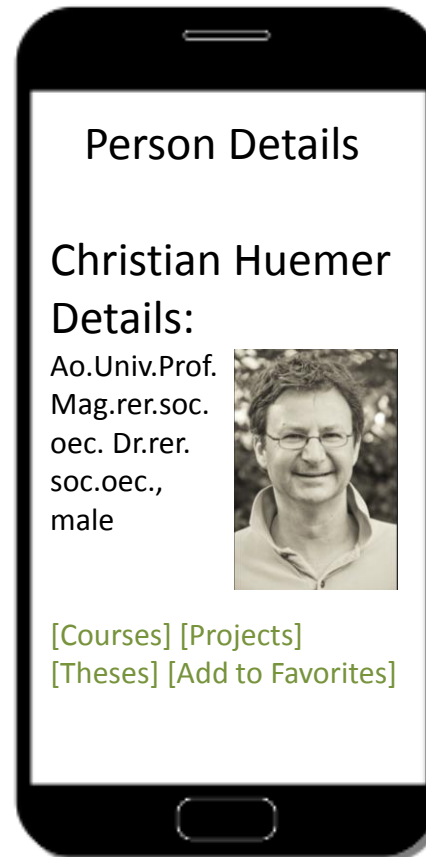
1. Search



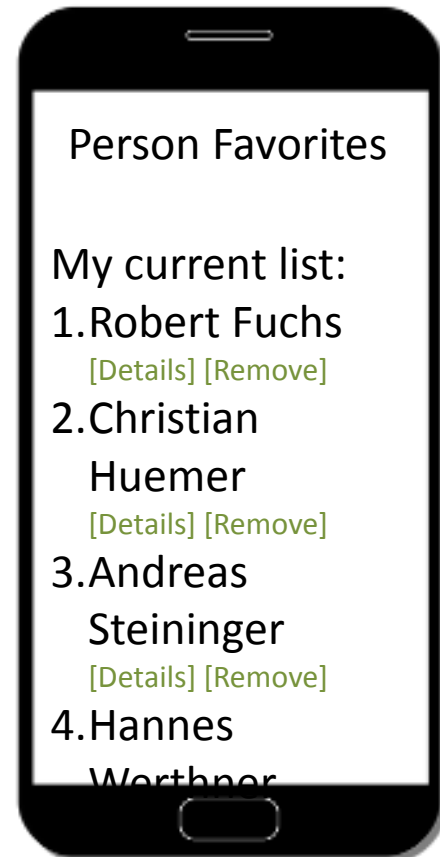
2. List



3. Details



4. Favorites



Application Requirements

1. Main menu with login/registration and links to the four search functions (after login)
2. Implement search functions based on public REST interfaces for people, courses, projects and theses at TU Wien
3. Use model-based validation in order to guarantee that no empty search requests are performed
4. Present the search results in form of a list that allows for
 1. Selection of only entry for detailed view
 2. Registration of an entry in a personal list
5. The detail view and all other pages have to be optimized for viewing on smartphones and other mobile devices – based on a self-defined CSS styling
6. Personalized lists must allow for the viewing of details, removal of entries and sorting by registration date and title A-Z



Formal Requirements

1. The entire system must be implemented as a gateway server based on Ruby on Rails
2. Data acquisition must be performed live based on REST interfaces to the TU Wien TISS system
3. Styling aspects must be encapsulated in a self-defined CSS stylesheet
4. User management (CRUD): users must be distinguished by server-based accounts of username and password



Recommended Approach

1. Do not split tasks but **work together as a team**
2. Install the RoR environment and a Cygwin Linux environment
3. Learn to work with the Cygwin command line for managing the Ruby environment
4. If you do not know Ruby yet, **learn programming Ruby** (see list of references below for tutorials)
5. **Then, learn Rails** through the beginners guide (see link below)
6. Implement a demo project and extend it by CRUD functions and database extensions
7. **Experiment with the REST/JSON** interfaces and embed them in your project
8. Glue everything together and implement all requirements
9. Refurbish your app through nice CSS stylesheets
10. Test everything, pack it and **submit on time**



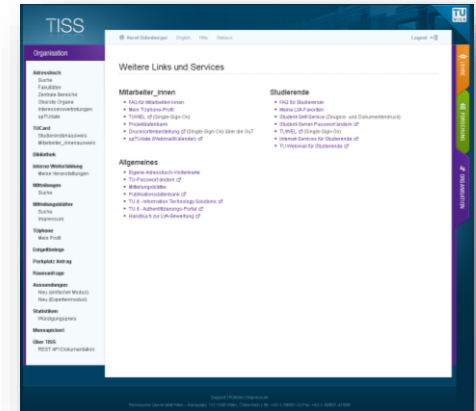
Approach: Practical Issues

- Notepad++ is a nice editor for Ruby projects. See below for the download link.
- RoR uses many conventions. Learn them thoroughly in order to understand what is going on in your code.
- Document your source code well, i.e. on three levels: inline (for complex operations), API (classes, methods, etc.) and architecture (in the readme file).
- Ruby HTTP requires a certificate for reading from REST interfaces. See here for first-time installation:
<https://stackoverflow.com/questions/4528101/ssl-connect-returned-1-errno-0-state-sslv3-read-server-certificate-b-certificat>

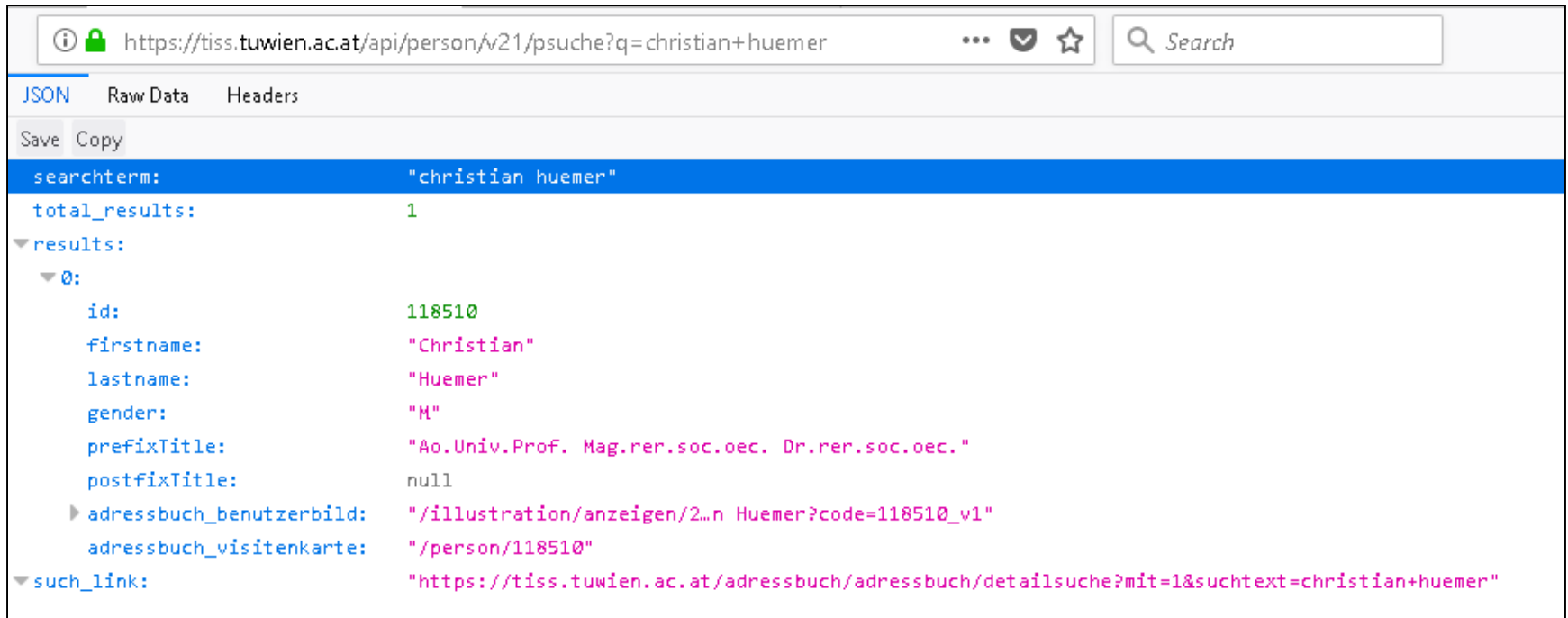


Available Services

- **People:**
[https://tiss.tuwien.ac.at/api/person/v21/psuche?q=<search_string>; e.g. „horst+eidenberger“](https://tiss.tuwien.ac.at/api/person/v21/psuche?q=<search_string>; e.g. „horst+eidenberger“ [&max_treffer=<max_hits>])
[&max_treffer=<max_hits>]
- **Courses:**
[https://tiss.tuwien.ac.at/api/search/course/v1.0/quickSearch?searchterm=<search_string>; e.g. „software+engineering“](https://tiss.tuwien.ac.at/api/search/course/v1.0/quickSearch?searchterm=<search_string>; e.g. „software+engineering“ [&count=<number_of_hits>] [&offset=<number_first_hit+1>] [&locale={DE | EN}])
[&count=<number_of_hits>]
[&offset=<number_first_hit+1>]
[&locale={DE | EN}]
- **Projects:**
[https://tiss.tuwien.ac.at/api/search/projectFullSearch/v1.0/projects?searchterm=<search_string>; e.g. „nano+technology“](https://tiss.tuwien.ac.at/api/search/projectFullSearch/v1.0/projects?searchterm=<search_string>; e.g. „nano+technology“ [&count=<number_of_hits>] [&offset=<number_first_hit+1>] [&locale={DE | EN}])
[&count=<number_of_hits>]
[&offset=<number_first_hit+1>]
[&locale={DE | EN}]
- **Theses:**
[https://tiss.tuwien.ac.at/api/search/thesis/v1.0/quickSearch?searchterm=<search_string>; e.g. „virtual+reality“](https://tiss.tuwien.ac.at/api/search/thesis/v1.0/quickSearch?searchterm=<search_string>; e.g. „virtual+reality“ [&count=<number_of_hits>] [&offset=<number_first_hit+1>])
[&count=<number_of_hits>]
[&offset=<number_first_hit+1>]



Usage of REST Services



The screenshot shows a web browser window with the address bar displaying the URL: `https://tiss.tuwien.ac.at/api/person/v21/psuche?q=christian+huemer`. The browser interface includes a search bar and navigation icons. Below the address bar, the 'JSON' tab is selected, showing the response data in a collapsible tree structure. The data is as follows:

```
{
  "searchterm": "christian huemer",
  "total_results": 1,
  "results": {
    "0": {
      "id": 118510,
      "firstname": "Christian",
      "lastname": "Huemer",
      "gender": "M",
      "prefixTitle": "Ao.Univ.Prof. Mag.rer.soc.oec. Dr.rer.soc.oec.",
      "postfixTitle": null,
      "adressbuch_benutzerbild": "/illustration/anzeigen/2...n Huemer?code=118510_v1",
      "adressbuch_visitenkarte": "/person/118510"
    }
  },
  "such_link": "https://tiss.tuwien.ac.at/adressbuch/adressbuch/detailsuche?mit=1&suchtext=christian+huemer"
}
```

- Use HTTP get for REST Calls and fill parameters into the URL string
- All results are JSON-encoded. Use the browser to understand the fields and values

See www.ims.tuwien.ac.at for more information.

Regulations for Hand-ins

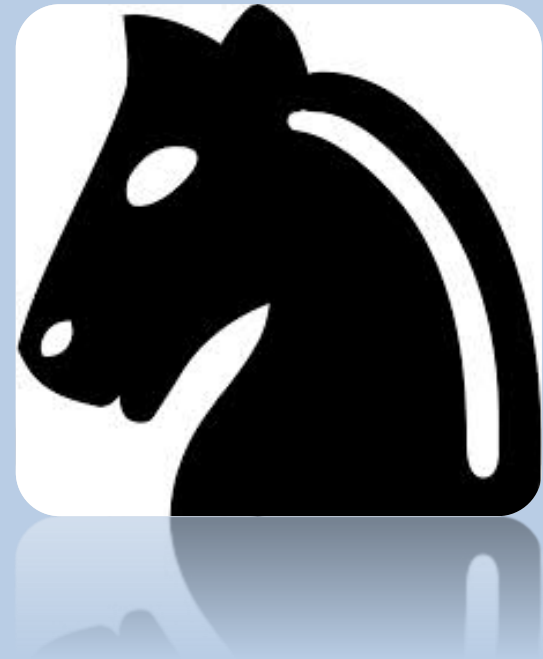
- Register your project at the TUWEL course page
- Define and use a unique lab project name as RoR project name (must not contain whitespace characters)
- When finished, zip your project folder and e-mail it to the supervisor (if too large, please use a cloud service for storage)
- The hand-in must also comprise:
 - A readme file with student data + RoR version information + all relevant usage information
 - The time sheets of the participating students
- Your supervisor will reply with a list of time slots for the face-to-face discussion of the project results with all group members („Abgabegespräch“)



Summary

1. Implement all application and formal requirements.
2. Follow the recommended process.
3. Employ Cygwin for server management.



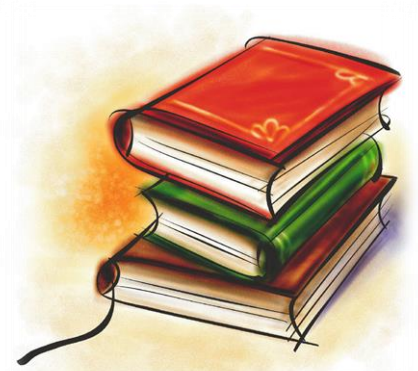


FRONTMATTER
LECTURE
PROJECT

BACKMATTER

- Literature
- More Information

Useful Resources



- Tutorials, guides, etc.:
 - General RoR guides: <http://guides.rubyonrails.org/>
 - Beginners RoR guide: http://guides.rubyonrails.org/getting_started.html
 - Comprehensive RoR tutorial: <https://www.railstutorial.org/book>
 - Ruby tutorial: <https://www.tutorialspoint.com/ruby/>
- Useful software:
 - Ruby on Rails Installer: <http://railsinstaller.org/en>
 - Notepad++ Editor: <https://notepad-plus-plus.org>
 - Cygwin for server management and programming: <https://www.cygwin.com/>
- Modules, APIs, etc.:
 - Ruby HTTP: <https://ruby-doc.org/stdlib-2.4.2/libdoc/net/http/rdoc/Net/HTTP.html>
 - Ruby JSON: <http://ruby-doc.org/stdlib-2.0.0/libdoc/json/rdoc/JSON.html>
 - SSL certificate issue: <https://stackoverflow.com/questions/4528101/ssl-connect-returned-1-errno-0-state-ssl3-read-server-certificate-b-certificat>

Do's and Don'ts

- Do's:
 - Be active!
 - Ask questions when they occur!
 - Take notes!
- Don'ts:
 - **Never use any text of someone else in any of your submissions without proper citation.**
 - **Never submit source code under your name that was written by someone else.**
- See the faculty web page for proper citations:
<http://www.informatik.tuwien.ac.at/lehre/richtlinien/index.html>
- Please follow the TU guideline for plagiarism:
http://www.tuwien.ac.at/fileadmin/t/ukanzlei/Lehre_-_Leitfaden_zum_Umgang_mit_Plagiaten.pdf
- Consequences:
 - Negative mark on the course
 - No re-sit, no negotiating, etc.



More Information

- Literature, HOWTOs, etc. can (soon) be found in TUWEL! Please contribute your own material.
- Use the TUWEL course for questions etc.:
<https://tuwel.tuwien.ac.at/course/view.php?id=13034>
- E-mail contact (last resort): Horst Eidenberger
eidenberger@tuwien.ac.at
- Be patient, I am part-time (Tue, Thu, Fri)!



Summary



1. Learn Ruby and the Rails framework.
2. Implement the requirements.
3. Submit all project Information on time.
4. Use the TUWEL forum for questions.

The End

Thank you!