Introduction to Ruby (on Rails)

Scalable Software Engineering
WS 2021/22

Enterprise Platform and Integration Concepts

Ruby

Following slides partly based on https://speakerdeck.com/y310/ruby-on-rails-introduction

Ruby

Some basic facts

http://www.ruby-lang.org/

- **Interpreted**, high-level, general-purpose language
- Supports multiple programming paradigms, incl.
  - procedural
  - object-oriented
  - Functional
- **Dynamically typed**
- Influenced by Perl, Smalltalk, Eiffel, and Lisp
- **Open-source, mature software**
- Different implementations available, e.g. JRuby

Yukihiro "Matz" Matsumoto with R. Stallman

Differences: interpreted vs …?, procedural vs. functional? Dynamically vs. statically typed?

Basic Ruby Syntax

Object-orientation and procedural programming

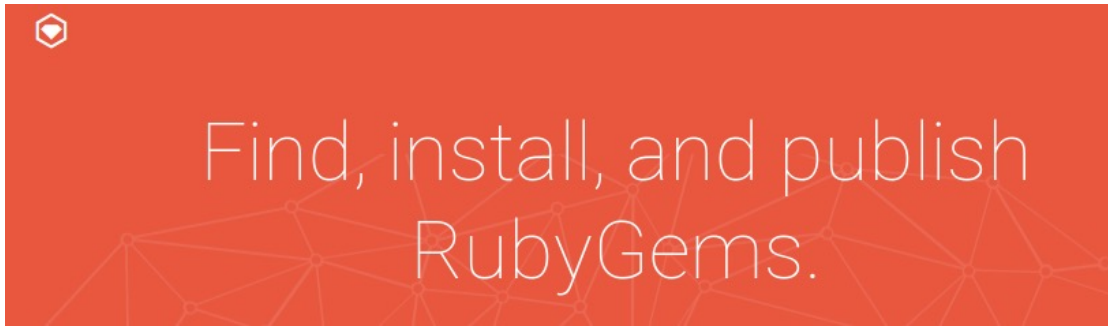
class declaration →
instance variable →
method declaration →
Instantiation →
Method call →

```
$ cat hello.rb
class Hello
  def initialize(name)
    @name = name
  end

  def say
    puts "Hello, #{@name}!"
  end
end

hello = Hello.new("David")
hello.say

$ ruby hello.rb
#=> Hello, David!
```

Important Ruby Tools

Tools needed for every day development

- Ruby version manager: e.g. **rbenv** or RVM
  - `$ rbenv install ruby 2.5.0`
- Package manager: **RubyGems**
  - `$ gem install rails`
  - Like *pip* for Python, *npm* for Node.js
- Dependency management: **bundler**
  - Persist list of dependencies in **Gemfile**
  - Setup same dependencies everywhere




RAILS

History of Ruby on Rails

Web application development framework written in Ruby

http://rubyonrails.org/

Version	Release Date	Compatible Ruby Version(s)[25][26][27]
1.0[28]	December 13, 2005	1.8.6
1.2[29]	January 19, 2007	1.8.6
2.0[30]	December 7, 2007	1.8.6
2.1[31]	June 1, 2008	1.8.6
2.2[32]	November 21, 2008	1.8.7 recommended; 1.8.6 possible
2.3[33]	March 16, 2009	1.8.7 recommended; 1.8.6 and 1.9.1 possible
3.0[34]	August 29, 2010	1.9.3 recommended; 1.8.7 and 1.9.2 possible
3.1[35]	August 31, 2011	1.9.3 recommended; 1.8.7 and 1.9.2 possible
3.2[36]	January 20, 2012	1.9.3 recommended; 1.8.7 and 1.9.2 possible
4.0[37]	June 25, 2013	2.0 preferred; 1.9.3 or newer required
4.1[17]	April 8, 2014	2.0 preferred; 1.9.3 or newer required
4.2[18]	December 19, 2014	2.0 preferred; 1.9.3 or newer required
5.0[19]	June 30, 2016	2.2.2 or newer
5.1[20]	May 10, 2017	2.2.2 or newer
5.2[21]	April 9, 2018	2.2.2 or newer
6.0[23]	August 16, 2019	2.5.0 or newer
6.1[24]	December 9, 2020	2.5.0 or newer

https://en.wikipedia.org/wiki/Ruby_on_Rails#History

Ruby on Rails

Philosophy

- "Don't repeat yourself" – DRY
- Convention over Configuration
  - **There is "The Rails Way",** try to understand the framework, not fight it
- RESTful architecture
- Everything in its place
- **No prebuilt backend**, generators for creating needed code

More on the details of REST in a little bit…

Adopters

- GitHub, Shopify, Airbnb, Twitch, SoundCloud, **openHPI**

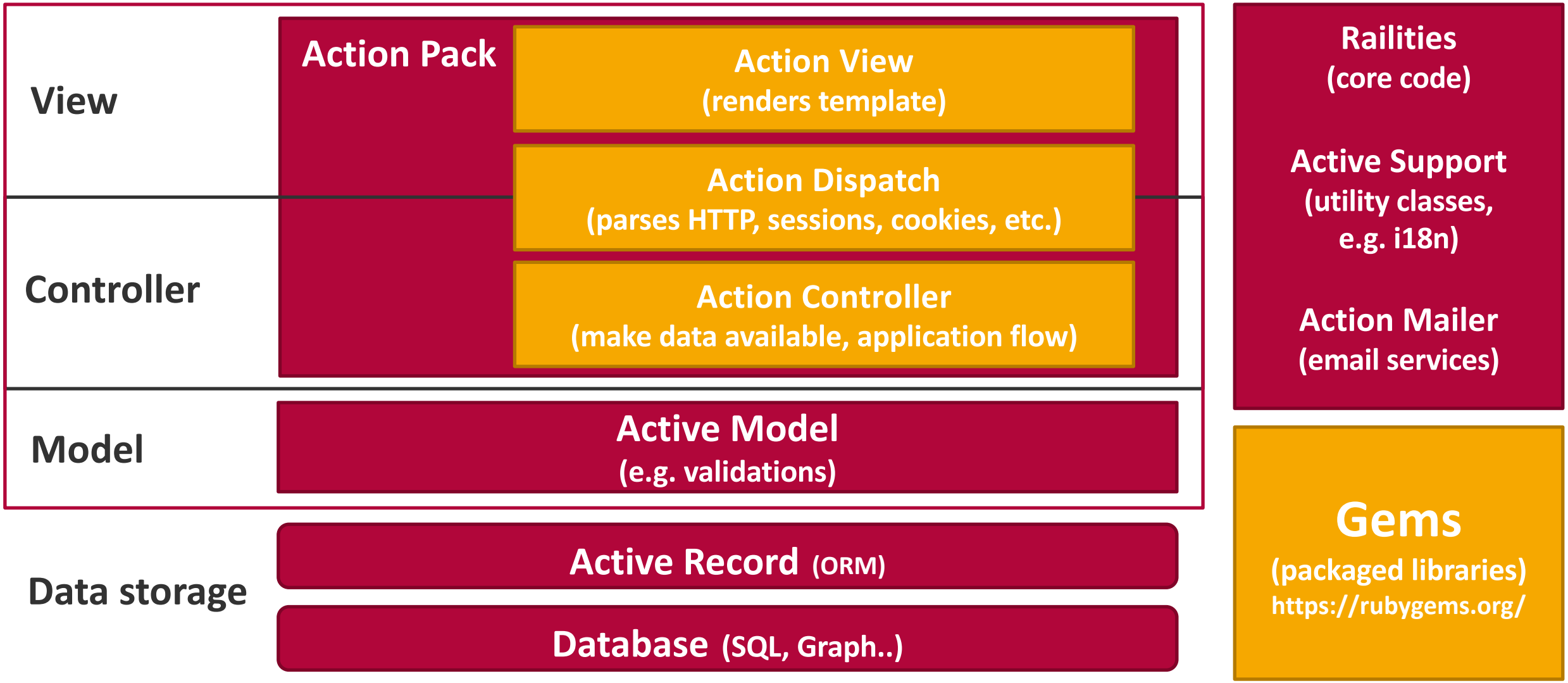
Convention Over Configuration

Founding Ideas

- Web app basics are always similar: there are **conventions**
  - Database layout, e.g. tables for model instances
  - Interfaces for create, read, update delete (CRUD) functionality
  - Setting these up manually is time consuming and adds little value
- Ruby on Rails has **hardcoded these conventions**
  - E.g. AuthorsController expects '`new`' view in '`app/views/authors/new.html.erb`'
  - If you want to deviate, this must be configured
- Included generators produce files adhering to conventions

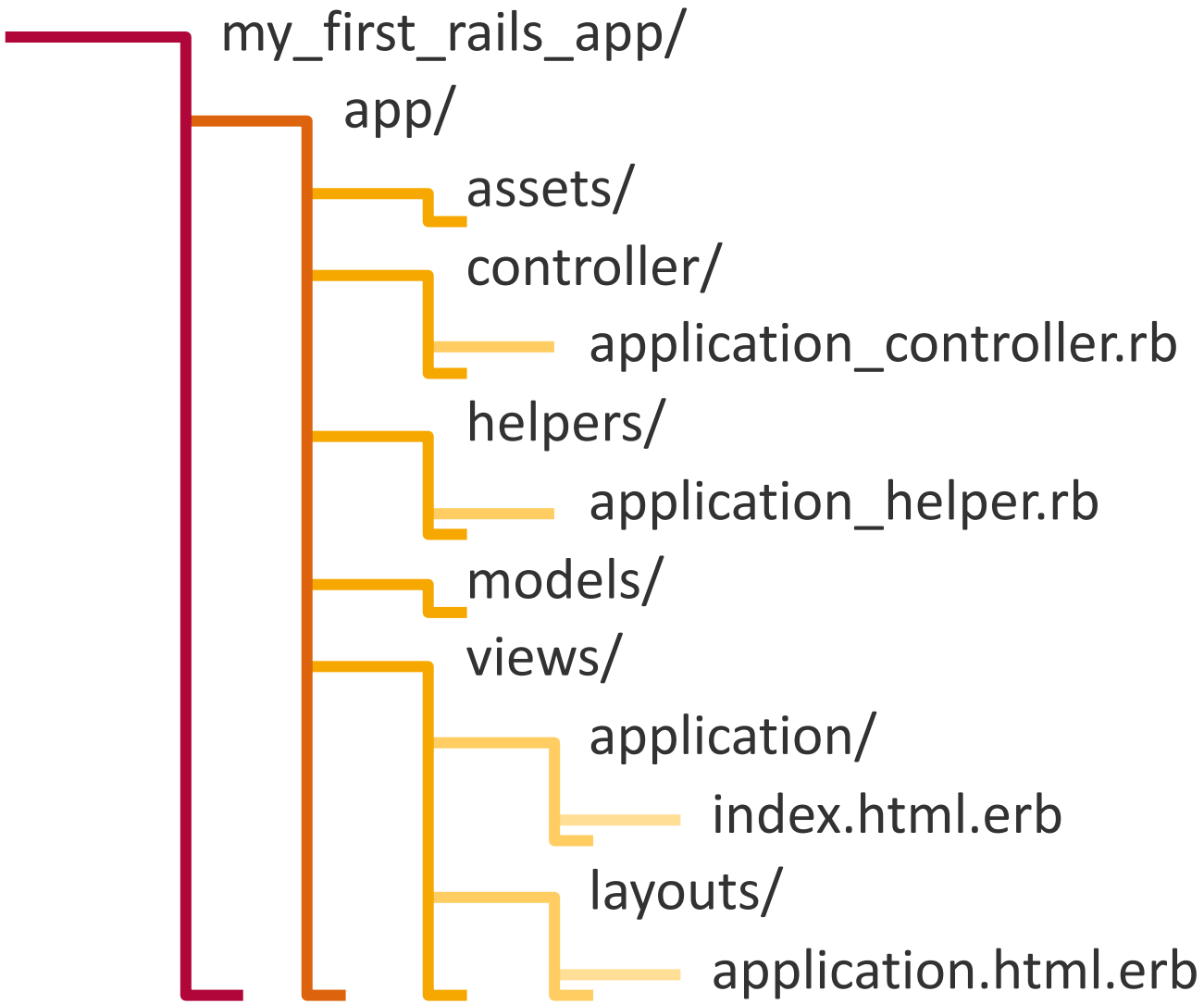
Further reading:
https://rubyonrails.org/doctrine/#convention-over-configuration

Rails Core Components



http://adrianmejia.com/blog/2011/08/11/ruby-on-rails-architectural-design/

Rails Application Layout



RESTful Architecture

- **Re**presentational **S**tate **T**ransfer (REST) is a software architecture style
- Guidelines for creating web APIs, when they obey REST constraints -> **RESTful**

- **Client-server architecture**
  - Separate user interface concerns from data storage concerns
- **Stateless interactions**
  - No session information is retained by the server
- **Cacheability**
  - Clients and intermediaries can cache responses
- **Layered system**
  - No difference whether client is connected to end server or a proxy
- **Uniform interface**
  - Resource representation is sufficient to modify resource's state

Examples of RESTful Routes

- GET / # invoke “home” controller
- GET /authors # retrieve a list of all authors
- GET /authors/new # get the form to enter a new author
- POST /authors # create a new author
- GET /authors/1 # show details of the first author
- GET /authors/1/edit # get the form to edit the first author
- PUT /authors/1 # update the first author
- DELETE /authors/1 # delete the first author

Once you know about the `/authors` route, you can deduce the REST

Development in Ruby on Rails

Scalable Software Engineering
WS 2021/22

Enterprise Platform and Integration Concepts

How to Start?

- **MacOS or Linux**
  - Install natively, use Ruby version manager (e.g. rbenv, RVM)
  - http://guides.rubyonrails.org/getting_started.html#installing-rails

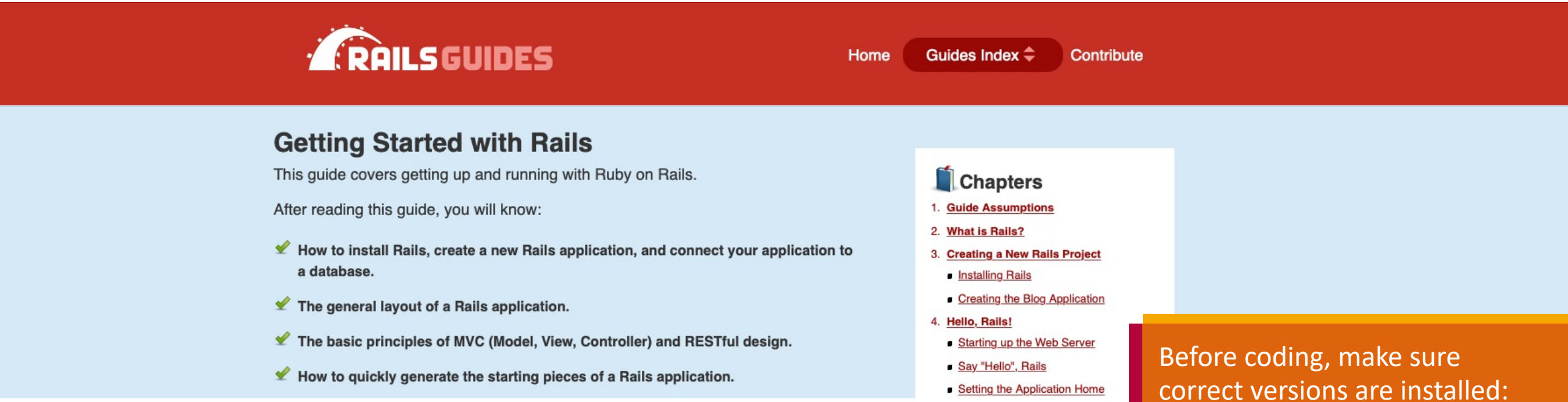
- **Windows**
  - Install Windows Subsystem for Linux (WSL), install Ubuntu
  - Follow Linux instructions
  - Use Windows tools & editors

- **You want separation**
  - Use virtual machine or container
  - E.g. VirtualBox (https://www.virtualbox.org/) & Vagrant (https://www.vagrantup.com/)
  - Use your own tools & editors, run the project in a headless VM

Comprehensive RoR tutorial

Recommended to work through/read the hands-on tutorial.

We promise it will help: https://guides.rubyonrails.org/getting_started.html



Before coding, make sure correct versions are installed:

```
$ rails --version
```

The following slides give a general overview

Bundler – Ruby package manager

- Ruby libraries: gems
- **Large ecosystem**
  - Most likely there is a gem for your need!
- `Gemfile` holds a list of required gems
  - Specify versions, e.g. gem 'rails' >= '4.1.6'
- `Gemfile.lock` contains resolved dependencies
  - Should be under version control

```
# Bundle edge Rails instead: gem 'rails', github: 'rails/rails'
gem 'rails', '4.1.6'
# Use sqlite3 as the database for Active Record
gem 'sqlite3', group: :development

# use postgresql in production (for deployment on heroku)
gem 'pg', group: :production

# Use Bootstrap, see app/assets/stylesheets
gem 'twitter-bootstrap-rails'
```

`'--without'`,
e.g. --without production,
ignores parts of the Gemfile

`Gemfile.lock` contains all currently installed versions of gems.

Manually install a gem (Ruby package)

```
$ gem install
```

Install all gems listed as depencies in Gemfile

```
$ bundle install
```

rails – Main executable

Start interactive shell to test out ideas

```
$ rails console
```

Start new rails application

```
$ rails new
```

Generate boilerplate for models, controllers & views (vs ready-made backends of e.g. Django)

```
$ rails generate
```

Start the development server

```
$ rails server
```

Start a direct database shell

```
$ rails dbconsole
```

Example: generate model, controller and view without controller specs

```
$ rails g scaffold author last_name:string
homepage:string --controller-specs false
```


http://guides.rubyonrails.org/command_line.html

rails – The cmd Utility

List all available commands

```
$ rails --help
```

List all routes, i.e. application URIs and controller actions

```
$ rails routes
```

```
$ rails db:setup db:migrate
```

Replace database with db layout from db/schema.rb
Do not run migrations.

```
$ rails db:schema:load
```

Run Rspec (testing framework for RoR) tests

```
$ bundle exec rspec
```

Reading about migrations will be very helpful!
Why are they needed?!

Git – Distributed VCS

- **Install Git:**
  - sudo apt-get install git
  - http://git-scm.com/ (Installers for all systems)

- **Setting up user name and email:**
  - Mandatory to commit changes
  - Use email connected to GitHub

```
$ cd /path/to/your/project
$ git config user.email "first.last@student.hpi.de"
$ git config user.name "Max Mustermann"
```

Quick Git Workflow

Checkout remote repository to local copy

```
$ git clone https://github.com/hpi-swt2/<insert_project>
```

Change main layout template app/views/layouts/application.html.erb

Stage changes (add files from working copy to repository index)

```
$ git add app/views/layouts
```

List changes to be committed

```
$ git status
```

Commit with commit messages. Reference Github issue #25

```
$ git commit –m "Fixed issue #25"
```

Fetch and merge changes from remote repository

```
$ git pull
```

Publish local commits

```
$ git push
```

When cloning: be aware of difference between HTTPS & SSH

Introductory Exercise

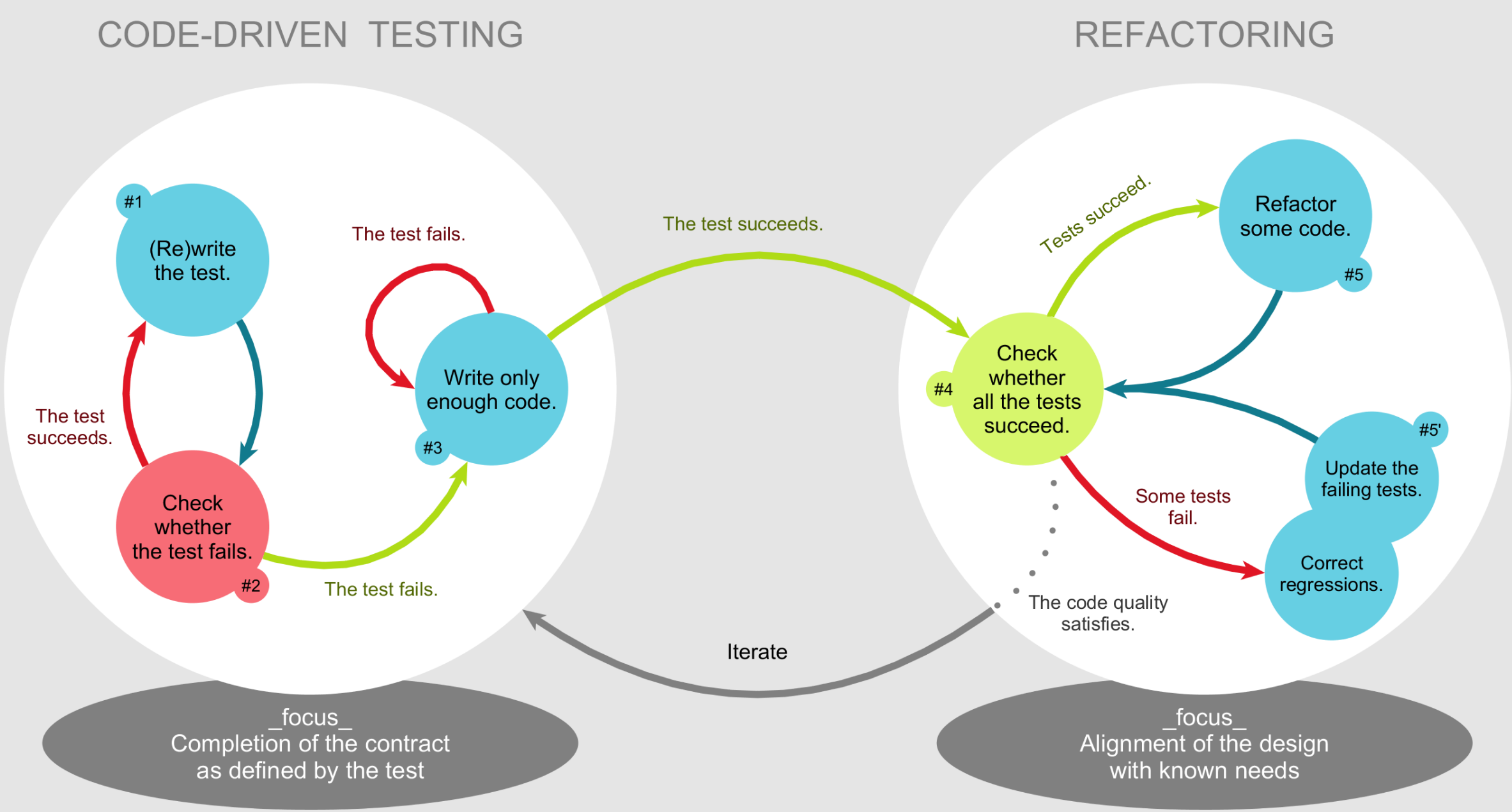
Scalable Software Engineering
WS 2021/22

Enterprise Platform and Integration Concepts

Introductory Exercise

- **Goals**
  - Get familiar with Ruby (on Rails)
  - Create necessary accounts
  - Setup your machine, editors and tools
  - Apply **Test-First Development** *(why?)*
- **Notes**
  - Product Owners are not required to complete this task
  - Product Owners will work on project envisioning

Check your favorite editor for Ruby on Rails plug-ins/extensions.
Better tools == faster & more fun



TEST-DRIVEN DEVELOPMENT

https://upload.wikimedia.org/wikipedia/commons/0/0b/TDD_Global_Lifecycle.png

Acceptance Test-Driven Development

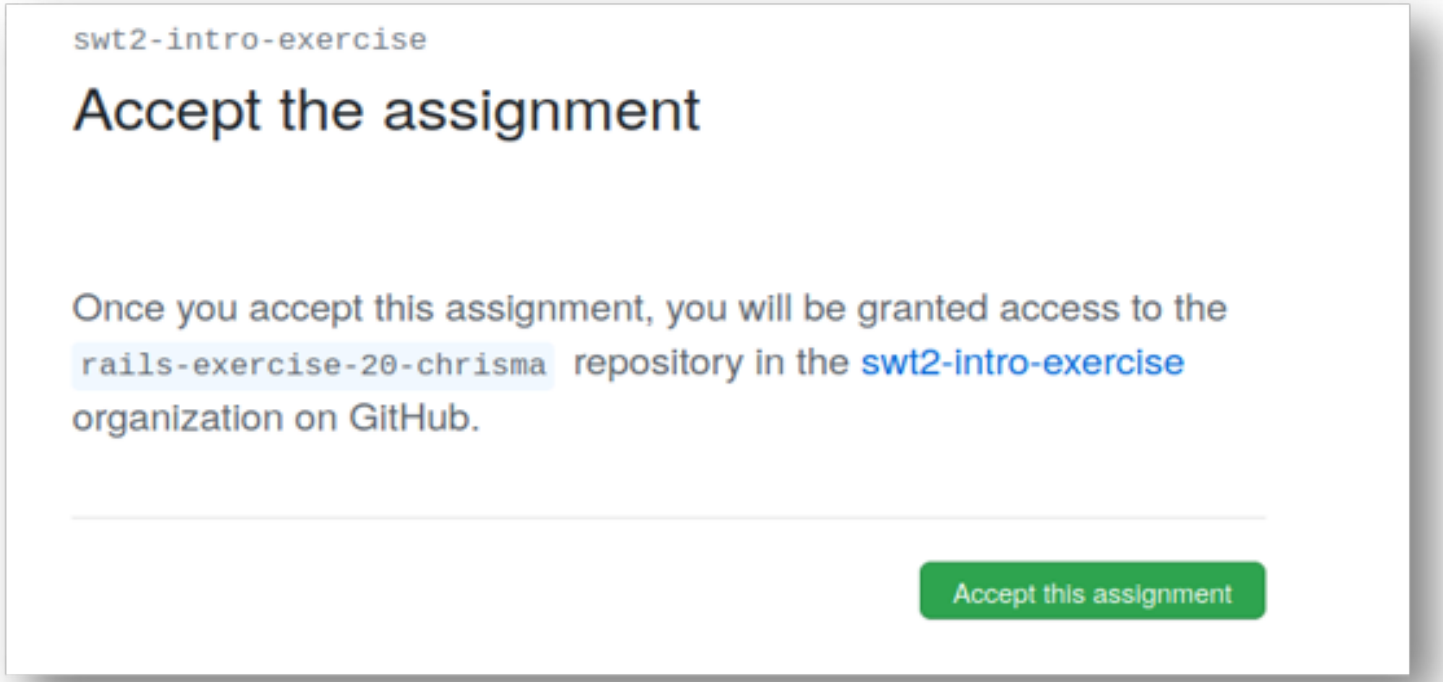
- **Acceptance Test**
  - Determine if details of specification (i.e. acceptance criteria) are met
  - May involve **black-box testing of system**
  - Enable **stakeholders** to determine whether requirements are already met
- **Test-Driven Development (TDD)**
  - **Developer's tool** to help create code that correctly performs operations
- **Acceptance Test-Driven Development (ATDD)**
  - **Communication tool** between customer, developer & tester
  - TDD requires test automation. ATDD does not, although it helps
  - ATDD tests should be readable by the customer

Acceptance testing may also be referred to as *user acceptance testing (UAT)*, *end-user testing* or *field testing*

Introductory Exercise

ATDD in an exercise

- Requirements are given
- **You write (acceptance) tests to check your solutions**
- Automated service provides next requirements



Next Steps

- **Get to know Ruby on Rails and web development**
  - Setup your machine and tools
  - Read the Ruby on Rails guide (https://guides.rubyonrails.org/getting_started.html)
  - Understand the core Ruby language constructs (coming from other languages) (https://www.ruby-lang.org/en/documentation/ruby-from-other-languages/)

- **Start the exercise**
  - Visit GitHub classroom link (on the course website)
  - Accept the assignment
  - Follow the instructions in the README file