# Exercise 5 – Spring

*תזכורת: יש* [*באתר הקורס*](https://drive.google.com/drive/folders/1dNnLAetBHDe42g-6yPiB6mrcKzckSvsf?usp=share_link) *מסמכים הכוללים רשימת בעיות ופתרונות בפרויקטים, מומלץ לדפדף שם לפני שאתם מחפשים ברשת או שולחים מייל!*

Neviim: <https://classroom.github.com/a/e99KNix9>

Strauss Women: <https://classroom.github.com/a/hW8nnBf1>

Straus Men: <https://classroom.github.com/a/RqN2jg-->

Goal: build a website combining the Spring technology taught in this course.

המטרה היא להשתמש ב-Spring MVC בלבד (ללא React) ולהעביר את כל ניהול לוגיקת האתר אל הצד שרת (בניגוד ל-React)

The website topic is up to you. We have some suggestions below if you’re out of ideas.

You don’t need to build a big website with lots of functionality, you need a fully working website based on the Spring technology taught in this course that fulfill these requirements:

1. A full SpringBoot MVC project : the MVC structure of your project uses EJS view engine, controllers, beans. The logic is server side, you will write limited Javascript logic if needed (such as validation, polling…)
2. There should be at least 3 major pages. All html pages are built with EJS (server side dynamically generated pages). You may implement one page in a SPA style (by developing your own REST api).
3. Use session : for example to store a cart, game current state, user authentication
4. Use Beans, dependency injection: make sure to inject beans in your controllers (do not access application/session/request scope directly from the http request)
5. Storing persistent data with JPA - mySQL database named “**ex5**” with at least 2 repo beans (SQL tables) with a relation: for example, *User* and *Order* with a one-to-many relation (no data duplication!).

Optionally use additional material (removed from this year’s program, and possibly presented in last couple weeks):

1. The most useful: Spring Security provides user authentication and authorization, and saves you the hassle of implementing user login (a lot of useful material is available on Baeldung’s website).
2. Interceptors : interceptors are similar to Express middleware to avoid code duplication and process http requests before your controllers, for example to redirect unauthenticated user (if you don’t use Spring Security)

## Grading criteria

* Completeness of the functionality
  + it is better to submit a limited and fully working website, than a pile of code that does not run.
  + It is acceptable to have minor bugs
* Fulfilling technology requirements (should result in very clean and modular code)
* Quality of the UI experience (navigation, design)
* Robustness: server-side integrity (server side validation, thread safe coding)
* your ability to answer any questions about your solution: after submission, some of you may be required to demo their project
* *Please do not submit someone else’s code, we hate giving zero grades*.

Here are some suggestions:

* You could re-implement something similar to previous exercise (A pizza restaurant with online orders) putting now all logic server side. Use session to store the cart, create users and orders tables, allow a user to see all of their orders by entering email and phone number.
* A course registration system: admin can create courses with maximum number of participants, users can register to one (or X) class(es). Admin can browse/edit classes, remove participants. No need for user registration, the form to join a class should automatically register and identify users based on their email.
* A game (such as submarines…)
* A website for posting ads
* A chatroom
* A backend for a store (handle products, stock, orders and payments)

## Notes:

* In order to fulfill the use of JPA-SQL, store some application state in a database: user settings, high score, history…
* For any website, you could develop an admin page with a single admin account, to manage anything relevant to the website , for example in the case of a game, you may allow admin to erase the high score, set some game parameters stored in database), consult some history/logfile of actions etc…

## Submission:

* If your project is ready to be demo-ed on July 1/7 (last class at 12:00), you may present your exercise on zoom to the teacher. In this case, please contact Solange to set a time.  
  Otherwise:
  + In addition to your code, you should submit a short recording demo of your website
    - Length of the recording limited to 5 minutes
    - Continuous straight recording, no editing/cut
    - If you are a team of 2, make sure to split the task of presenting
    - Upload the recording to your repo or put a link in your README.md
* Your readme should explain:
  + The general functionality of the website
  + Any known bugs
  + How to run:
    - How to initialize the website (if any DB is needed for example)
    - Any credentials (for example the admin login and password)

## Good luck!