# SeatSelection 路由与端口-数据库映射审查

本文针对如下 5 个页面地址，梳理前端路由 → 后端接口 → 数据库表的完整链路，并解释路径格式来源：

* http://localhost:5200/seat/selection/1/10
* http://localhost:5200/seat/selection/2/11
* http://localhost:5200/seat/selection/4/13
* http://localhost:5200/seat/selection/5/14
* http://localhost:5200/seat/selection/6/15

这些地址均为“前端路由 URL”，对应座位选择页面，不是后端接口 URL。页面加载后会由前端调用后端 REST 接口查询座位区域与座位布局，并进行座位锁定/释放。

## 一、路径格式说明（为什么是 /seat/selection/:showId/:sessionId）

* 定义位置：ticket-frontend/src/router/index.ts
* {  
   path: '/seat',  
   component: Layout,  
   meta: { hidden: true },  
   children: [  
   {  
   path: 'selection/:showId/:sessionId',  
   component: () => import('@/views/seat/selection.vue'),  
   name: 'SeatSelection',  
   meta: { title: '选择座位', hidden: true }  
   }  
   ]  
  }
* 说明：
  + :showId 与 :sessionId 是“前端路由参数”，用于在页面内携带演出与场次上下文（页面逻辑会基于这两个参数去查询座位区域与布局）。
  + 上述示例 URL（如 /seat/selection/1/10）即为 showId=1、sessionId=10 的场景。

页面组件通过 vue-router 获取参数： - 文件：ticket-frontend/src/views/seat/selection.vue ts const route = useRoute() // 用法示例 const showId = parseInt(route.params.showId) const sessionId = parseInt(route.params.sessionId)

## 二、前端 → 后端接口调用链

* 页面组件：ticket-frontend/src/views/seat/selection.vue
  + 初始化时：
    1. 获取演出信息（getShowDetail(showId)）
    2. 获取可用座位区域（getSeatAreas(showType, showId)）
    3. 默认选择楼层与区域后，获取座位布局（getSeatLayout(areaId, showId, sessionId)）
  + 用户操作：
    - 选择座位 → lockSeats([seatId])
    - 取消/清空选择 → releaseSeats([seatId])
* 前端 API 定义：ticket-frontend/src/api/seat.ts
* // GET /seat/areas?showType=&showId=  
  export function getSeatAreas(showType: number, showId: number)  
    
  // GET /seat/layout/{areaId}?showId=&sessionId=  
  export function getSeatLayout(areaId: number, showId: number, sessionId: number)  
    
  // POST /seat/lock body: { seatIds: number[] }  
  export function lockSeats(seatIds: number[])  
    
  // POST /seat/release body: { seatIds: number[] }  
  export function releaseSeats(seatIds: number[])
* Axios 基址：ticket-frontend/src/utils/request.ts
  + baseURL = import.meta.env.VITE\_API\_BASE\_URL || '/api'
  + 若未配置 VITE\_API\_BASE\_URL，则前端调用实际发起到 /api/seat/...
* 网关路由转发：ticket-gateway/src/main/resources/application.yml
  + Path 匹配：/api/seat/\*\* 会被 Gateway 转发至 lb://ticket-show
* 后端控制器：ticket-show/src/main/java/com/ticketsystem/show/controller/SeatController.java
* @RestController  
  @RequestMapping("/api/seat")  
  public class SeatController {  
   @GetMapping("/areas") // GET /api/seat/areas  
   public Result<List<SeatAreaVO>> getSeatAreas(@RequestParam Integer showType, @RequestParam Long showId)  
    
   @GetMapping("/layout/{areaId}") // GET /api/seat/layout/{areaId}  
   public Result<SeatLayoutVO> getSeatLayout(@PathVariable Long areaId, @RequestParam Long showId, @RequestParam Long sessionId, HttpServletRequest request)  
    
   @PostMapping("/lock") // POST /api/seat/lock  
   public Result<Boolean> lockSeats(@RequestBody @Valid SeatLockRequest request, HttpServletRequest httpRequest)  
    
   @PostMapping("/release") // POST /api/seat/release  
   public Result<Boolean> releaseSeats(@RequestBody @Valid SeatLockRequest request, HttpServletRequest httpRequest)  
  }

小结： - 浏览器访问 http://localhost:5200/seat/selection/{showId}/{sessionId} 加载前端页面； - 前端随后调用 /api/seat/... 接口； - 网关将 /api/seat/... 转发至 ticket-show 服务； - SeatController 等后端代码执行业务并访问数据库。

## 三、后端服务与数据库表

* 服务层：ticket-show/src/main/java/com/ticketsystem/show/service/SeatService.java
* 实现：ticket-show/src/main/java/com/ticketsystem/show/service/impl/SeatServiceImpl.java
  + getAvailableAreas(showType, showId)
    - 通过 SeatMapper 从座位表查询可用区域（按 areaId 分组）
  + getSeatLayout(areaId, showId, sessionId, currentUserId)
    - 通过 SeatMapper 查询指定区域座位列表，组装为 SeatLayoutVO
    - 标记当前用户是否锁定（lockedByCurrentUser），供前端态展示
  + lockSeats(seatIds, userId) / releaseSeats(seatIds, userId)
    - 优先走 Redisson 原子锁与异步落库；若失败，回退到数据库直接锁/解锁（lockSeatsDatabaseOnly/releaseSeatsDatabaseOnly）
* 数据表与实体映射：
  + 表：t\_seat ←→ 实体：ticket-show/src/main/java/com/ticketsystem/show/entity/Seat.java
    - 关键字段：
      * id, area\_id, seat\_code, row\_num, seat\_num
      * status（0-维护，1-可用）
      * lock\_status（0-空闲，1-已锁定，2-已占用）
      * lock\_user\_id, lock\_time, lock\_expire\_time
      * is\_deleted（逻辑删除）
  + Mapper：ticket-show/src/main/java/com/ticketsystem/show/mapper/SeatMapper.java
    - 锁定：UPDATE t\_seat SET lock\_status=1, lock\_user\_id=?, lock\_time=NOW(), lock\_expire\_time=NOW()+5min WHERE id IN (...) AND lock\_status=0 AND status=1 AND is\_deleted=0
    - 释放：UPDATE t\_seat SET lock\_status=0, lock\_user\_id=NULL, lock\_time=NULL, lock\_expire\_time=NULL WHERE id IN (...) AND lock\_user\_id=? AND is\_deleted=0
    - 布局查询：SELECT \* FROM t\_seat WHERE area\_id=? AND is\_deleted=0 ORDER BY row\_num, seat\_num
    - 清理过期锁：UPDATE t\_seat SET lock\_status=0, ... WHERE lock\_status=1 AND lock\_expire\_time<NOW() AND is\_deleted=0

说明：座位区域本可由 t\_seat\_area（实体：SeatArea）承载描述信息，但当前 getAvailableAreas 直接以 t\_seat 分组构造区域 VO（简化实现）。

## 四、路径与数据库映射总结

* 你看到的 5 个 URL 是“前端路由”，通过 :showId/:sessionId 携带上下文；
* 页面加载后，前端根据这两个参数调用 /api/seat/areas、/api/seat/layout/{areaId} 等后端接口；
* 网关将这些接口转发到 ticket-show；
* ticket-show 访问数据库：核心表为 t\_seat，用于读取布局与进行座位锁定/释放；
* 锁定/释放持久化规则由 SeatMapper 中的 SQL 直接反映。

## 五、附：临时 SQL 校验（对齐页面行为）

1. 查询某区域的布局（供 /seat/layout/{areaId} 返回对齐）

SELECT id, area\_id, seat\_code, row\_num, seat\_num, status, lock\_status,  
 lock\_user\_id, lock\_time, lock\_expire\_time  
FROM t\_seat  
WHERE area\_id = :areaId AND is\_deleted = 0  
ORDER BY row\_num, seat\_num;

1. 校验锁定结果（页面点击“选择座位”后应更新）

SELECT id, lock\_status, lock\_user\_id, lock\_time, lock\_expire\_time  
FROM t\_seat  
WHERE id IN (:seatId1, :seatId2, ...);

* 预期：被当前用户锁定的行应为 lock\_status=1 且 lock\_user\_id=:currentUserId，lock\_expire\_time 约 5 分钟后

1. 清理过期锁定的影响

UPDATE t\_seat  
SET lock\_status = 0, lock\_user\_id = NULL, lock\_time = NULL, lock\_expire\_time = NULL  
WHERE lock\_status = 1 AND lock\_expire\_time < NOW() AND is\_deleted = 0;

* 说明：与 SeatMapper.clearExpiredLocks() 保持一致

## 六、排障要点

* 页面 404 或白屏：优先检查前端路由是否匹配 selection/:showId/:sessionId
* 调用失败 401：SeatController 需要 X-User-Id 头；前端已在 axios 拦截器中从用户状态附带该头
* 座位不可锁定：检查 t\_seat.status=1、lock\_status=0、is\_deleted=0 条件是否满足
* 锁定后未反映：注意前端 30s 同步与 4min 锁续期逻辑；也可手动刷新页面触发重新拉取布局