



MizDooni

مقدمه

هدف از این فاز پروژه، آشنایی با ابزار جاوا سرولت و کانتینر تامکت، طراحی MVC و رندر کردن سمت سرور با استفاده از JSP می باشد. همینطور مانند فاز قبل به پیاده سازی چند ویژگی دیگر از دامنه پرداخته می شود.

پیش نیازها

ابتدا لازم است که تامکت را در سیستم خود نصب و راه اندازی کنید. برای نصب و راه اندازی این ابزار و اجرا کردن یک سرولت و JSP روی آن، از [این آموزش](#) استفاده کنید. علاوه بر محتوای مطرح شده در کلاس درس، در صورت نیاز می توانید از آموزش های موجود در [این مجموعه](#) برای آشنایی بیشتر با سرولت ها استفاده کنید. در مرحله بعد لازم است که قابلیت های جدیدی که معرفی می شوند را پیاده سازی کنید و سپس در ادامه خروجی مجموعه سرویس هایتان را به وسیله JSP تولید کنید. همچنین در پیاده سازی این پروژه باید از طراحی MVC استفاده کنید. در مورد این طراحی و نحوه پیاده سازی آن می توانید از [این آموزش](#) یا آموزش های مشابه استفاده کنید.

تغییرات و قابلیت‌های جدید

● بازخورد کاربران

○ کاربر بازخورد خود به یک رستوران که شامل امتیازات به سرویس‌دهی (Service)، غذا (Food)، محیط (Ambiance) و امتیاز کلی (Overall) و کامنت می‌باشد را فقط یکبار می‌تواند ثبت کند و اگر دوباره بخواهد برای آن رستوران بازخورد ثبت کند، بازخورد جدید با تاریخ جدید جایگزین می‌شود. (بازخورد قبلی حذف می‌شود)

○ فقط کاربرانی می‌توانند برای یک رستوران بازخورد ثبت کنند که میزی از آن رستوران رزرو کرده باشند و زمانی که می‌خواهند به آن رستوران بازخورد دهند، از زمان رزرو میز گذشته باشد. همچنین کاربرانی که رزرو خود را لغو کرده‌اند، نباید بتوانند بازخورد ثبت کنند.

● اضافه کردن میانگین امتیازها به رستوران

هر رستوران باید میانگین امتیازهای کاربران را داشته باشد. این میانگین، شامل میانگین امتیازات به سرویس‌دهی (Service)، غذا (Food)، محیط (Ambiance) و امتیاز کلی (Overall) می‌باشد. توجه داشته باشید زمانی که هنوز هیچ بازخوردی برای یک رستوران توسط کاربران ثبت نشده باشد، این میانگین‌ها باید صفر در نظر گرفته شوند.

صفحات JSP

خروجی تمامی سرویس‌های شما در قالب چند صفحه JSP به کاربر نمایش داده می‌شوند. در این فاز ظاهر صفحات اهمیت ندارند و باید از قالب‌هایی که در اختیارتان قرار داده شده است استفاده کنید (در فازهای بعدی به مسئله فرانت‌اند و ظاهر پروژه بیشتر پرداخته می‌شود). می‌توانید در صورت نیاز به این قالب‌ها اطلاعات بیشتری اضافه کنید. شما شش صفحه اصلی خانه‌ی کاربر، خانه‌ی مدیر رستوران، لاگین، رستوران‌ها، یک رستوران و میزهای رزرو شده توسط کاربر را برای MizDooni از سمت سرور رندر می‌کنید.

صفحه‌ی خانه

/

این صفحه از طریق آدرس بالا در دسترس قرار می‌گیرد. قالب آن client_home.html یا manager_home.html است و شامل تعدادی لینک به صفحات دیگر برنامه شماست که کاربر می‌تواند از آن‌ها استفاده کند. توجه کنید این صفحه با توجه به نوع کاربر لاگین شده باید متفاوت باشد (کاربر client یا manager):

- صفحه کاربر عادی (client):

- تنها پارامتر پویای آن که لازم است به آن توجه داشته باشید نام کاربری فرد در بالای صفحه است.

- صفحه کاربر مدیر رستوران (manager):

- در این صفحه علاوه بر نام کاربری در بالای صفحه تنها لینک logout قرار دارد و لینکی به صفحات

دیگر قرار ندارد. در واقع کاربر manager امکان دیدن اطلاعات رستوران‌ها و رزرو میز را ندارد.

- همچنین در صفحه خانه مدیر رستوران، باید اطلاعات رستوران شامل نام، نوع، لیست میزها و ... قرار

گرفته باشد و تنها عملیاتی که مدیر رستوران می‌تواند انجام دهد اضافه کردن میز به رستوران است که در

قالب یک فرم انجام می‌شود.

صفحه‌ی لاگین

/login

در این صفحه نام کاربری و رمز کاربر دریافت شده و سرویس‌ها در درخواست‌های بعدی با توجه به آن کار خود را انجام می‌دهند. اگر کاربر با موفقیت لاگین شد باید به صفحه خانه متناسب کاربر ریدایرکت شود. شما باید کاربر لاگین شده را یک متغیر جاوا ذخیره کنید و با توجه به آن، اطلاعات را تغییر دهید. همچنین اگر درخواستی برای برنامه شما آمد اما کاربری لاگین نکرده بود، باید به صفحه **/login** ریدایرکت شود. کاملاً ساده به این قابلیت نگاه کنید و خود را درگیر حالات خاص نکنید. کاربر از صفحه خانه با زدن دکمه (Log Out) باید به آدرس /logout برود. در صورتی که در هنگام login کردن، کاربر دیگری از قبل login بود (متغیری که برای نگه داشتن نام کاربری مشخص کرده‌اید مقدار داشت!)، کاربر جدید جای کاربر قبلی را می‌گیرد.

/logout

● صفحه‌ی رستوران‌ها

/restaurants

در این صفحه باید اطلاعات همه رستوران‌ها بصورت لیست مطابق نمونه‌ی `restaurants.html` در اختیار کاربر (client) قرار بگیرد و نام هر رستوران هم لینکی باشد که به صفحه آن رستوران اشاره می‌کند. همچنین در این فاز باید بخش‌های زیر برای جستجو و فیلتر رستوران‌ها به این صفحه اضافه شود:

● بخش جستجو رستوران بر اساس نام

کاربر می‌تواند با جستجو کردن با استفاده از اسم رستوران‌ها جداول را فیلتر کند. این جستجو باید بصورت شامل بودن نام جستجو شده در نام رستوران‌ها انجام شود. مثلاً اگر کاربر نام `food` را سرچ کرد، همه رستوران‌هایی که شامل کلمه `food` در نام آن‌ها است، باید نشان داده شود. (مثلاً `best food` یا `fast food` و ... همه نشان داده می‌شوند.)

● بخش جستجو رستوران بر اساس نوع

کاربر می‌تواند با استفاده از نوع رستوران (type) آن را فیلتر کند.

● بخش جستجو رستوران بر اساس شهر

کاربر می‌تواند با استفاده از نام شهر رستوران آن را فیلتر کند. در این بخش اسم شهر باید دقیق باشد مثلاً اگر `Teh` را وارد کرد نیاز به نشان دادن رستوران‌های شهر `Tehran` نیست!

● بخش مرتب‌سازی بر اساس بازخورد

کاربر می‌تواند با توجه به مجموع امتیازات (Overall Score) رستوران‌ها را بصورت صعودی مرتب کند. (نیاز به پیاده‌سازی مرتب‌سازی نزولی نیست)

● صفحه‌ی رستوران

/restaurants/{restaurant_id}

در این صفحه باید اطلاعات رستوران مورد نظر و لیست بازخوردهای کاربران مطابق نمونه restaurant.html نمایش داده شود. همچنین امکانات زیر در این صفحه باید در اختیار کاربر قرار بگیرد:

● بخش اضافه کردن بازخورد

کاربر باید بتواند بازخورد خود را به رستوران مورد نظر ثبت کند که شامل امتیاز (به کیفیت غذا، سرویس‌دهی، محیط رستوران و امتیاز کلی) و کامنت است که مانند فاز قبل، هر دو بازخورد بصورت همزمان و در یک مرحله انجام می‌شود. توجه کنید اگر کاربر برای یک رستوران چند بار بازخورد ثبت کند، در دفعه اول بازخورد جدید ثبت می‌شود اما در دفعات بعد بازخورد قبلی آپدیت می‌شود.

● بخش رزرو میز

در این قسمت کاربر می‌تواند از منوی dropdown نام میز و تاریخ و زمان خود را انتخاب کند و دکمه رزرو را بزند و اگر میز در آن زمان خالی بود، به صفحه reservations/ که در ادامه آمده ریدایرکت شود. همچنین اگر اروری رخ داد، با استفاده از تمپلیت error.html، ارور مورد نظر نمایش داده می‌شود.

● صفحه میزهای رزور شده توسط کاربر

/reservations

این صفحه لیست میزهای رزرو شده کاربر را نمایش می‌دهد و کاربر می‌تواند عملیات‌های زیر را در این صفحه انجام دهد:

● لغو رزرو میز

در کنار هر میز رزرو شده، یک دکمه‌ی cancel reservation قرار دارد که کاربر می‌تواند آن رزرو مورد نظر را لغو کند. بعد از لغو کردن، باید دوباره صفحه reservations/ آپدیت شود و میز مورد نظر از لیست میزهای رزرو شده حذف شده باشد.

● صفحات خطا

خطاهای موجود در این فاز در دو قالب error و 404 است:

- در صورت درخواست دسترسی به صفحه ای که وجود ندارد. باید خطای 404 نمایش داده شود.
- در صورت وقوع بقیه خطاها مانند رزرو میزی که قبلاً رزور شده یا رمز یا نام کاربری اشتباه در صفحه لاگین و ... باید از قالب error.html (با رندر شدن متن مناسب خطا بصورت پویا در سمت سرور) استفاده کنید.

نکات مهم

- در صفحاتی که چند فرم و چند دکمه داریم، برای اینکه تفاوت آنها در متد post مشخص شود، از پارامتری به نام action استفاده کردیم که در سرور بتوانید درخواستهای مختلف را تشخیص دهید و به هر کدام رسیدگی کنید.
- سعی کنید بخش منطق پروژه خود را از کار با I/O جدا کنید، چرا که در فازهای بعد فرمت خروجی تغییر می کند.
- در صورت وجود ابهام، فایل های قالب خروجی را مشاهده کنید و اگر ابهامتان رفع نشد سوال پرسید.

Software Engineering Best Practices

چهار اصل مهم و اساسی در مهندسی نرم افزار

در این قسمت، به چند اصل مهم در توسعه نرم افزار می پردازیم. در هر بخش، هر یک از این اصول به صورت خلاصه معرفی می شوند که فقط شما را با این مباحث آشنا می کنند. هدف از این بخش، صرفاً آشنایی شما با این اصول است.

KISS (Keep It Simple, Stupid)

اصل KISS یک اصل طراحی است که بیان می کند که طرح ها و یا سیستم ها باید تا حد امکان ساده باشند و تا جایی که امکان دارد، باید از پیچیدگی ها در یک سیستم پرهیز کرد. این اصل بر پایه این است که سادگی، بیشترین سطح پذیرش از سوی کاربر را دارد و امکان تعامل هر چه بهتر کاربر با سیستم را تضمین می کند. این اصطلاح برای اولین بار در نیروی دریایی ایالات متحده توسط یک مهندس استفاده شد و بر پایه این فرضیه مطرح شد که طرح ها باید به اندازه ای ساده باشند که توسط یک شخص در شرایط جنگی، تنها با برخی آموزش های اولیه، قابل درک باشد. برای مطالعه بیشتر، به این [لینک](#) مراجعه کنید.

DRY (Don't Repeat Yourself)

این اصل بیان می کند که تکرار در منطق برنامه باید از طریق abstraction، و تکرار در فرایندها باید از طریق اتوماسیون حذف شود. این اصل بر این اساس است که افزودن کد تکراری، میزان کار مورد نیاز برای گسترش و نگهداری (maintenance) نرم افزار را در آینده افزایش می دهد. همچنین، این اصل بیان می کند که اگر یک فرایند قابل خودکارسازی باشد، (مانند آزمون نرم افزار) ولی به صورت خودکار انجام نشود، می تواند به هدر رفتن وقت منجر شود و ممکن است به این دلیل توسط برنامه نویسان به صورت منظم انجام نشود.

YAGNI (You Aren't Gonna Need It)

این اصل بیان می کند که فقط زمانی باید یک ویژگی (feature) به یک نرم افزار اضافه شود که مورد نیاز باشد. به عبارت دیگر، اضافه کردن ویژگی های اضافی با فرض اینکه در آینده مورد استفاده قرار می گیرند، باعث اضافه شدن پیچیدگی غیرضروری به نرم افزار می شود. این اصل بخشی از فلسفه extreme programming می باشد.

S.O.L.I.D.

S.O.L.I.D مخفف پنج اصل طراحی در برنامه نویسی شیء گرا می باشد. برای مطالعه بیشتر، به این [لینک](#) مراجعه کنید.

- Single Responsibility Principle: هر موجودیت در نرم افزار (کلاس، تابع و ...) باید تنها یک مسئولیت داشته باشد.

- Open-Closed Principle: هر موجودیت در نرم افزار، باید برای توسعه باز باشند، اما برای اصلاح بسته باشند. به عبارت دیگر، این اصل بیان می کند که کد خود را باید به نحوی بنویسید تا بتوانید بدون تغییر کد موجود، ویژگی (feature) جدیدی اضافه کنید. این اصل از موقعیت هایی جلوگیری می کند که در آن، تغییر یکی از کلاس های شما، نیازمند تطبیق کلاس های دیگری است که به کلاس تغییر داده شده، وابسته هستند. همچنین بیان می کند که موجودیت های نرم افزار، پس از پیاده سازی، نباید تغییر کنند.

- Liskov Substitution Principle: وقتی از یک شیء از یک subclass، به جای یک شیء از superclass آن استفاده می کنید، همه چیز همچنان باید به درستی کار کند. به عبارت دیگر، اشیاء subclass باید دقیقاً مانند اشیاء superclass رفتار کنند.

- Interface Segregation Principle: کاربران نباید مجبور شوند به واسطه هایی که استفاده نمی کنند، وابسته باشند! مشابه Single Responsibility Principle، هدف Interface Segregation Principle کاهش side effect ها و دفعاتی است که نرم افزار نیاز به تغییر دارد.

- Dependency Inversion Principle: ماژول های سطح بالا، که منطق پیچیده ای را ارائه می دهند، باید به راحتی قابل استفاده مجدد باشند و تحت تأثیر تغییرات در ماژول های سطح پایین، که ویژگی های کاربردی را ارائه می دهند، قرار نگیرند. برای رسیدن به این اصل، باید abstraction ای را معرفی کنید که ماژول های سطح بالا و سطح پایین را از یکدیگر جدا می کند.

Git Commit

همان طور که در پروژه اول توضیح داده شد، کامیت ها اهمیت زیادی در توسعه پروژه های نرم افزاری دارند. در این پروژه نیز باید مواردی که در پروژه اول گفته شدند، رعایت شوند. رعایت این قسمت، بخشی از نمره شما را در این پروژه تعیین می کند.

نکات پایانی

- این تمرین در گروه‌های حداکثر دو نفره انجام می‌شود و کافی است که یکی از اعضای گروه Hash مربوط به آخرین کامیت پروژه را در سایت درس آپلود کند. در هنگام تحویل، پروژه روی این کامیت مورد ارزیابی قرار می‌گیرد.
- حتماً کاربر [IE-S03](#) را به پروژه خود اضافه کنید.
- ساختار صحیح و تمیزی کد برنامه، بخشی از نمره‌ی این فاز پروژه‌ی شما خواهد بود. بنابراین در طراحی ساختار برنامه دقت به خرج دهید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت مشاهده‌ی مشابهت بین کدهای دو گروه، از نمره هر دو گروه مطابق سیاستی که در کلاس گفته شده است کسر خواهد شد.
- سوالات خود را تا حد ممکن در گروه درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آنها بهره‌مند شوند. در صورتی که قصد مطرح کردن سوال خاص‌تری داشتید، از طریق ایمیل با طراحان این تمرین ارتباط برقرار کنید.