ECE 448/528
Application Software Design

Lecture 23. User Interface Design II
Spring 2025

Won-Jae Yi, Ph.D.

Department of Electrical and Computer Engineering
Illinois Institute of Technology

Handling UI Events

Delete Group: the Controller

```
class Members extends React.Component {
  ...
  render() {
    return (<MembersTable members={this.state.members}
      onDeleteGroup={this.onDeleteGroup} />);
  }
  onDeleteGroup = groupName => {
    this.deleteGroup(groupName);
  }
  ...
}
```

- Under branch `lec23-events`
- The event handler onDeleteGroup is defined in the controller.
  - Passed to the view in render.
- Let the deleteGroup method send the RESTful request.

Delete Group: the Views

```
// public/web/members_table.js
function MembersTable(props) {
  ...
  return (<table className="table table-striped  table-bordered">
       <Header groupNames={props.members.get_group_names()} />
       <Body members={props.members} onDeleteGroup={props.onDeleteGroup} />
    </table>);
}
function Body(props) {
  ...
  return (<tbody>{rows}
    <DeleteGroupsRow groupNames={props.members.get_group_names()}
      onDeleteGroup={props.onDeleteGroup} /></tbody>);
}
function DeleteGroupsRow(props) {
  var tds =...
  return (<tr><td></td>{tds}<td></td></tr>);
}
```

- Each view passes onDeleteGroup to child views that need it.
- Until the DeleteGroupsRow view, where it is associated with UI elements in tds.

Delete Group: the Views

```
function DeleteGroupsRow(props) {
  var tds =props.groupNames.map(groupName => {
    var onClick =() => props.onDeleteGroup(groupName);
    return <td key={groupName}>
      <button className={btnClassDel} onClick={onClick}>X</button></td>;
  });
  ...
}
```

- The onClick attribute of button would expect a function to be called later.
  - You cannot put props.onDeleteGroup(groupName) directly – otherwise will be called now → causing onDeleteGroup to run!
  - Make a lambda function onClick to call it later and pass the lambda function to the attribute.
- Use map instead of for loops.
  - It is difficult to capture loop variables correctly in JavaScript.

Delete Group: Interact with RESTful Backend

```
class Members extends React.Component {
  ...
  deleteGroup = groupName => {
    var delReq = {method: "DELETE"};
      fetch("api/groups/"+groupName, delReq)
        .then(rsp => this.getGroups())
        .catch(err => console.error("Members: deleteGroup", err));
  }
  ...
}
```

- Recall the event handler onDeleteGroup uses deleteGroup to send the RESTful request.
- Similar to Lecture 19, we choose to use another RESTful request to get all groups after deleting the group.
  - Make sure the server deletes the group correctly.
- getGroup will take care of state/models changes when the response arrives.
  - And React will call render's automatically.

Member Change: the Controller

```
class Members extends React.Component {
  ...
  onMemberChange = (memberName, groupName) => {
    var groupMembers = new Set(this.state.members.get_group_members(groupName))
    if (groupMembers.has(memberName))
      groupMembers.delete(memberName);
    else
      groupMembers.add(memberName);
    this.createGroup(groupName, Array.from(groupMembers));
  }
  ...
}
```

- If users click checkboxes to change members, we need to calculate whether the member should be added or removed, then let createGroup to send the RESTful request.

Member Change: the Views

```
function Row(props) {
  var members = props.members;
  var tds = members.get_group_names().map(groupName => {
    var onChange = () => props.onMemberChange(props.memberName, groupName);
    var checked = members.is_member_in_group(props.memberName, groupName);
    return (<td key={groupName}>
      <input type="checkbox" onChange={onChange} checked={checked}/></td>);
  });
  ...
}
```

- The views are created using the models.
  - The checkbox is checked or not solely depending on the model.
- What if the server backend fails right after users click the checkbox? Nothing should happen…

Hints for Troubleshooting React Code

- Read logging messages generated by the server backend to make sure it is not a problem there.
- Make sure you don't fall into JavaScript pitfalls as highlighted in this and previous lectures with red fonts.
- Add a lot of logging messages via console.info etc. in the JavaScript code.
- Read logging messages in the console of the browser.
- Install React Developer Tools in the browser to inspect React components.
    - https://addons.mozilla.org/en-US/firefox/addon/ react-devtools/