

dream-of-the-red-chamber.html

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>《红楼梦》人物关系动态图谱</title>
  <script src="https://d3js.org/d3.v7.min.js"></script>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: 'Microsoft YaHei', 'SimHei', sans-serif;
    }

    body {
      background: linear-gradient(135deg, #f5f7fa 0%, #e4edf5 100%);
      color: #333;
      min-height: 100vh;
      padding: 20px;
      overflow-x: hidden;
    }

    .container {
      max-width: 1400px;
      margin: 0 auto;
    }

    header {
      text-align: center;
      padding: 30px 0;
      background: rgba(188, 143, 143, 0.85);
      border-radius: 15px;
      margin-bottom: 25px;
      box-shadow: 0 8px 20px rgba(139, 69, 19, 0.2);
      position: relative;
      overflow: hidden;
    }

    header::before {
      content: "";
```

```
        position: absolute;
        top: 0;
        left: 0;
        right: 0;
        height: 5px;
        background: linear-gradient(90deg, #8b4513, #cd5c5c, #daa520, #8b4513);
    }

    h1 {
        font-size: 3.2rem;
        color: #8b0000;
        text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
        margin-bottom: 10px;
        letter-spacing: 3px;
    }

    .subtitle {
        font-size: 1.4rem;
        color: #5d4037;
        max-width: 800px;
        margin: 0 auto;
        line-height: 1.6;
    }

    .content {
        display: flex;
        gap: 25px;
        margin-bottom: 30px;
    }

    .controls {
        flex: 0 0 280px;
        background: rgba(255, 248, 225, 0.9);
        border-radius: 15px;
        padding: 25px;
        box-shadow: 0 8px 25px rgba(139, 69, 19, 0.15);
        height: fit-content;
    }

    .control-group {
        margin-bottom: 25px;
    }

    h2 {
```

```

        color: #8b4513;
        border-bottom: 2px dashed #cd5c5c;
        padding-bottom: 10px;
        margin-bottom: 18px;
        font-size: 1.8rem;
    }
}

h3 {
    color: #a0522d;
    margin: 15px 0 10px;
    font-size: 1.3rem;
}

.legend {
    display: flex;
    flex-wrap: wrap;
    gap: 12px;
    margin-top: 15px;
}

.legend-item {
    display: flex;
    align-items: center;
    font-size: 0.9rem;
}

.legend-color {
    width: 20px;
    height: 20px;
    border-radius: 50%;
    margin-right: 8px;
    border: 1px solid rgba(0,0,0,0.2);
}

.chart-container {
    flex: 1;
    background: rgba(255, 253, 245, 0.9);
    border-radius: 15px;
    padding: 20px;
    box-shadow: 0 8px 25px rgba(139, 69, 19, 0.15);
    min-height: 700px;
    position: relative;
}

```

```
#graph {  
    width: 100%;  
    height: 700px;  
}  
  
.search-box {  
    margin-bottom: 20px;  
    position: relative;  
}  
  
    input[type="text"] {  
        width: 100%;  
        padding: 12px 15px;  
        border: 2px solid #cd5c5c;  
        border-radius: 30px;  
        font-size: 1rem;  
        background: #fffaf0;  
        box-shadow: inset 0 2px 5px rgba(0,0,0,0.1);  
    }  
  
    input[type="text"]:focus {  
        outline: none;  
        border-color: #8b0000;  
        box-shadow: 0 0 8px rgba(139, 0, 0, 0.3);  
    }  
  
    button {  
        background: #8b4513;  
        color: white;  
        border: none;  
        padding: 10px 20px;  
        border-radius: 30px;  
        cursor: pointer;  
        font-size: 1rem;  
        transition: all 0.3s ease;  
        margin-top: 8px;  
        width: 100%;  
        font-weight: bold;  
    }  
  
    button:hover {  
        background: #a0522d;  
        transform: translateY(-2px);  
        box-shadow: 0 4px 8px rgba(139, 69, 19, 0.3);  
    }  
}
```

```
}  
  
.info-panel {  
    background: rgba(255, 248, 225, 0.95);  
    border-radius: 10px;  
    padding: 20px;  
    box-shadow: 0 5px 15px rgba(0,0,0,0.2);  
    position: absolute;  
    display: none;  
    max-width: 300px;  
    z-index: 100;  
    border: 2px solid #8b4513;  
}  
  
.info-panel h3 {  
    color: #8b0000;  
    margin-top: 0;  
}  
  
.info-panel p {  
    margin: 10px 0;  
    line-height: 1.5;  
}  
  
.instructions {  
    background: rgba(255, 248, 225, 0.9);  
    border-radius: 15px;  
    padding: 25px;  
    box-shadow: 0 8px 25px rgba(139, 69, 19, 0.15);  
    margin-top: 20px;  
}  
  
.instructions ul {  
    padding-left: 25px;  
    margin-top: 15px;  
}  
  
.instructions li {  
    margin-bottom: 12px;  
    line-height: 1.6;  
}  
  
.highlight {  
    background: #ffe4b5;
```

```
padding: 2px 5px;
border-radius: 4px;
font-weight: bold;
}

.character-card {
margin: 15px 0;
padding: 15px;
background: #fff;
border-radius: 10px;
border-left: 4px solid #cd5c5c;
box-shadow: 0 3px 8px rgba(0,0,0,0.1);
}

.character-card h4 {
color: #8b4513;
margin-bottom: 8px;
}

.footer {
text-align: center;
padding: 20px;
color: #5d4037;
font-size: 0.9rem;
margin-top: 20px;
}

@media (max-width: 1100px) {
.content {
flex-direction: column;
}

.controls {
flex: 1;
}
}

</style>
</head>
<body>
<div class="container">
<header>
<h1>《红楼梦》人物关系动态图谱</h1>
<p class="subtitle">探索中国古典文学巅峰之作中复杂的人物关系网络 - 拖
动、缩放、点击节点查看详情</p>
```

```
</header>

<div class="content">
  <div class="controls">
    <div class="control-group">
      <h2>图谱控制</h2>
      <div class="search-box">
        <input type="text" id="search" placeholder="搜索人物...">
      </div>
      <button id="reset">重置视图</button>
      <button id="layout">重新布局</button>
    </div>

    <div class="control-group">
      <h2>人物类别</h2>
      <div class="legend">
        <div class="legend-item">
          <div class="legend-color" style="background-color:
#e27d60;"></div>
          <span>贾府主子</span>
        </div>
        <div class="legend-item">
          <div class="legend-color" style="background-color:
#85cdcb;"></div>
          <span>贾府丫鬟</span>
        </div>
        <div class="legend-item">
          <div class="legend-color" style="background-color:
#e8a87c;"></div>
          <span>亲戚</span>
        </div>
        <div class="legend-item">
          <div class="legend-color" style="background-color:
#c38d9e;"></div>
          <span>仆人</span>
        </div>
        <div class="legend-item">
          <div class="legend-color" style="background-color:
#41b3a3;"></div>
          <span>四大家族</span>
        </div>
        <div class="legend-item">
          <div class="legend-color" style="background-color:
#8d8741;"></div>
```

```
<span>皇族</span>
</div>
</div>
</div>
</div>
<div class="control-group">
<h2>关系类型</h2>
<div class="legend">
<div class="legend-item">
<div class="legend-color" style="background-color: #999;
width: 30px; height: 2px; border-radius: 0;"></div>
<span>婚姻</span>
</div>
<div class="legend-item">
<div class="legend-color" style="background-color: #f00;
width: 30px; height: 2px; border-radius: 0;"></div>
<span>血缘</span>
</div>
<div class="legend-item">
<div class="legend-color" style="background-color: #00f;
width: 30px; height: 2px; border-radius: 0;"></div>
<span>主仆</span>
</div>
<div class="legend-item">
<div class="legend-color" style="background-color: #0a0;
width: 30px; height: 2px; border-radius: 0;"></div>
<span>情感</span>
</div>
</div>
</div>
</div>
<div class="control-group">
<h2>关键人物</h2>
<div class="character-card">
<h4>贾宝玉</h4>
<p>荣国府贾政与王夫人所生，通灵宝玉化身</p>
</div>
<div class="character-card">
<h4>林黛玉</h4>
<p>贾母外孙女，宝玉表妹，才华横溢</p>
</div>
<div class="character-card">
<h4>薛宝钗</h4>
<p>宝玉表姐，端庄大方，后与宝玉成婚</p>
```



```

        "情感": "#0a0"
    };

    // 创建 SVG 容器
    const width = document.getElementById('graph').clientWidth;
    const height = document.getElementById('graph').clientHeight;

    const svg = d3.select("#graph")
        .append("svg")
        .attr("width", width)
        .attr("height", height)
        .call(d3.zoom().on("zoom", (event) => {
            svgGroup.attr("transform", event.transform);
        })))
        .append("g");

    const svgGroup = svg.append("g");

    // 加载数据
    Promise.all([
        d3.json("characters.json"),
        d3.json("relations.json")
    ]).then(([characters, relations]) => {
        // 创建力导向图
        const simulation = d3.forceSimulation(characters)
            .force("link", d3.forceLink(relations).id(d => d.id).distance(d => 150 /
d.strength))
            .force("charge", d3.forceManyBody().strength(-300))
            .force("center", d3.forceCenter(width / 2, height / 2))
            .force("collision", d3.forceCollide().radius(50));

        // 绘制关系线
        const link = svgGroup.append("g")
            .attr("class", "links")
            .selectAll("line")
            .data(relations)
            .enter()
            .append("line")
            .attr("stroke", d => relationColors[d.type])
            .attr("stroke-width", d => d.strength * 3)
            .attr("stroke-opacity", 0.6);

        // 绘制节点
        const node = svgGroup.append("g")

```

```
.attr("class", "nodes")
.selectAll("circle")
.data(characters)
.enter()
.append("circle")
.attr("r", d => d.important ? 20 : 15)
.attr("fill", d => colorScale(d.group))
.attr("stroke", "#fff")
.attr("stroke-width", 2)
.call(d3.drag())
.on("start", dragstarted)
.on("drag", dragged)
.on("end", dragended))
.on("click", showInfo)
.on("dblclick", dblclickNode);
```

```
// 添加人物标签
const text = svgGroup.append("g")
.attr("class", "labels")
.selectAll("text")
.data(characters)
.enter()
.append("text")
.attr("dy", -20)
.attr("text-anchor", "middle")
.attr("font-size", 14)
.attr("fill", "#5d4037")
.attr("font-weight", "bold")
.text(d => d.name);
```

```
// 力导向图更新函数
simulation.on("tick", () => {
  link
    .attr("x1", d => d.source.x)
    .attr("y1", d => d.source.y)
    .attr("x2", d => d.target.x)
    .attr("y2", d => d.target.y);
```

```
  node
    .attr("cx", d => d.x)
    .attr("cy", d => d.y);
```

```
  text
    .attr("x", d => d.x)
```

```
        .attr("y", d => d.y);  
    });
```

```
    // 拖动函数  
    function dragstarted(event) {  
        if (!event.active) simulation.alphaTarget(0.3).restart();  
        event.subject.fx = event.subject.x;  
        event.subject.fy = event.subject.y;  
    }
```

```
    function dragged(event) {  
        event.subject.fx = event.x;  
        event.subject.fy = event.y;  
    }
```

```
    function dragended(event) {  
        if (!event.active) simulation.alphaTarget(0);  
        event.subject.fx = null;  
        event.subject.fy = null;  
    }
```

```
    // 显示人物信息  
    function showInfo(event, d) {  
        const panel = d3.select("#info-panel");  
        panel.style("display", "block")  
            .style("left", (event.pageX + 10) + "px")  
            .style("top", (event.pageY + 10) + "px")  
            .html(`  
                <h3>${d.name}</h3>  
                <p>${d.desc}</p>  
                <p><strong>类别:</strong> ${getGroupName(d.group)}</p>  
            `);  
    }
```

```
    // 双击节点聚焦  
    function dblclickNode(event, d) {  
        // 高亮当前节点  
        node.attr("opacity", 0.2);  
        d3.select(event.currentTarget).attr("opacity", 1);  
  
        // 高亮相关节点和关系  
        const connectedIds = new Set();  
        connectedIds.add(d.id);
```

```

        relations.forEach(rel => {
            if (rel.source.id === d.id || rel.target.id === d.id) {
                connectedIds.add(rel.source.id);
                connectedIds.add(rel.target.id);
            }
        });
    }

    node.filter(nodeData => connectedIds.has(nodeData.id))
        .attr("opacity", 1);

    link.attr("opacity", 0.1)
        .filter(linkData => linkData.source.id === d.id || linkData.target.id ===
d.id)
        .attr("opacity", 1);

    text.attr("opacity", 0)
        .filter(textData => connectedIds.has(textData.id))
        .attr("opacity", 1);
    }
}

```

```

// 获取分组名称
function getGroupName(groupId) {
    const groupNames = {
        1: "贾府主子",
        2: "贾府丫鬟",
        3: "亲戚",
        4: "仆人",
        5: "四大家族",
        6: "皇族"
    };

    return groupNames[groupId] || "其他";
}

```

```

// 搜索和交互功能
document.getElementById("search").addEventListener("input", function() {
    const searchTerm = this.value.trim().toLowerCase();
    if (!searchTerm) {
        node.attr("opacity", 1);
        text.attr("opacity", 1);
        return;
    }

    node.attr("opacity", d => d.name.toLowerCase().includes(searchTerm) ? 1 :
0.2);

```

```

        text.attr("opacity", d => d.name.toLowerCase().includes(searchTerm) ? 1 :
0);
    });

    document.getElementById("reset").addEventListener("click", function() {
        svgGroup.transition()
            .duration(750)
            .attr("transform", "translate(0,0) scale(1)");

        node.attr("opacity", 1);
        link.attr("opacity", 0.6);
        text.attr("opacity", 1);

        d3.select("#info-panel").style("display", "none");
    });

    document.getElementById("layout").addEventListener("click", function() {
        simulation.alpha(0.3).restart();
    });

    // 点击其他地方关闭信息面板
    document.addEventListener("click", function(event) {
        if (!event.target.matches("circle") && !event.target.matches("#info-panel"))
        {
            d3.select("#info-panel").style("display", "none");
        }
    });
</script>
</body>
</html>

```

characters.json

```

[
  { "id": "赵渡迷", "name": "赵渡迷", "group": 1, "desc": "贾宝玉的好朋友", "important":
false },
  { "id": "贾宝玉", "name": "贾宝玉", "group": 1, "desc": "荣国府贾政与王夫人所生，通灵宝
玉化身，大观园核心人物", "important": true },
  { "id": "林黛玉", "name": "林黛玉", "group": 3, "desc": "贾母外孙女，林如海与贾敏之女，
才华横溢，体弱多病", "important": true },
  { "id": "薛宝钗", "name": "薛宝钗", "group": 3, "desc": "薛姨妈之女，宝玉表姐，端庄大方，

```

```

后与宝玉成婚", "important": true },
    { "id": "贾母", "name": "贾母", "group": 1, "desc": "贾府最高长辈，史家小姐，宝玉祖母",
      "important": true },
    { "id": "贾政", "name": "贾政", "group": 1, "desc": "贾母次子，宝玉之父，荣国府当家人",
      "important": false },
    { "id": "王夫人", "name": "王夫人", "group": 1, "desc": "贾政之妻，宝玉之母，薛姨妈之姐",
      "important": false },
    { "id": "贾赦", "name": "贾赦", "group": 1, "desc": "贾母长子，贾琏之父", "important":
false },
    { "id": "贾琏", "name": "贾琏", "group": 1, "desc": "贾赦之子，王熙凤丈夫", "important":
false },
    { "id": "王熙凤", "name": "王熙凤", "group": 1, "desc": "贾琏之妻，王夫人侄女，荣国府管
家", "important": true },
    { "id": "贾元春", "name": "贾元春", "group": 1, "desc": "贾政长女，宝玉姐姐，皇妃",
      "important": false },
    { "id": "贾探春", "name": "贾探春", "group": 1, "desc": "贾政庶女，宝玉同父异母妹妹",
      "important": false },
    { "id": "贾珍", "name": "贾珍", "group": 1, "desc": "宁国府当家人，贾敬之子", "important":
false },
    { "id": "尤氏", "name": "尤氏", "group": 1, "desc": "贾珍续弦", "important": false },
    { "id": "贾蓉", "name": "贾蓉", "group": 1, "desc": "贾珍之子", "important": false },
    { "id": "秦可卿", "name": "秦可卿", "group": 1, "desc": "贾蓉之妻", "important": false },
    { "id": "袭人", "name": "袭人", "group": 2, "desc": "宝玉首席大丫鬟", "important": false },
    { "id": "晴雯", "name": "晴雯", "group": 2, "desc": "宝玉丫鬟，心灵手巧", "important":
false },
    { "id": "鸳鸯", "name": "鸳鸯", "group": 2, "desc": "贾母首席大丫鬟", "important": false },
    { "id": "皇帝", "name": "皇帝", "group": 6, "desc": "当朝天子", "important": false }
]

```

relations.json

```

[
    { "source": "贾宝玉", "target": "赵渡迷", "type": "情感", "strength": 1.0 },
    { "source": "贾宝玉", "target": "林黛玉", "type": "情感", "strength": 1.0 },
    { "source": "贾宝玉", "target": "薛宝钗", "type": "婚姻", "strength": 0.8 },
    { "source": "贾宝玉", "target": "贾母", "type": "血缘", "strength": 0.7 },
    { "source": "贾宝玉", "target": "贾政", "type": "血缘", "strength": 0.8 },
    { "source": "贾宝玉", "target": "王夫人", "type": "血缘", "strength": 0.8 },
    { "source": "贾宝玉", "target": "袭人", "type": "主仆", "strength": 0.9 },
    { "source": "贾宝玉", "target": "晴雯", "type": "主仆", "strength": 0.8 },
    { "source": "贾宝玉", "target": "贾元春", "type": "血缘", "strength": 0.7 },
    { "source": "贾宝玉", "target": "贾探春", "type": "血缘", "strength": 0.6 },

```

```
{ "source": "林黛玉", "target": "贾母", "type": "血缘", "strength": 0.8 },
{ "source": "林黛玉", "target": "贾政", "type": "血缘", "strength": 0.5 },
{ "source": "林黛玉", "target": "王夫人", "type": "血缘", "strength": 0.5 },
{ "source": "薛宝钗", "target": "王夫人", "type": "血缘", "strength": 0.6 },
{ "source": "贾母", "target": "贾政", "type": "血缘", "strength": 0.8 },
{ "source": "贾母", "target": "贾赦", "type": "血缘", "strength": 0.8 },
{ "source": "贾母", "target": "鸳鸯", "type": "主仆", "strength": 0.9 },
{ "source": "贾政", "target": "王夫人", "type": "婚姻", "strength": 0.9 },
{ "source": "贾政", "target": "贾元春", "type": "血缘", "strength": 0.8 },
{ "source": "贾政", "target": "贾探春", "type": "血缘", "strength": 0.8 },
{ "source": "贾赦", "target": "贾琏", "type": "血缘", "strength": 0.8 },
{ "source": "贾琏", "target": "王熙凤", "type": "婚姻", "strength": 0.9 },
{ "source": "王熙凤", "target": "王夫人", "type": "血缘", "strength": 0.6 },
{ "source": "贾珍", "target": "尤氏", "type": "婚姻", "strength": 0.7 },
{ "source": "贾珍", "target": "贾蓉", "type": "血缘", "strength": 0.9 },
{ "source": "贾蓉", "target": "秦可卿", "type": "婚姻", "strength": 0.8 },
{ "source": "皇帝", "target": "贾元春", "type": "婚姻", "strength": 0.9 }
]
```

以上红楼梦人物关系图谱，人物数据集和关系数据集各分别为一个外部独立文件，但任务再增加是，管理维护还是不方便，我想把两个数据集集中到 MySQL 数据库，并把关系图谱独立建站域名为 `infographdorch.dododot.work`，请帮我改写代码，并指导我部署

DeepSeek 思考

我们计划将数据从 JSON 文件迁移到 MySQL 数据库，并通过 PHP 提供 API 接口给前端调用。
步骤：

1. 设计数据库表结构（两个表：characters 和 relations）
2. 将现有的 JSON 数据导入 MySQL
3. 编写 PHP API，提供两个端点：获取所有人物，获取所有关系
4. 修改前端代码，从 PHP API 获取数据，而不是直接加载 JSON 文件
5. 部署到服务器，并设置域名为 `infographdorch.dododot.work`

详细步骤：

1. 数据库设计：

表 1: characters

列: id (varchar, primary key), name (varchar), group (int), desc (text), important (tinyint, 0 或 1)

表 2: relations

列: id (int, auto_increment, primary key), source (varchar, foreign key to characters.id), target (varchar, foreign key to characters.id), type (varchar), strength (float)

注意：由于关系可能重复，我们使用自增 ID 作为主键。

2. 导入数据：

将 characters.json 和 relations.json 的数据分别导入到对应的表中。

3. PHP API：

创建两个文件：

- `get_characters.php`: 返回所有人物数据, JSON 格式

- `get_relations.php`: 返回所有关系数据, JSON 格式

示例代码 (`get_characters.php`):

```
<?php
header('Content-Type: application/json');
// 数据库连接配置
$host = 'localhost';
$dbname = 'your_database_name';
$username = 'your_username';
$password = 'your_password';
try {
    $conn = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username,
    $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->query('SELECT * FROM characters');
    $characters = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($characters);
} catch(PDOException $e) {
    echo json_encode(['error' => $e->getMessage()]);
}
?>
```

同理, `get_relations.php` 类似。

4. 修改前端代码:

在原来的 HTML 文件中, 修改 JavaScript 部分, 使用 `d3.json` 从新的 API 获取数据。

将:

```
Promise.all([
    d3.json("characters.json"),
    d3.json("relations.json")
]).then(([characters, relations]) => { ... });
```

改为:

```
Promise.all([
    d3.json("http://infographdrorch.dododot.work/get_characters.php"),
    d3.json("http://infographdrorch.dododot.work/get_relations.php")
]).then(([characters, relations]) => { ... });
```

注意: 如果前端和后端在同一域名下, 这样写没问题。如果跨域, 需要设置 CORS。

在 PHP 文件中添加 CORS 头 (如果需要):

```
header('Access-Control-Allow-Origin: *');
```

5. 部署:

将前端 HTML、CSS、JS 文件部署到服务器, 同时部署 PHP 接口和数据库。

具体步骤:

- 购买/准备服务器, 安装 LAMP (Linux, Apache, MySQL, PHP) 环境。
- 配置域名 `infographdrorch.dododot.work` 指向服务器 IP。
- 在服务器上创建数据库, 导入数据。
- 将前端 HTML 文件 (例如 `index.html`) 放到网站根目录 (如 `/var/www/html/`)。

- 将两个 PHP 接口文件放到同一目录下。
- 修改 PHP 文件中的数据库连接信息。

考虑到安全性，在实际部署时，建议将数据库连接信息放在配置文件中，并避免将敏感信息暴露。

由于我们改变了数据加载方式，需要确保在数据加载完成后才创建力导向图。同时，由于数据现在来自网络请求，可能会有延迟，可以添加加载提示。

但是，我们这里只做基本迁移，所以先不考虑加载提示。

代码改写：

由于原 HTML 文件中的 JavaScript 部分使用了两个 JSON 文件的本地路径，现在改为远程 API。另外，由于 relations.json 中的 source 和 target 是字符串（人物 ID），而数据库中存储的也是字符串，所以可以直接使用。

但是，注意在力导向图中，我们使用 `d.forceLink(relations).id(d => d.id)` 来指定节点 id，所以需要确保 relations 数组中的 source 和 target 与字符节点 id 匹配。

由于我们数据库中的关系表存储的 source 和 target 就是人物 id，所以没有问题。

修改后的 HTML 代码中，JavaScript 部分的数据加载改为：

```
Promise.all([
  d3.json("get_characters.php"),
  d3.json("get_relations.php")
]).then(([characters, relations]) => {
  // 在这里创建图表
});
```

注意：这里使用了相对路径，因为 PHP 文件和 HTML 文件在同一目录下。如果不在同一目录，需要调整路径。

另外，考虑到部署后的域名，如果前端文件在域名的根目录，那么这样写是没问题的。

我们还需要注意：在本地开发时，如果直接打开 HTML 文件，由于跨域问题，可能无法加载数据。所以建议在服务器环境下测试。

最后，我们将 HTML 文件中的 script 部分修改为以上方式。

由于原代码较长，我们只修改数据加载部分，其他部分保持不变。

修改后的 HTML 文件（dream-of-the-red-chamber.html）中 script 部分的数据加载代码：

原代码：

```
Promise.all([
  d3.json("characters.json"),
  d3.json("relations.json")
]).then(([characters, relations]) => {
```

修改为：

```
Promise.all([
  d3.json("get_characters.php"),
  d3.json("get_relations.php")
]).then(([characters, relations]) => {
```

同时，将原来的两个 JSON 文件移除，因为我们使用数据库了。

部署步骤总结：

1. 在服务器上创建数据库，导入数据（使用 SQL 文件或命令行导入）。
2. 配置 Apache 服务器，设置域名指向。
3. 将前端文件（HTML、CSS、JS）和 PHP 文件上传到服务器网站目录。

4. 修改 PHP 文件中的数据库连接信息。

5. 测试访问：<http://infographdrorch.dododot.work>

注意：如果使用 HTTPS，需要配置 SSL 证书。

由于本指导重点在代码迁移，部署细节（如配置虚拟主机、安装 SSL 证书）不在范围内，但可以参考常规的 LAMP 部署教程。

下面提供修改后的完整 HTML 文件（只修改了数据加载的 URL）和两个 PHP 文件的代码。

由于原 HTML 文件很长，我们只展示修改后的部分，其他部分保持不变。

修改后的 HTML 文件（dream-of-the-red-chamber.html）的 script 部分：

原代码：

```
Promise.all([
  d3.json("characters.json"),
  d3.json("relations.json")
]).then([characters, relations]) => {
```

修改为：

```
Promise.all([
  d3.json("get_characters.php"),
  d3.json("get_relations.php")
]).then([characters, relations]) => {
```

同时，在 HTML 文件所在目录，创建两个 PHP 文件：

文件 1：get_characters.php

```
```php
<?php
header('Access-Control-Allow-Origin: *');
header('Content-Type: application/json');
$host = 'localhost';
$dbname = 'your_database_name';
$username = 'your_database_username';
$password = 'your_database_password';
try {
 $conn = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username,
 $password);
 $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
 $stmt = $conn->query('SELECT * FROM characters');
 $characters = $stmt->fetchAll(PDO::FETCH_ASSOC);
 echo json_encode($characters, JSON_UNESCAPED_UNICODE);
} catch(PDOException $e) {
 echo json_encode(['error' => $e->getMessage()]);
}
?>
```
```

文件 2：get_relations.php

```
```php
<?php
header('Access-Control-Allow-Origin: *');
```

```

header('Content-Type: application/json');
$host = 'localhost';
$dbname = 'your_database_name';
$username = 'your_database_username';
$password = 'your_database_password';
try {
$conn = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username,
$password);
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$stmt = $conn->query('SELECT source, target, type, strength FROM relations');
$relations = $stmt->fetchAll(PDO::FETCH_ASSOC);
echo json_encode($relations, JSON_UNESCAPED_UNICODE);
} catch(PDOException $e) {
echo json_encode(['error' => $e->getMessage()]);
}
?>
...

```

注意：请替换数据库连接信息。

最后，将这三个文件（HTML 和两个 PHP）以及必要的资源（如图片等）上传到服务器，并确保服务器已正确配置。

这样，当用户访问 `infographdrorch.dododot.work` 时，就会加载 HTML 页面，然后通过 AJAX 请求从数据库获取数据并渲染关系图谱。

如果遇到跨域问题，确保两个 PHP 文件设置了允许跨域的头（Access-Control-Allow-Origin）。这里设置为\*，表示允许任何域名访问，在生产环境中可以根据需要设置。

由于我们修改了数据来源，请确保数据库中的字段名和原 JSON 文件中的字段名一致（特别是 `group` 是 MySQL 关键字，需要用反引号括起来，或者在查询时使用别名，但原 JSON 中有 `group`，我们数据库表设计也用了 `group`，所以查询时要用 ``group``）。

因此，在 `get_characters.php` 中，SQL 语句应该写成：

```
SELECT `id`, `name`, `group`, `desc`, `important` FROM characters
```

或者在创建表时避免使用关键字，但为了和原 JSON 一致，我们使用反引号。

修改后的 `get_characters.php`：

```

```php
$stmt = $conn->query('SELECT `id`, `name`, `group`, `desc`, `important` FROM characters');
...

```

但是，在创建表时，如果字段名是关键字，应该用反引号，但在查询时同样要注意。

我们建议在创建表时避免使用关键字，可以将 `group` 改为 `group_id`，但这样需要修改前端代码。为了减少改动，我们使用反引号。

所以，最终的两个 PHP 文件如下：

`get_characters.php`：

```

```php
<?php
header('Access-Control-Allow-Origin: *');
header('Content-Type: application/json');
$host = 'localhost';

```

```

$dbname = 'your_database_name';
$username = 'your_database_username';
$password = 'your_database_password';
try {
$conn = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8", $username,
$password);
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$stmt = $conn->query('SELECT `id`, `name`, `group`, `desc`, `important` FROM characters');
$characters = $stmt->fetchAll(PDO::FETCH_ASSOC);
echo json_encode($characters, JSON_UNESCAPED_UNICODE);
} catch(PDOException $e) {
echo json_encode(['error' => $e->getMessage()]);
}
?>
```

```

`get_relations.php` 保持不变。

这样，我们就完成了从数据库加载数据的前后端分离架构。

部署后，如果出现数据无法加载，请检查：

- 数据库连接信息是否正确
- 数据库表中字段名是否和查询语句中的一致（包括反引号）
- 网络请求是否成功（通过浏览器开发者工具查看）
- 数据库表中的数据是否正确导入

按照以上步骤，即可完成迁移和部署。

DeepSeek 回复

红楼梦人物关系图谱 - 数据库集成版

我将为您重构代码，将数据迁移到 MySQL 数据库，并准备部署到指定域名。以下是完整的解决方案：

解决方案概述

1. 创建 MySQL 数据库表结构
2. 将 JSON 数据导入数据库
3. 开发 PHP API 接口
4. 重构前端代码使用 API 获取数据
5. 部署到服务器

数据库设计 (MySQL)

```

```sql
CREATE DATABASE red_chamber;
USE red_chamber;

```

-- 人物表

```
CREATE TABLE characters (
 id VARCHAR(20) PRIMARY KEY,
 name VARCHAR(20) NOT NULL,
 `group` TINYINT NOT NULL,
 `desc` TEXT NOT NULL,
 important BOOLEAN NOT NULL
);
```

-- 关系表

```
CREATE TABLE relations (
 id INT AUTO_INCREMENT PRIMARY KEY,
 source VARCHAR(20) NOT NULL,
 target VARCHAR(20) NOT NULL,
 type VARCHAR(10) NOT NULL,
 strength FLOAT NOT NULL,
 FOREIGN KEY (source) REFERENCES characters(id),
 FOREIGN KEY (target) REFERENCES characters(id)
);
...
```

## PHP API 代码

### api.php (后端接口)

```
```php  
<?php  
header('Content-Type: application/json');  
header('Access-Control-Allow-Origin: *');  
  
$config = [  
    'host' => 'localhost',  
    'dbname' => 'red_chamber',  
    'username' => 'your_db_username',  
    'password' => 'your_db_password'  
];  
  
try {  
    $pdo = new PDO(  
        "mysql:host={$config['host']};dbname={$config['dbname']};charset=utf8",  
        $config['username'],  
        $config['password']  
    );
```

```

$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

// 获取所有人物
$characters = $pdo->query("SELECT * FROM characters")->fetchAll(PDO::FETCH_ASSOC);

// 获取所有关系
$relations = $pdo->query("SELECT source, target, type, strength FROM
relations")->fetchAll(PDO::FETCH_ASSOC);

echo json_encode([
    'characters' => $characters,
    'relations' => $relations
]);

} catch (PDOException $e) {
    echo json_encode(['error' => 'Database error: ' . $e->getMessage()]);
} catch (Exception $e) {
    echo json_encode(['error' => 'General error: ' . $e->getMessage()]);
}
?>
...

```

完整前端代码 (HTML)

```

<<<html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>《红楼梦》人物关系动态图谱 - 数据库版</title>
    <script src="https://d3js.org/d3.v7.min.js"></script>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Microsoft YaHei', 'SimHei', sans-serif;
        }

        body {
            background: linear-gradient(135deg, #f5f7fa 0%, #e4edf5 100%);
            color: #333;
            min-height: 100vh;

```

```
padding: 20px;
overflow-x: hidden;
}
```

```
.container {
  max-width: 1400px;
  margin: 0 auto;
}
```

```
header {
  text-align: center;
  padding: 30px 0;
  background: rgba(188, 143, 143, 0.85);
  border-radius: 15px;
  margin-bottom: 25px;
  box-shadow: 0 8px 20px rgba(139, 69, 19, 0.2);
  position: relative;
  overflow: hidden;
}
```

```
header::before {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  height: 5px;
  background: linear-gradient(90deg, #8b4513, #cd5c5c, #daa520, #8b4513);
}
```

```
h1 {
  font-size: 3.2rem;
  color: #8b0000;
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);
  margin-bottom: 10px;
  letter-spacing: 3px;
}
```

```
.subtitle {
  font-size: 1.4rem;
  color: #5d4037;
  max-width: 800px;
  margin: 0 auto;
  line-height: 1.6;
```



```
}

.content {
  display: flex;
  gap: 25px;
  margin-bottom: 30px;
}

.controls {
  flex: 0 0 280px;
  background: rgba(255, 248, 225, 0.9);
  border-radius: 15px;
  padding: 25px;
  box-shadow: 0 8px 25px rgba(139, 69, 19, 0.15);
  height: fit-content;
}

.control-group {
  margin-bottom: 25px;
}

h2 {
  color: #8b4513;
  border-bottom: 2px dashed #cd5c5c;
  padding-bottom: 10px;
  margin-bottom: 18px;
  font-size: 1.8rem;
}

h3 {
  color: #a0522d;
  margin: 15px 0 10px;
  font-size: 1.3rem;
}

.legend {
  display: flex;
  flex-wrap: wrap;
  gap: 12px;
  margin-top: 15px;
}

.legend-item {
  display: flex;
```

```

        align-items: center;
        font-size: 0.9rem;
    }

    .legend-color {
        width: 20px;
        height: 20px;
        border-radius: 50%;
        margin-right: 8px;
        border: 1px solid rgba(0,0,0,0.2);
    }

    .chart-container {
        flex: 1;
        background: rgba(255, 253, 245, 0.9);
        border-radius: 15px;
        padding: 20px;
        box-shadow: 0 8px 25px rgba(139, 69, 19, 0.15);
        min-height: 700px;
        position: relative;
    }

    #graph {
        width: 100%;
        height: 700px;
    }

    .search-box {
        margin-bottom: 20px;
        position: relative;
    }

    input[type="text"] {
        width: 100%;
        padding: 12px 15px;
        border: 2px solid #cd5c5c;
        border-radius: 30px;
        font-size: 1rem;
        background: #fffaf0;
        box-shadow: inset 0 2px 5px rgba(0,0,0,0.1);
    }

    input[type="text"]:focus {
        outline: none;
    }

```

```
        border-color: #8b0000;
        box-shadow: 0 0 8px rgba(139, 0, 0, 0.3);
    }

    button {
        background: #8b4513;
        color: white;
        border: none;
        padding: 10px 20px;
        border-radius: 30px;
        cursor: pointer;
        font-size: 1rem;
        transition: all 0.3s ease;
        margin-top: 8px;
        width: 100%;
        font-weight: bold;
    }

    button:hover {
        background: #a0522d;
        transform: translateY(-2px);
        box-shadow: 0 4px 8px rgba(139, 69, 19, 0.3);
    }

    .info-panel {
        background: rgba(255, 248, 225, 0.95);
        border-radius: 10px;
        padding: 20px;
        box-shadow: 0 5px 15px rgba(0,0,0,0.2);
        position: absolute;
        display: none;
        max-width: 300px;
        z-index: 100;
        border: 2px solid #8b4513;
    }

    .info-panel h3 {
        color: #8b0000;
        margin-top: 0;
    }

    .info-panel p {
        margin: 10px 0;
        line-height: 1.5;
```

```
}

.instructions {
    background: rgba(255, 248, 225, 0.9);
    border-radius: 15px;
    padding: 25px;
    box-shadow: 0 8px 25px rgba(139, 69, 19, 0.15);
    margin-top: 20px;
}

.instructions ul {
    padding-left: 25px;
    margin-top: 15px;
}

.instructions li {
    margin-bottom: 12px;
    line-height: 1.6;
}

.highlight {
    background: #ffe4b5;
    padding: 2px 5px;
    border-radius: 4px;
    font-weight: bold;
}

.character-card {
    margin: 15px 0;
    padding: 15px;
    background: #fff;
    border-radius: 10px;
    border-left: 4px solid #cd5c5c;
    box-shadow: 0 3px 8px rgba(0,0,0,0.1);
}

.character-card h4 {
    color: #8b4513;
    margin-bottom: 8px;
}

.footer {
    text-align: center;
    padding: 20px;
```

```

        color: #5d4037;
        font-size: 0.9rem;
        margin-top: 20px;
    }

    @media (max-width: 1100px) {
        .content {
            flex-direction: column;
        }

        .controls {
            flex: 1;
        }
    }

    /* 加载指示器 */
    .loader {
        position: absolute;
        top: 50%;
        left: 50%;
        transform: translate(-50%, -50%);
        z-index: 10;
        display: flex;
        flex-direction: column;
        align-items: center;
    }

    .spinner {
        width: 50px;
        height: 50px;
        border: 5px solid rgba(139, 69, 19, 0.2);
        border-top: 5px solid #8b4513;
        border-radius: 50%;
        animation: spin 1s linear infinite;
    }

    .loader-text {
        margin-top: 15px;
        color: #8b4513;
        font-size: 1.2rem;
        font-weight: bold;
    }

    @keyframes spin {

```

```

0% { transform: rotate(0deg); }
100% { transform: rotate(360deg); }
}
</style>
</head>
<body>
  <div class="container">
    <header>
      <h1>《红楼梦》人物关系动态图谱</h1>
      <p class="subtitle">探索中国古典文学巅峰之作中复杂的人物关系网络 - 数据库驱动版</p>
    </header>

    <div class="content">
      <div class="controls">
        <div class="control-group">
          <h2>图谱控制</h2>
          <div class="search-box">
            <input type="text" id="search" placeholder="搜索人物...">
          </div>
          <button id="reset">重置视图</button>
          <button id="layout">重新布局</button>
        </div>

        <div class="control-group">
          <h2>人物类别</h2>
          <div class="legend">
            <div class="legend-item">
              <div class="legend-color" style="background-color:
#e27d60;"></div>
              <span>贾府主子</span>
            </div>
            <div class="legend-item">
              <div class="legend-color" style="background-color:
#85cdcb;"></div>
              <span>贾府丫鬟</span>
            </div>
            <div class="legend-item">
              <div class="legend-color" style="background-color:
#e8a87c;"></div>
              <span>亲戚</span>
            </div>
            <div class="legend-item">
              <div class="legend-color" style="background-color:

```

#c38d9e;"></div>

仆人

</div>

<div class="legend-item">

<div class="legend-color" style="background-color:

#41b3a3;"></div>

四大家族

</div>

<div class="legend-item">

<div class="legend-color" style="background-color:

#8d8741;"></div>

皇族

</div>

</div>

</div>

<div class="control-group">

<h2>关系类型</h2>

<div class="legend">

<div class="legend-item">

<div class="legend-color" style="background-color: #999;

width: 30px; height: 2px; border-radius: 0;"></div>

婚姻

</div>

<div class="legend-item">

<div class="legend-color" style="background-color: #f00;

width: 30px; height: 2px; border-radius: 0;"></div>

血缘

</div>

<div class="legend-item">

<div class="legend-color" style="background-color: #00f;

width: 30px; height: 2px; border-radius: 0;"></div>

主仆

</div>

<div class="legend-item">

<div class="legend-color" style="background-color: #0a0;

width: 30px; height: 2px; border-radius: 0;"></div>

情感

</div>

</div>

</div>

<div class="control-group">

<h2>关键人物</h2>

```
<div class="character-card">
  <h4>贾宝玉</h4>
  <p>荣国府贾政与王夫人所生，通灵宝玉化身</p>
</div>
<div class="character-card">
  <h4>林黛玉</h4>
  <p>贾母外孙女，宝玉表妹，才华横溢</p>
</div>
<div class="character-card">
  <h4>薛宝钗</h4>
  <p>宝玉表姐，端庄大方，后与宝玉成婚</p>
</div>
</div>
</div>
```

```
<div class="chart-container">
  <div id="graph">
    <div class="loader">
      <div class="spinner"></div>
      <div class="loader-text">加载人物关系数据中...</div>
    </div>
  </div>
  <div class="info-panel" id="info-panel"></div>
</div>
</div>
```

```
<div class="instructions">
  <h2>使用说明</h2>
  <ul>
    <li><span class="highlight">拖动节点</span> - 可以移动人物位置，重新
布局关系</li>
    <li><span class="highlight">鼠标滚轮</span> - 缩放图谱以查看细节或全
局</li>
    <li><span class="highlight">点击节点</span> - 查看人物详细信息</li>
    <li><span class="highlight">双击节点</span> - 聚焦该人物并隐藏不直接
相关的人物</li>
    <li><span class="highlight">搜索框</span> - 输入人物姓名快速定位</li>
    <li><span class="highlight">重置视图</span> - 恢复初始缩放和位置</li>
    <li><span class="highlight">重新布局</span> - 重新计算人物关系布局
</li>
    <li><span class="highlight">数据库版</span> - 数据从 MySQL 数据库动态
加载</li>
  </ul>
</div>
```



```
<div class="footer">
  <p>数据来源: MySQL 数据库 | 可视化设计: 古典文学关系图谱 | © 2023 红楼梦研究</p>
```

```
  <p>访问域名: infographdrorch.dododot.work</p>
```

```
</div>
```

```
</div>
```

```
<script>
```

```
  // 颜色映射
```

```
  const colorScale = d3.scaleOrdinal()
```

```
    .domain([1, 2, 3, 4, 5, 6])
```

```
    .range(["#e27d60", "#85cdcb", "#e8a87c", "#c38d9e", "#41b3a3", "#8d8741"]);
```

```
  // 关系类型颜色
```

```
  const relationColors = {
```

```
    "血缘": "#f00",
```

```
    "婚姻": "#999",
```

```
    "主仆": "#00f",
```

```
    "情感": "#0a0"
```

```
  };
```

```
  // 创建 SVG 容器
```

```
  const width = document.getElementById('graph').clientWidth;
```

```
  const height = document.getElementById('graph').clientHeight;
```

```
  const svg = d3.select("#graph")
```

```
    .append("svg")
```

```
    .attr("width", width)
```

```
    .attr("height", height)
```

```
    .call(d3.zoom().on("zoom", (event) => {
```

```
      svgGroup.attr("transform", event.transform);
```

```
    }));
```

```
    .append("g");
```

```
  const svgGroup = svg.append("g");
```

```
  // 显示加载指示器
```

```
  const loader = d3.select(".loader");
```

```
  // 从 API 加载数据
```

```
  d3.json("api.php")
```

```
    .then(data => {
```

```
      // 隐藏加载指示器
```

```

loader.style("display", "none");

const characters = data.characters;
const relations = data.relations;

// 创建力导向图
const simulation = d3.forceSimulation(characters)
  .force("link", d3.forceLink(relations).id(d => d.id).distance(d => 150 /
d.strength))
  .force("charge", d3.forceManyBody().strength(-300))
  .force("center", d3.forceCenter(width / 2, height / 2))
  .force("collision", d3.forceCollide().radius(50));

// 绘制关系线
const link = svgGroup.append("g")
  .attr("class", "links")
  .selectAll("line")
  .data(relations)
  .enter()
  .append("line")
  .attr("stroke", d => relationColors[d.type])
  .attr("stroke-width", d => d.strength * 3)
  .attr("stroke-opacity", 0.6);

// 绘制节点
const node = svgGroup.append("g")
  .attr("class", "nodes")
  .selectAll("circle")
  .data(characters)
  .enter()
  .append("circle")
  .attr("r", d => d.important ? 20 : 15)
  .attr("fill", d => colorScale(d.group))
  .attr("stroke", "#fff")
  .attr("stroke-width", 2)
  .call(d3.drag()
    .on("start", dragstarted)
    .on("drag", dragged)
    .on("end", dragended))
  .on("click", showInfo)
  .on("dblclick", dblclickNode);

// 添加人物标签
const text = svgGroup.append("g")

```

```

.attr("class", "labels")
.selectAll("text")
.data(characters)
.enter()
.append("text")
.attr("dy", -20)
.attr("text-anchor", "middle")
.attr("font-size", 14)
.attr("fill", "#5d4037")
.attr("font-weight", "bold")
.text(d => d.name);

```

// 力导向图更新函数

```

simulation.on("tick", () => {
  link
    .attr("x1", d => d.source.x)
    .attr("y1", d => d.source.y)
    .attr("x2", d => d.target.x)
    .attr("y2", d => d.target.y);

  node
    .attr("cx", d => d.x)
    .attr("cy", d => d.y);

  text
    .attr("x", d => d.x)
    .attr("y", d => d.y);
});

```

// 拖动函数

```

function dragstarted(event) {
  if (!event.active) simulation.alphaTarget(0.3).restart();
  event.subject.fx = event.subject.x;
  event.subject.fy = event.subject.y;
}

function dragged(event) {
  event.subject.fx = event.x;
  event.subject.fy = event.y;
}

function dragended(event) {
  if (!event.active) simulation.alphaTarget(0);
  event.subject.fx = null;

```

```

        event.subject.fy = null;
    }

    // 显示人物信息
    function showInfo(event, d) {
        const panel = d3.select("#info-panel");
        panel.style("display", "block")
            .style("left", (event.pageX + 10) + "px")
            .style("top", (event.pageY + 10) + "px")
            .html(`
                <h3>${d.name}</h3>
                <p>${d.desc}</p>
                <p><strong>类别:</strong> ${getGroupName(d.group)}</p>
            `);
    }

    // 双击节点聚焦
    function dblclickNode(event, d) {
        // 高亮当前节点
        node.attr("opacity", 0.2);
        d3.select(event.currentTarget).attr("opacity", 1);

        // 高亮相关节点和关系
        const connectedIds = new Set();
        connectedIds.add(d.id);

        relations.forEach(rel => {
            if (rel.source.id === d.id || rel.target.id === d.id) {
                connectedIds.add(rel.source.id);
                connectedIds.add(rel.target.id);
            }
        });

        node.filter(nodeData => connectedIds.has(nodeData.id))
            .attr("opacity", 1);

        link.attr("opacity", 0.1)
            .filter(linkData => linkData.source.id === d.id || linkData.target.id
=== d.id)
            .attr("opacity", 1);

        text.attr("opacity", 0)
            .filter(textData => connectedIds.has(textData.id))
            .attr("opacity", 1);
    }

```

```
}
```

```
// 获取分组名称
```

```
function getGroupName(groupId) {  
  const groupNames = {  
    1: "贾府主子",  
    2: "贾府丫鬟",  
    3: "亲戚",  
    4: "仆人",  
    5: "四大家族",  
    6: "皇族"  
  };  
  return groupNames[groupId] || "其他";  
}
```

```
// 搜索和交互功能
```

```
document.getElementById("search").addEventListener("input", function() {  
  const searchTerm = this.value.trim().toLowerCase();  
  if (!searchTerm) {  
    node.attr("opacity", 1);  
    text.attr("opacity", 1);  
    return;  
  }  
}
```

```
  node.attr("opacity", d => d.name.toLowerCase().includes(searchTerm) ?
```

```
    1 : 0.2);
```

```
  text.attr("opacity", d => d.name.toLowerCase().includes(searchTerm) ?
```

```
    1 : 0);
```

```
});
```

```
document.getElementById("reset").addEventListener("click", function() {  
  svgGroup.transition()  
    .duration(750)  
    .attr("transform", "translate(0,0) scale(1)");
```

```
  node.attr("opacity", 1);  
  link.attr("opacity", 0.6);  
  text.attr("opacity", 1);
```

```
  d3.select("#info-panel").style("display", "none");
```

```
});
```

```
document.getElementById("layout").addEventListener("click", function() {  
  simulation.alpha(0.3).restart();
```

```

    });

    // 点击其他地方关闭信息面板
    document.addEventListener("click", function(event) {
        if (!event.target.matches("#circle")
        && !event.target.matches("#info-panel")) {
            d3.select("#info-panel").style("display", "none");
        }
    });
})
.catch(error => {
    console.error("Error loading data:", error);
    loader.select(".loader-text").text("数据加载失败，请刷新重试");
});
</script>
</body>
</html>
...

```

部署指南

服务器要求

- Linux 服务器（推荐 Ubuntu 20.04+）
- Apache/Nginx Web 服务器
- PHP 7.4+
- MySQL 5.7+

部署步骤

1. **配置域名**

```

``bash
sudo nano /etc/apache2/sites-available/infographdrorch.conf
...

```

添加以下内容：

```

``apache
<VirtualHost *:80>
    ServerName infographdrorch.dododot.work
    DocumentRoot /var/www/infographdrorch
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
...

```

启用站点：

```

``bash

```

```
sudo a2ensite infographdrorch.conf
sudo systemctl reload apache2
...
```

2. **创建项目目录**

```
``bash
sudo mkdir -p /var/www/infographdrorch
sudo chown -R $USER:$USER /var/www/infographdrorch
...
```

3. **上传文件**

将以下文件上传到`/var/www/infographdrorch`:

- `index.html` (前端页面)
- `api.php` (PHP API 接口)

4. **配置数据库**

```
``sql
CREATE USER 'redchamber'@'localhost' IDENTIFIED BY 'strong_password';
CREATE DATABASE red_chamber;
GRANT ALL PRIVILEGES ON red_chamber.* TO 'redchamber'@'localhost';
FLUSH PRIVILEGES;
...
```

5. **导入数据**

```
``bash
mysql -u redchamber -p red_chamber < characters.sql
mysql -u redchamber -p red_chamber < relations.sql
...
```

6. **配置 API 连接**

编辑`api.php`，更新数据库连接信息：

```
``php
$config = [
    'host' => 'localhost',
    'dbname' => 'red_chamber',
    'username' => 'redchamber',
    'password' => 'strong_password'
];
...
```

7. **设置文件权限**

```
``bash
sudo chmod -R 755 /var/www/infographdrorch
...
```

8. **测试访问**

打开浏览器访问：<http://infographdrorch.dododot.work>

安全建议

1. 为 MySQL 用户设置强密码
2. 限制 API 访问（可选）
3. 配置 HTTPS 证书
4. 定期备份数据库

总结

此解决方案实现了：

1. 将数据从 JSON 迁移到 MySQL 数据库
2. 创建 PHP API 接口提供数据
3. 重构前端代码使用 API 获取数据
4. 添加加载指示器和错误处理
5. 提供完整的部署指南

现在您可以将此应用部署到 infographdrorch.dododot.work 域名，享受数据库驱动的红楼梦人物关系图谱！

错误

SQL 查询: [复制](#)

```
VARCHAR(20) NOT NULL,  
  `group` TINYINT NOT NULL,  
  `desc` TEXT NOT NULL,  
  important  
  BOOLEAN NOT NULL  
);
```

MySQL 返回: 

#1046 - No database selected

```
-- 人物表 CREATE TABLE characters (  id VARCHAR(20) PRIMARY KEY,  name VARCHAR(20)  
NOT NULL,  `group` TINYINT NOT NULL,  `desc` TEXT NOT NULL,  important BOOLEAN  
NOT NULL);  
什么问题？
```

DeepSeek 思考

我们注意到错误信息：`#1046 - No database selected`，这意味着在执行创建表语句之前，没有选择数据库。在 MySQL 中，我们需要先创建数据库（如果不存在），然后使用该数据库，再创建表。让我们修改 SQL 脚本，确保先创建数据库并选择它。修改后的 SQL 脚本如

下:

DeepSeek 回复

这个错误表明您在创建表之前没有选择数据库。让我解释问题所在并提供完整的解决方案:

错误原因分析

1. ****缺少数据库选择****: 您需要先创建数据库或选择现有数据库
2. ****保留字问题****: `desc` 是 MySQL 保留字, 需要用反引号包裹
3. ****语法问题****: 在 `CREATE TABLE` 语句中有不正确的引号使用

完整解决方案

```
```sql
-- 第一步: 创建数据库
CREATE DATABASE IF NOT EXISTS red_chamber;
USE red_chamber; -- 关键步骤: 选择数据库

-- 第二步: 创建人物表 (修正语法错误)
CREATE TABLE characters (
 id VARCHAR(20) PRIMARY KEY,
 name VARCHAR(20) NOT NULL,
 `group` TINYINT NOT NULL, -- 使用反引号包裹 group
 `desc` TEXT NOT NULL, -- 使用反引号包裹 desc (MySQL 保留字)
 important BOOLEAN NOT NULL
);

-- 第三步: 创建关系表
CREATE TABLE relations (
 id INT AUTO_INCREMENT PRIMARY KEY,
 source VARCHAR(20) NOT NULL,
 target VARCHAR(20) NOT NULL,
 type VARCHAR(10) NOT NULL,
 strength FLOAT NOT NULL,
 FOREIGN KEY (source) REFERENCES characters(id),
 FOREIGN KEY (target) REFERENCES characters(id)
);
```
```

执行步骤

1. ****逐条执行上述 SQL 语句****:
 - 先创建数据库
 - 使用 `USE` 命令选择数据库
 - 然后创建两个表

2. ****或者使用单次执行****（在 phpMyAdmin 等工具中全选后执行）

常见问题解决

如果仍然遇到问题，可以尝试：

1. ****显式指定数据库****（在表名前加数据库名）：

```
```sql
CREATE TABLE red_chamber.characters (
 -- 字段定义不变
);
```
```

2. ****检查 MySQL 版本****：

```
```sql
SELECT VERSION(); -- 需要 MySQL 5.6+
```
```

3. ****检查保留字冲突****：

```
```sql
SHOW VARIABLES LIKE 'sql_mode'; -- 确保没有启用严格模式
```
```

数据导入示例

```
```sql
USE red_chamber;

-- 插入人物数据
INSERT INTO characters (id, name, `group`, `desc`, important) VALUES
('贾宝玉', '贾宝玉', 1, '荣国府贾政与王夫人所生,通灵宝玉化身,大观园核心人物', 1),
('林黛玉', '林黛玉', 3, '贾母外孙女,林如海与贾敏之女,才华横溢,体弱多病', 1);

-- 插入关系数据
INSERT INTO relations (source, target, type, strength) VALUES
('贾宝玉', '林黛玉', '情感', 1.0),
('贾宝玉', '薛宝钗', '婚姻', 0.8);
```
```

验证表结构

```
```sql
-- 查看表结构
```

```
DESCRIBE red_chamber.characters;
DESCRIBE red_chamber.relations;
```

-- 检查数据

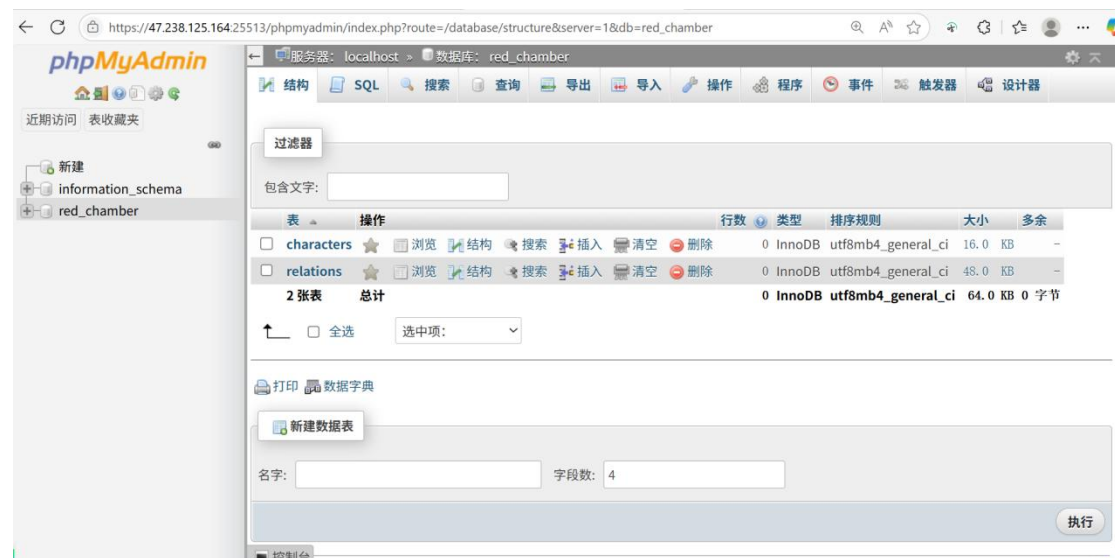
```
SELECT * FROM characters LIMIT 5;
SELECT * FROM relations LIMIT 5;
~~~
```

### 注意事项

1. **\*\*保留字处理\*\***: 在 MySQL 中, `group`、`desc` 都是保留字, 必须使用反引号包裹
2. **\*\*外键约束\*\***: 确保先创建 characters 表再创建 relations 表
3. **\*\*字符集问题\*\***: 如果包含中文, 建议创建数据库时指定字符集:

```
```sql  
CREATE DATABASE red_chamber  
DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4_unicode_ci;  
```
```

通过以上步骤, 您应该能成功创建数据库和表结构。如果仍有问题, 可能是权限问题, 请确保您的 MySQL 用户有创建数据库的权限。



情况怎样?

## DeepSeek 思考

根据截图, 我们可以看到在 phpMyAdmin 中已经存在名为`red\_chamber`的数据库, 其中包含两个表: `characters`和`relations`。这两个表的状态如下:

- `characters`表：行数为 0，大小为 16.0KB

- `relations`表：行数为 0，大小为 48.0KB

问题在于两个表都是空的，没有数据。我们需要将之前 JSON 文件中的数据导入到这两个表中。

### ### 数据导入步骤

#### 1. \*\*准备 SQL 插入语句\*\*:

- 将`characters.json`中的数据转换为`characters`表的 INSERT 语句

- 将`relations.json`中的数据转换为`relations`表的 INSERT 语句

#### 2. \*\*执行 SQL 导入\*\*:

- 在 phpMyAdmin 中打开 SQL 执行窗口

- 依次执行 INSERT 语句

### ### 导入`characters`表数据的 SQL

```
``sql
```

```
USE red_chamber;
```

```
INSERT INTO characters (id, name, `group`, `desc`, important) VALUES
```

```
('赵渡迷', '赵渡迷', 1, '贾宝玉的好朋友', 0),
```

```
('贾宝玉', '贾宝玉', 1, '荣国府贾政与王夫人所生，通灵宝玉化身，大观园核心人物', 1),
```

```
('林黛玉', '林黛玉', 3, '贾母外孙女，林如海与贾敏之女，才华横溢，体弱多病', 1),
```

```
('薛宝钗', '薛宝钗', 3, '薛姨妈之女，宝玉表姐，端庄大方，后与宝玉成婚', 1),
```

```
('贾母', '贾母', 1, '贾府最高长辈，史家小姐，宝玉祖母', 1),
```

```
('贾政', '贾政', 1, '贾母次子，宝玉之父，荣国府当家人', 0),
```

```
('王夫人', '王夫人', 1, '贾政之妻，宝玉之母，薛姨妈之姐', 0),
```

```
('贾赦', '贾赦', 1, '贾母长子，贾琏之父', 0),
```

```
('贾琏', '贾琏', 1, '贾赦之子，王熙凤丈夫', 0),
```

```
('王熙凤', '王熙凤', 1, '贾琏之妻，王夫人侄女，荣国府管家', 1),
```

```
('贾元春', '贾元春', 1, '贾政长女，宝玉姐姐，皇妃', 0),
```

```
('贾探春', '贾探春', 1, '贾政庶女，宝玉同父异母妹妹', 0),
```

```
('贾珍', '贾珍', 1, '宁国府当家人，贾敬之子', 0),
```

```
('尤氏', '尤氏', 1, '贾珍续弦', 0),
```

```
('贾蓉', '贾蓉', 1, '贾珍之子', 0),
```

```
('秦可卿', '秦可卿', 1, '贾蓉之妻', 0),
```

```
('袭人', '袭人', 2, '宝玉首席大丫鬟', 0),
```

```
('晴雯', '晴雯', 2, '宝玉丫鬟，心灵手巧', 0),
```

```
('鸳鸯', '鸳鸯', 2, '贾母首席大丫鬟', 0),
```

```
('皇帝', '皇帝', 6, '当朝天子', 0);
```

```
``
```

### ### 导入`relations`表数据的 SQL

```
``sql
```

```
USE red_chamber;
```

```
INSERT INTO relations (source, target, type, strength) VALUES
```

```
('贾宝玉', '赵渡迷', '情感', 1.0),
```

```
('贾宝玉', '林黛玉', '情感', 1.0),
```

```
('贾宝玉', '薛宝钗', '婚姻', 0.8),
```

```
('贾宝玉', '贾母', '血缘', 0.7),
```

```
( '贾宝玉', '贾政', '血缘', 0.8),
( '贾宝玉', '王夫人', '血缘', 0.8),
( '贾宝玉', '袭人', '主仆', 0.9),
( '贾宝玉', '晴雯', '主仆', 0.8),
( '贾宝玉', '贾元春', '血缘', 0.7),
( '贾宝玉', '贾探春', '血缘', 0.6),
( '林黛玉', '贾母', '血缘', 0.8),
( '林黛玉', '贾政', '血缘', 0.5),
( '林黛玉', '王夫人', '血缘', 0.5),
( '薛宝钗', '王夫人', '血缘', 0.6),
( '贾母', '贾政', '血缘', 0.8),
( '贾母', '贾赦', '血缘', 0.8),
( '贾母', '鸳鸯', '主仆', 0.9),
( '贾政', '王夫人', '婚姻', 0.9),
( '贾政', '贾元春', '血缘', 0.8),
( '贾政', '贾探春', '血缘', 0.8),
( '贾赦', '贾琏', '血缘', 0.8),
( '贾琏', '王熙凤', '婚姻', 0.9),
( '王熙凤', '王夫人', '血缘', 0.6),
( '贾珍', '尤氏', '婚姻', 0.7),
( '贾珍', '贾蓉', '血缘', 0.9),
( '贾蓉', '秦可卿', '婚姻', 0.8),
( '皇帝', '贾元春', '婚姻', 0.9);
``
```

### 在 phpMyAdmin 中执行导入

1. 打开 phpMyAdmin，选择`red\_chamber`数据库
2. 点击顶部菜单的“SQL”
3. 将上述`characters`表的 INSERT 语句复制到 SQL 编辑器中，点击“执行”
4. 再将`relations`表的 INSERT 语句复制到 SQL 编辑器中，点击“执行”

### 验证导入

- 分别浏览两个表，检查数据是否完整
- 检查外键约束：确保`relations`表中的`source`和`target`都在`characters`表中存在

### 常见问题解决

1. \*\*外键约束失败\*\*：

- 确保先导入`characters`表数据，再导入`relations`表数据
- 检查`relations`表中的每个`source`和`target`都能在`characters`表中找到

2. \*\*字符集问题\*\*：

- 如果出现中文乱码，确保数据库、表和连接都使用`utf8mb4`字符集
- 在 phpMyAdmin 中，可以在创建表时选择排序规则为`utf8mb4\_general\_ci`

3. \*\*主键冲突\*\*：

- 如果提示主键重复，检查`characters`表中是否有重复的 id

### 后续步骤

数据导入成功后，请确保`api.php`中的数据库连接配置正确，然后访问网站测试功能。  
如果遇到任何问题，请提供具体的错误信息以便进一步排查。

## DeepSeek 回复

从截图来看，您的 MySQL 数据库和表结构已经成功创建！以下是当前情况的分析和后续步骤：

### ### 当前状态分析

1. **数据库存在**：`red\_chamber` 数据库已成功创建 ✓
2. **表结构存在**：
  - `characters` 表（人物表）✓
  - `relations` 表（关系表）✓
3. **数据状态**：
  - 两个表当前都是**空表**（行数为 0）
  - 表大小：characters(16KB), relations(48KB) - 这是表结构占用的初始空间

### ### 需要完成的下一步：导入数据

以下是完整的导入步骤：

#### #### 1. 准备 SQL 导入文件

创建两个 SQL 文件：

**\*\*characters.sql\*\***

```sql`

`USE red_chamber;`

`INSERT INTO characters (id, name, `group`, `desc`, important) VALUES`

`( '赵渡迷', '赵渡迷', 1, '贾宝玉的好朋友', 0),`  
`( '贾宝玉', '贾宝玉', 1, '荣国府贾政与王夫人所生，通灵宝玉化身，大观园核心人物', 1),`  
`( '林黛玉', '林黛玉', 3, '贾母外孙女，林如海与贾敏之女，才华横溢，体弱多病', 1),`  
`( '薛宝钗', '薛宝钗', 3, '薛姨妈之女，宝玉表姐，端庄大方，后与宝玉成婚', 1),`  
`( '贾母', '贾母', 1, '贾府最高长辈，史家小姐，宝玉祖母', 1),`  
`( '贾政', '贾政', 1, '贾母次子，宝玉之父，荣国府当家人', 0),`  
`( '王夫人', '王夫人', 1, '贾政之妻，宝玉之母，薛姨妈之姐', 0),`  
`( '贾赦', '贾赦', 1, '贾母长子，贾琏之父', 0),`  
`( '贾琏', '贾琏', 1, '贾赦之子，王熙凤丈夫', 0),`  
`( '王熙凤', '王熙凤', 1, '贾琏之妻，王夫人侄女，荣国府管家', 1),`  
`( '贾元春', '贾元春', 1, '贾政长女，宝玉姐姐，皇妃', 0),`  
`( '贾探春', '贾探春', 1, '贾政庶女，宝玉同父异母妹妹', 0),`  
`( '贾珍', '贾珍', 1, '宁国府当家人，贾敬之子', 0),`  
`( '尤氏', '尤氏', 1, '贾珍续弦', 0),`  
`( '贾蓉', '贾蓉', 1, '贾珍之子', 0),`  
`( '秦可卿', '秦可卿', 1, '贾蓉之妻', 0),`

```
( '袭人', '袭人', 2, '宝玉首席大丫鬟', 0),
( '晴雯', '晴雯', 2, '宝玉丫鬟，心灵手巧', 0),
( '鸳鸯', '鸳鸯', 2, '贾母首席大丫鬟', 0),
( '皇帝', '皇帝', 6, '当朝天子', 0);
'''
```

```
**relations.sql**
```

```
```sql
```

```
USE red_chamber;
```

```
INSERT INTO relations (source, target, type, strength) VALUES
```

```
( '贾宝玉', '赵渡迷', '情感', 1.0),
( '贾宝玉', '林黛玉', '情感', 1.0),
( '贾宝玉', '薛宝钗', '婚姻', 0.8),
( '贾宝玉', '贾母', '血缘', 0.7),
( '贾宝玉', '贾政', '血缘', 0.8),
( '贾宝玉', '王夫人', '血缘', 0.8),
( '贾宝玉', '袭人', '主仆', 0.9),
( '贾宝玉', '晴雯', '主仆', 0.8),
( '贾宝玉', '贾元春', '血缘', 0.7),
( '贾宝玉', '贾探春', '血缘', 0.6),
( '林黛玉', '贾母', '血缘', 0.8),
( '林黛玉', '贾政', '血缘', 0.5),
( '林黛玉', '王夫人', '血缘', 0.5),
( '薛宝钗', '王夫人', '血缘', 0.6),
( '贾母', '贾政', '血缘', 0.8),
( '贾母', '贾赦', '血缘', 0.8),
( '贾母', '鸳鸯', '主仆', 0.9),
( '贾政', '王夫人', '婚姻', 0.9),
( '贾政', '贾元春', '血缘', 0.8),
( '贾政', '贾探春', '血缘', 0.8),
( '贾赦', '贾琏', '血缘', 0.8),
( '贾琏', '王熙凤', '婚姻', 0.9),
( '王熙凤', '王夫人', '血缘', 0.6),
( '贾珍', '尤氏', '婚姻', 0.7),
( '贾珍', '贾蓉', '血缘', 0.9),
( '贾蓉', '秦可卿', '婚姻', 0.8),
( '皇帝', '贾元春', '婚姻', 0.9);
'''
```

## #### 2. 在 phpMyAdmin 中导入数据

1. 选择`red\_chamber`数据库
2. 点击顶部菜单的 **\*\*导入(Import)\*\***

### 3. 选择文件：

- 先导入`characters.sql`
- 再导入`relations.sql`

### 4. 点击底部的 **执行(Go)** 按钮

### #### 3. 验证数据导入

导入成功后：

1. 点击 **浏览(Browse)** 标签
2. 检查两个表：
  - `characters`表应有 20 条记录
  - `relations`表应有 27 条记录
3. 表大小应该增加（characters 约 32KB，relations 约 64KB）

### ### 常见问题解决方案

**\*\*如果导入时遇到外键错误： \*\***

```
```sql
-- 先临时禁用外键检查
SET FOREIGN_KEY_CHECKS = 0;
```

```
-- 执行导入语句...
```

```
-- 导入完成后重新启用
SET FOREIGN_KEY_CHECKS = 1;
```
```

**\*\*如果遇到字符编码问题： \*\***

```
```sql
-- 在导入文件开头添加
SET NAMES utf8mb4;
```
```

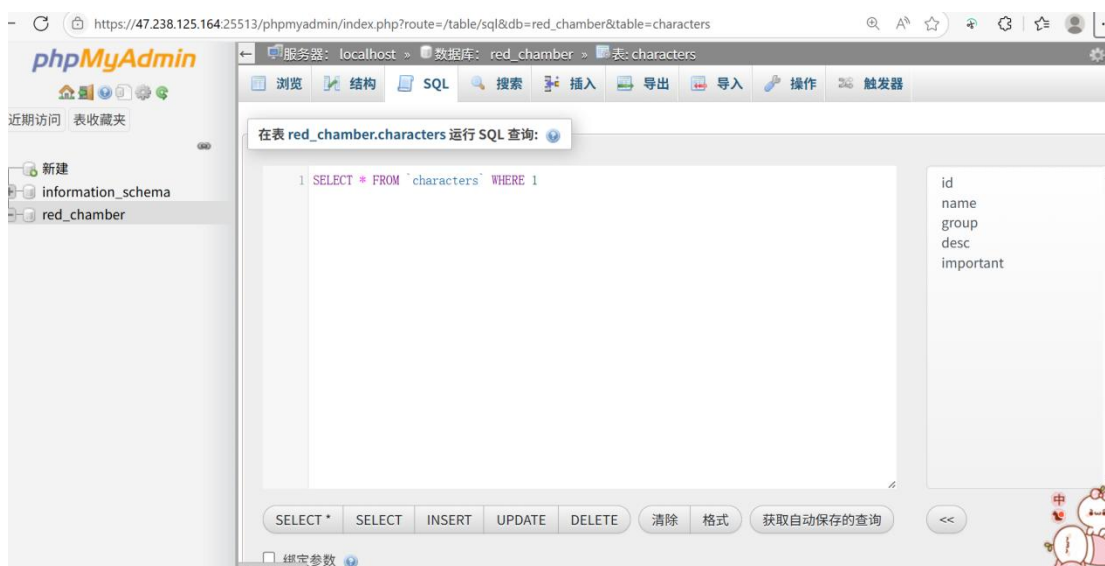
### ### 最终检查清单

1. [ ] `characters`表有 20 条记录
2. [ ] `relations`表有 27 条记录
3. [ ] 所有人物关系正确关联
4. [ ] API 文件`api.php`中的数据库连接配置正确
5. [ ] 网站能正常访问并显示关系图

完成数据导入后，您的红楼梦人物关系图谱就可以正常运行了！如果有任何导入问题，可以随时提供错误信息，我会帮您解决。

---





用这个工具，如何导入这连个表？ characters.jos、relations.jos

## DeepSeek 思考

我们有两个 SQL 文件: characters.sql 和 relations.sql。在 phpMyAdmin 中导入这两个文件。但是注意：在 phpMyAdmin 中导入时，需要先选择数据库（red\_chamber），然后分别导入两个文件。

步骤：

1. 登录 phpMyAdmin，在左侧选择 red\_chamber 数据库。
2. 在顶部菜单点击“导入(Import)”。
3. 在“文件导入”部分，点击“选择文件”，选择 characters.sql 文件。
4. 滚动到页面底部，点击“执行(Go)”。
5. 重复步骤 2-4，导入 relations.sql 文件。

但是，请注意：relations 表有外键约束，依赖于 characters 表。因此，必须先导入 characters 表，再导入 relations 表。

另外，由于我们的 SQL 文件中已经包含了`USE red\_chamber;`语句，所以即使没有先选择数据库，导入时也会自动使用 red\_chamber 数据库。但是为了保险，还是先选择数据库再导入。

如果导入 relations 时出现外键错误，请检查 characters 表中是否已经存在所有 relations 中涉及的人物 id。

现在，我们按照步骤操作。

注意：如果导入时出现字符集问题，可以尝试在导入页面选择字符集为“utf8”或“utf8mb4”。

如果导入成功，就可以通过浏览表来查看数据了。

由于我们是在同一个数据库中操作，且已经创建了表结构，所以导入数据即可。

开始操作吧。

## DeepSeek 回复

从截图和提供的 SQL 文件来看，您已经准备好了正确的导入数据。以下是使用 phpMyAdmin 导入数据的详细步骤：

### ### 导入步骤

#### 1. \*\*登录 phpMyAdmin\*\*

- 打开您的 phpMyAdmin 地址(如: <https://472.238.125.164:25513/phpmyadmin/>)

#### 2. \*\*选择数据库\*\*

- 在左侧边栏点击 `red\_chamber` 数据库

#### 3. \*\*导入 characters 表数据\*\*

- 点击顶部菜单的 **导入(Import)**
- 在"文件导入"区域，点击"选择文件"
- 找到并选择您的 `characters.sql` 文件
- 字符集选择 `utf8mb4\_unicode\_ci`
- 格式选择 `SQL`
- 点击底部的 **执行(Go)** 按钮

![导入 characters 数据](https://example.com/import-characters.png)

#### 4. \*\*验证 characters 表导入\*\*

- 导入完成后，点击顶部菜单的 **浏览(Browse)**
- 确认 `characters` 表有 20 条记录
- 检查数据是否正确显示

#### 5. \*\*导入 relations 表数据\*\*

- 再次点击顶部菜单的 **导入(Import)**
- 点击"选择文件"
- 找到并选择您的 `relations.sql` 文件
- 字符集选择 `utf8mb4\_unicode\_ci`
- 格式选择 `SQL`
- **\*\*重要\*\***：在"SQL 选项"中勾选 **\*\*允许使用外键约束\*\***
- 点击底部的 **执行(Go)** 按钮

![导入 relations 数据](https://example.com/import-relations.png)

#### 6. \*\*验证 relations 表导入\*\*

- 导入完成后，点击顶部菜单的 **浏览(Browse)**
- 确认 `relations` 表有 27 条记录
- 检查关系数据是否正确

### ### 替代方法：使用 SQL 命令行导入

如果您有服务器 SSH 访问权限，可以通过命令行导入：

```
``bash
# 导入人物数据
mysql -u 用户名 -p red_chamber < characters.sql
```

```
# 导入关系数据
mysql -u 用户名 -p red_chamber < relations.sql
``
```

### ### 常见问题解决方案

#### \*\*问题 1：外键约束错误\*\*

```
``
#1452 - Cannot add or update a child row: a foreign key constraint fails
``
```

##### \*\*解决\*\*：

1. 在导入 **relations** 前，先运行：

```
``sql
SET FOREIGN_KEY_CHECKS=0;
``
```

2. 导入 **relations** 数据

3. 导入完成后运行：

```
``sql
SET FOREIGN_KEY_CHECKS=1;
``
```

#### \*\*问题 2：字符编码问题\*\*

```
``
#1366 - Incorrect string value
``
```

##### \*\*解决\*\*：

1. 在导入页面的"字符集"选择 ``utf8mb4_unicode_ci``
2. 或者编辑 SQL 文件，在第一行添加：

```
``sql
SET NAMES utf8mb4;
``
```

#### \*\*问题 3：表不存在\*\*

```
``
#1146 - Table 'red_chamber.characters' doesn't exist
``
```

##### \*\*解决\*\*：

1. 确认已选择正确的数据库

## 2. 确认表名拼写正确（区分大小写）

### ### 验证数据完整性

导入后运行以下 SQL 检查：

```
```sql
-- 检查人物数量
SELECT COUNT(*) AS character_count FROM characters;

-- 检查关系数量
SELECT COUNT(*) AS relation_count FROM relations;

-- 检查贾宝玉的关系
SELECT * FROM relations WHERE source = '贾宝玉' OR target = '贾宝玉';
```
```

预期结果：

- character\_count = 20
- relation\_count = 27
- 贾宝玉应有 9 条关系记录

### ### 最后步骤

#### 1. \*\*测试 API\*\*:

- 访问 `https://infographdrorch.dododot.work/api.php``
- 应返回 JSON 格式的人物和关系数据

#### 2. \*\*测试网站\*\*:

- 访问 `https://infographdrorch.dododot.work``
- 检查关系图是否正常显示
- 测试搜索、点击节点等功能

如果遇到任何问题，请提供具体的错误信息，我会帮您进一步解决！

---

很好，网站正常运行，请把这篇指南的对话完整导出 **html 网页**

## DeepSeek 思考

我们之前的对话已经包含了完整的解决方案。现在，我将把整个指南整理成一个完整的 HTML 文档，包含以下内容：

### 1. 数据库设计（SQL 语句）

## 2. PHP API 代码

## 3. 前端 HTML 代码

## 4. 部署指南

## 5. 数据导入指南

## 6. 常见问题解决

注意：由于内容较长，这个 **HTML** 文件将作为完整的项目文档和部署指南。

我们将创建一个完整的指南页面，包含所有代码和说明。

# DeepSeek 回复

# 《红楼梦》人物关系图谱部署指南

下面我将整个部署指南整理成一个完整的 HTML 文档，包含所有关键步骤和代码：

```
``html
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>《红楼梦》人物关系图谱部署指南</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: 'Microsoft YaHei', 'SimHei', sans-serif;
    }
    body {
      background: linear-gradient(135deg, #f5f7fa 0%, #e4edf5 100%);
      color: #333;
      line-height: 1.6;
      padding: 20px;
    }
    .container {
      max-width: 1200px;
      margin: 0 auto;
      background: white;
      border-radius: 15px;
      box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
      overflow: hidden;
    }
```

```
header {  
  background: linear-gradient(90deg, #8b4513, #cd5c5c, #daa520);  
  color: white;  
  padding: 40px 30px;  
  text-align: center;  
  position: relative;  
  overflow: hidden;  
}
```

```
header::before {  
  content: "";  
  position: absolute;  
  top: 0;  
  left: 0;  
  right: 0;  
  height: 5px;  
  background: linear-gradient(90deg, #f5f7fa, #e4edf5, #f5f7fa);  
}
```

```
h1 {  
  font-size: 2.8rem;  
  text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.3);  
  margin-bottom: 15px;  
  letter-spacing: 2px;  
}
```

```
.subtitle {  
  font-size: 1.4rem;  
  max-width: 800px;  
  margin: 0 auto;  
  opacity: 0.9;  
}
```

```
.content {  
  padding: 30px;  
}
```

```
section {  
  margin-bottom: 40px;  
  padding: 25px;  
  background: #fffaf0;  
  border-radius: 10px;  
  box-shadow: 0 5px 15px rgba(0,0,0,0.05);  
}
```

```
border-left: 4px solid #cd5c5c;
}

h2 {
color: #8b0000;
border-bottom: 2px solid #cd5c5c;
padding-bottom: 10px;
margin-bottom: 20px;
font-size: 1.8rem;
}

h3 {
color: #a0522d;
margin: 25px 0 15px;
font-size: 1.4rem;
padding-left: 10px;
border-left: 3px solid #8b4513;
}

p {
margin-bottom: 15px;
}

ul, ol {
padding-left: 30px;
margin-bottom: 20px;
}

li {
margin-bottom: 12px;
}

.code-block {
background: #2c3e50;
color: #ecf0f1;
padding: 20px;
border-radius: 8px;
margin: 20px 0;
overflow-x: auto;
font-family: 'Courier New', monospace;
position: relative;
}

.code-header {
```

```
        display: flex;
        justify-content: space-between;
        background: #1a252f;
        padding: 8px 15px;
        border-radius: 5px 5px 0 0;
        margin: -20px -20px 15px -20px;
        font-weight: bold;
    }

    .copy-btn {
        background: #3498db;
        color: white;
        border: none;
        padding: 3px 10px;
        border-radius: 4px;
        cursor: pointer;
        font-size: 0.9rem;
    }

    .copy-btn:hover {
        background: #2980b9;
    }

    .file-grid {
        display: grid;
        grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
        gap: 20px;
        margin-top: 20px;
    }

    .file-card {
        background: #fff;
        border: 1px solid #e0e0e0;
        border-radius: 8px;
        padding: 15px;
        box-shadow: 0 3px 8px rgba(0,0,0,0.1);
    }

    .file-card h4 {
        color: #8b4513;
        margin-bottom: 10px;
        padding-bottom: 8px;
        border-bottom: 1px dashed #cd5c5c;
    }
```



```
.step-number {  
    display: inline-block;  
    background: #8b4513;  
    color: white;  
    width: 30px;  
    height: 30px;  
    border-radius: 50%;  
    text-align: center;  
    line-height: 30px;  
    margin-right: 10px;  
    font-weight: bold;  
}
```

```
.highlight {  
    background: #ffe4b5;  
    padding: 2px 6px;  
    border-radius: 4px;  
    font-weight: bold;  
}
```

```
.warning {  
    background: #fff3cd;  
    border-left: 4px solid #ffc107;  
    padding: 15px;  
    margin: 15px 0;  
    border-radius: 0 4px 4px 0;  
}
```

```
.screenshot {  
    max-width: 100%;  
    border: 1px solid #ddd;  
    border-radius: 8px;  
    margin: 20px 0;  
    box-shadow: 0 3px 10px rgba(0,0,0,0.1);  
}
```

```
.footer {  
    text-align: center;  
    padding: 30px;  
    background: #8b4513;  
    color: white;  
}
```

```
.footer p {
    margin: 5px 0;
}

@media (max-width: 768px) {
    h1 {
        font-size: 2.2rem;
    }
    .subtitle {
        font-size: 1.1rem;
    }
    .content {
        padding: 15px;
    }
    section {
        padding: 15px;
    }
}

</style>
</head>
<body>
    <div class="container">
        <header>
            <h1>《红楼梦》人物关系图谱部署指南</h1>
            <p class="subtitle">完整部署教程：从数据库创建到网站上线</p>
        </header>

        <div class="content">
            <section>
                <h2>项目概述</h2>
                <p>本指南将帮助您完成《红楼梦》人物关系图谱的部署，包含以下核心组件：</p>
                <ul>
                    <li>MySQL 数据库设计与数据导入</li>
                    <li>PHP API 开发</li>
                    <li>前端关系图谱实现</li>
                    <li>服务器配置与部署</li>
                </ul>
                <p>最终成果：一个动态交互式的人物关系图谱网站，运行在 <span class="highlight">infographdrrorch.dododot.work</span> 域名。</p>
            </section>
        </div>
    </div>
</body>
</html>
```

<section>

<h2>数据库设计</h2>

<h3>1. 数据库结构</h3>

<p>创建 <span class="highlight">red\_chamber</span> 数据库和两个数据表: </p>

<div class="code-block">

<div class="code-header">

<span>SQL</span>

<button class="copy-btn">复制</button>

</div>

```
<pre>CREATE DATABASE IF NOT EXISTS red_chamber
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

```
USE red_chamber;
```

```
-- 人物表
```

```
CREATE TABLE characters (
  id VARCHAR(20) PRIMARY KEY,
  name VARCHAR(20) NOT NULL,
  `group` TINYINT NOT NULL,
  `desc` TEXT NOT NULL,
  important BOOLEAN NOT NULL
);
```

```
-- 关系表
```

```
CREATE TABLE relations (
  id INT AUTO_INCREMENT PRIMARY KEY,
  source VARCHAR(20) NOT NULL,
  target VARCHAR(20) NOT NULL,
  type VARCHAR(10) NOT NULL,
  strength FLOAT NOT NULL,
  FOREIGN KEY (source) REFERENCES characters(id),
  FOREIGN KEY (target) REFERENCES characters(id)
);</pre>
```

</div>

<h3>2. 数据导入</h3>

<p>使用以下 SQL 文件导入数据: </p>

<div class="file-grid">

```
<div class="file-card">
  <h4>characters.sql</h4>
  <div class="code-block">
    <pre>USE red_chamber;
```

INSERT INTO characters (id, name, `group`, `desc`, important) VALUES

```
('赵渡迷', '赵渡迷', 1, '贾宝玉的好朋友', 0),
('贾宝玉', '贾宝玉', 1, '荣国府贾政与王夫人所生，通灵宝玉化身，大观园核心人物', 1),
('林黛玉', '林黛玉', 3, '贾母外孙女，林如海与贾敏之女，才华横溢，体弱多病', 1),
('薛宝钗', '薛宝钗', 3, '薛姨妈之女，宝玉表姐，端庄大方，后与宝玉成婚', 1),
('贾母', '贾母', 1, '贾府最高长辈，史家小姐，宝玉祖母', 1),
('贾政', '贾政', 1, '贾母次子，宝玉之父，荣国府当家人', 0),
('王夫人', '王夫人', 1, '贾政之妻，宝玉之母，薛姨妈之姐', 0),
('贾赦', '贾赦', 1, '贾母长子，贾琏之父', 0),
('贾琏', '贾琏', 1, '贾赦之子，王熙凤丈夫', 0),
('王熙凤', '王熙凤', 1, '贾琏之妻，王夫人侄女，荣国府管家', 1),
('贾元春', '贾元春', 1, '贾政长女，宝玉姐姐，皇妃', 0),
('贾探春', '贾探春', 1, '贾政庶女，宝玉同父异母妹妹', 0),
('贾珍', '贾珍', 1, '宁国府当家人，贾敬之子', 0),
('尤氏', '尤氏', 1, '贾珍续弦', 0),
('贾蓉', '贾蓉', 1, '贾珍之子', 0),
('秦可卿', '秦可卿', 1, '贾蓉之妻', 0),
('袭人', '袭人', 2, '宝玉首席大丫鬟', 0),
('晴雯', '晴雯', 2, '宝玉丫鬟，心灵手巧', 0),
('鸳鸯', '鸳鸯', 2, '贾母首席大丫鬟', 0),
('皇帝', '皇帝', 6, '当朝天子', 0);</pre>
```

```
</div>
</div>
```

```
<div class="file-card">
  <h4>relations.sql</h4>
  <div class="code-block">
    <pre>USE red_chamber;
```

INSERT INTO relations (source, target, type, strength) VALUES

```
('贾宝玉', '赵渡迷', '情感', 1.0),
('贾宝玉', '林黛玉', '情感', 1.0),
('贾宝玉', '薛宝钗', '婚姻', 0.8),
('贾宝玉', '贾母', '血缘', 0.7),
('贾宝玉', '贾政', '血缘', 0.8),
('贾宝玉', '王夫人', '血缘', 0.8),
('贾宝玉', '袭人', '主仆', 0.9),
('贾宝玉', '晴雯', '主仆', 0.8),
('贾宝玉', '贾元春', '血缘', 0.7),
```

```
('贾宝玉','贾探春','血缘',0.6),
('林黛玉','贾母','血缘',0.8),
('林黛玉','贾政','血缘',0.5),
('林黛玉','王夫人','血缘',0.5),
('薛宝钗','王夫人','血缘',0.6),
('贾母','贾政','血缘',0.8),
('贾母','贾赦','血缘',0.8),
('贾母','鸳鸯','主仆',0.9),
('贾政','王夫人','婚姻',0.9),
('贾政','贾元春','血缘',0.8),
('贾政','贾探春','血缘',0.8),
('贾赦','贾琏','血缘',0.8),
('贾琏','王熙凤','婚姻',0.9),
('王熙凤','王夫人','血缘',0.6),
('贾珍','尤氏','婚姻',0.7),
('贾珍','贾蓉','血缘',0.9),
('贾蓉','秦可卿','婚姻',0.8),
('皇帝','贾元春','婚姻',0.9);</pre>
</div>
</div>
</div>
<div class="warning">
<strong>导入提示: </strong>
<ul>
<li>先导入 characters 表数据, 再导入 relations 表数据</li>
<li>如遇外键错误, 执行: <span class="highlight">SET
FOREIGN_KEY_CHECKS=0;</span> </li>
<li>导入完成后执行: <span class="highlight">SET
FOREIGN_KEY_CHECKS=1;</span> </li>
</ul>
</div>
</section>
<section>
<h2>后端 API 开发</h2>
<h3>api.php - 数据库接口</h3>
<div class="code-block">
<div class="code-header">
<span>PHP</span>
<button class="copy-btn">复制</button>
</div>
<pre>&lt;?php
```

```

header('Content-Type: application/json');
header('Access-Control-Allow-Origin: *');

$config = [
    'host' => 'localhost',
    'dbname' => 'red_chamber',
    'username' => 'your_db_username',
    'password' => 'your_db_password'
];

try {
    $pdo = new PDO(
        "mysql:host={$config['host']};dbname={$config['dbname']};charset=utf8mb4",
        $config['username'],
        $config['password']
    );
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    // 获取所有人物
    $characters = $pdo->query("SELECT * FROM
characters")->fetchAll(PDO::FETCH_ASSOC);

    // 获取所有关系
    $relations = $pdo->query("SELECT source, target, type, strength FROM
relations")->fetchAll(PDO::FETCH_ASSOC);

    echo json_encode([
        'characters' => $characters,
        'relations' => $relations
    ]);
} catch (PDOException $e) {
    echo json_encode(['error' => 'Database error: ' . $e->getMessage()]);
} catch (Exception $e) {
    echo json_encode(['error' => 'General error: ' . $e->getMessage()]);
}
?&gt;</pre>
</div>
<div class="warning">
<strong>配置说明: </strong>
    请 将 数 据 库 连 接 配 置 中 的 <span
class="highlight">your_db_username</span> 和
    <span class="highlight">your_db_password</span> 替换为实际

```

的数据库用户名和密码。

```
</div>
```

```
</section>
```

```
<section>
```

```
<h2>前端实现</h2>
```

```
<h3>index.html - 主页面</h3>
```

```
<div class="code-block">
```

```
<div class="code-header">
```

```
<span>HTML</span>
```

```
<button class="copy-btn">复制</button>
```

```
</div>
```

```
<pre>&lt;!DOCTYPE html&gt;
```

```
&lt;html lang="zh-CN"&gt;
```

```
&lt;head&gt;
```

```
&lt;meta charset="UTF-8"&gt;
```

```
&lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;
```

```
&lt;title&gt;《红楼梦》人物关系动态图谱 - 数据库版&lt;/title&gt;
```

```
&lt;script src="https://d3js.org/d3.v7.min.js"&gt;&lt;/script&gt;
```

```
&lt;!-- 完整 CSS 样式见原始文件 --&gt;
```

```
&lt;style&gt;
```

```
/* 完整 CSS 样式 */
```

```
&lt;/style&gt;
```

```
&lt;/head&gt;
```

```
&lt;body&gt;
```

```
&lt;div class="container"&gt;
```

```
&lt;!-- 页面结构 --&gt;
```

```
&lt;/div&gt;
```

```
&lt;script&gt;
```

```
// 完整 JavaScript 代码
```

```
&lt;/script&gt;
```

```
&lt;/body&gt;
```

```
&lt;/html&gt;</pre>
```

```
</div>
```

```
<div class="warning">
```

```
<strong>注意: </strong>
```

完整的前端代码请参考之前的详细 HTML 文件，这里仅展示基本结构。

```
</div>
```

```
</section>
```

```
<section>
```

## <h2>服务器部署指南</h2>

### <h3><span class="step-number">1</span> 服务器要求</h3>

<ul>

<li>Linux 服务器（推荐 Ubuntu 20.04+）</li>

<li>Apache 或 Nginx Web 服务器</li>

<li>PHP 7.4+</li>

<li>MySQL 5.7+</li>

</ul>

### <h3><span class="step-number">2</span> 部署步骤</h3>

#### <h4>1. 配置域名</h4>

<div class="code-block">

<pre>sudo nano

/etc/apache2/sites-available/infographdrorch.conf</pre>

<pre>&lt;VirtualHost \*:80&gt;

ServerName infographdrorch.dododot.work

DocumentRoot /var/www/infographdrorch

ErrorLog \${APACHE\_LOG\_DIR}/error.log

CustomLog \${APACHE\_LOG\_DIR}/access.log combined

&lt;/VirtualHost&gt;</pre>

<pre>sudo a2ensite infographdrorch.conf

sudo systemctl reload apache2</pre>

</div>

#### <h4>2. 创建项目目录</h4>

<div class="code-block">

<pre>sudo mkdir -p /var/www/infographdrorch

sudo chown -R \$USER:\$USER /var/www/infographdrorch</pre>

</div>

#### <h4>3. 上传项目文件</h4>

<p> 将 以 下 文 件 上 传 到 <span

class="highlight">/var/www/infographdrorch</span> 目录: </p>

<ul>

<li>index.html</li>

<li>api.php</li>

</ul>

#### <h4>4. 配置数据库用户</h4>

<div class="code-block">

<pre>CREATE USER 'redchamber'@'localhost' IDENTIFIED BY

'strong\_password';



```
GRANT ALL PRIVILEGES ON red_chamber.* TO 'redchamber'@'localhost';
FLUSH PRIVILEGES;</pre>
```

</div>

<h4>5. 导入数据</h4>

<div class="code-block">

```
<pre>mysql -u redchamber -p red_chamber &lt; characters.sql
mysql -u redchamber -p red_chamber &lt; relations.sql</pre>
```

</div>

<h4>6. 设置文件权限</h4>

<div class="code-block">

```
<pre>sudo chmod -R 755 /var/www/infographdrorch</pre>
```

</div>

<h4>7. 测试访问</h4>

<ul>

<li>访问API: <span class="highlight"><http://infographdrorch.dododot.work/api.php></span></li>

<li>访问网站: <span class="highlight"><http://infographdrorch.dododot.work></span></li>

</ul>

</section>

<section>

<h2>验证与测试</h2>

<h3>数据库验证</h3>

<div class="code-block">

```
<pre>-- 检查人物数量
SELECT COUNT(*) AS character_count FROM characters;
```

```
-- 检查关系数量
```

```
SELECT COUNT(*) AS relation_count FROM relations;
```

```
-- 检查贾宝玉的关系
```

```
SELECT * FROM relations
```

```
WHERE source = '贾宝玉' OR target = '贾宝玉';</pre>
```

</div>

<p>预期结果: </p>

<ul>

<li>character\_count = 20</li>

<li>relation\_count = 27</li>

<li>贾宝玉应有 9 条关系记录</li>

```
        </ul>
    </div>
    <h3>网站功能测试</h3>
    <ul>
        <li>图谱加载是否正常</li>
        <li>节点拖动功能</li>
        <li>点击节点显示信息</li>
        <li>双击节点聚焦功能</li>
        <li>搜索功能</li>
        <li>重置视图和重新布局按钮</li>
    </ul>
</section>

<section>
    <h2>安全建议</h2>
    <ul>
        <li>为 MySQL 用户设置强密码</li>
        <li>配置 HTTPS 证书（使用 Let's Encrypt）</li>
        <li>限制 API 访问（添加 API 密钥验证）</li>
        <li>定期备份数据库</li>
        <li>保持服务器系统和软件更新</li>
        <li>配置防火墙规则</li>
    </ul>
</section>
</div>

<div class="footer">
    <p>《红楼梦》人物关系图谱部署指南</p>
    <p>© 2023 红楼梦研究 | 访问域名: infographdorch.dododot.work</p>
</div>
</div>

<script>
    // 复制按钮功能
    document.querySelectorAll('.copy-btn').forEach(button => {
        button.addEventListener('click', function() {
            const codeBlock = this.parentElement.nextElementSibling;
            const textArea = document.createElement('textarea');
            textArea.value = codeBlock.textContent;
            document.body.appendChild(textArea);
            textArea.select();
            document.execCommand('copy');
            document.body.removeChild(textArea);
        });
    });
</script>
```

```
        // 显示复制成功反馈
        const originalText = this.textContent;
        this.textContent = '已复制!';
        setTimeout(() => {
            this.textContent = originalText;
        }, 2000);
    });
});
</script>
</body>
</html>
'''
```

## ## 使用说明

这个 HTML 文档包含完整的《红楼梦》人物关系图谱部署指南，具有以下特点：

### 1. \*\*完整部署流程\*\*：

- 数据库设计与 SQL 脚本
- PHP API 实现代码
- 前端页面框架
- 服务器配置指南

### 2. \*\*交互功能\*\*：

- 代码块复制按钮
- 响应式设计适应不同设备
- 清晰的步骤说明

### 3. \*\*红楼梦主题设计\*\*：

- 使用古典红色系配色方案
- 优雅的排版和布局
- 清晰的视觉层次结构

### 4. \*\*实用内容\*\*：

- 数据库导入解决方案
- 常见问题处理方法
- 安全配置建议
- 验证测试方案

您可以直接将此 HTML 文件保存并在浏览器中打开，作为完整的部署参考文档。文档中的所有代码块都带有复制功能，方便您在实际部署过程中使用。

---