DocManager documentation

Davide Figini and Riccardo Motta

A.Y. 2021 - 2022

1 Project's description

A web application allows the online management of folders, subfolders and documents. The application supports user registration and login via a public page with appropriate forms. Registration check the uniqueness of the username. A folder has an owner, a name and a creation date and can (only) contains subfolders. A subfolder can (only) contain documents. A document has an owner, name, creation date, a summary and a type. When the uses accesses the application, a HOME PAGE appears containing a tree of its folders and subfolders. In the HOME PAGE the user can select a subfolder and access to a DOCUMENTS page which shows the list of documents of a subfolder. Each document in the list has two links: *access* and *move*. When the user selects the *access* link, a DOCUMENT page appears showing all the data of the selected document. When the user selects the *move* link, the HOME PAGE appears with the tree of folders and subfolders; in this case the page shows the message "You're moving the document X from the subfolder Y. Choose the destination subfolder", the subfolder to which the document to move belongs is NOT selectable and it's name is highlighted (for example with a different colour). When the user selects the destination subfolder, the document is moved from the origin subfolder, to the destination one and the DOCUMENTS page appears showing the updated content of the destination subfolder. Every page, except from the HOME PAGE, contains a link to go back to the previous page. The application allows the user to logout. A DOCUMENT MANAGEMENT page reachable from the HOME PAGE allows the user to create a folder, a subfolder of an existing folder and a document inside a subfolder.

1.1 Changes in the JavaScript version

- The registration check the syntactic validity of the email address and the equality between the fields "password" and "repeat password", also client-side.
- After the login, the entire application is made of a single page.
- Every user interaction is manages without refreshing the whole page, but produces the asynchronous invocation of the server and the eventual modification of the content to update following the event.
- Server-sided errors have to be reported through an alert message inside the page.
- The move functionality of a document is realized through drag and drop.
- The creation of a subfolder is made inside the HOME PAGE through an ADD SUBFOLDER button placed next to each folder. The pressing of the button lets appear an input field to insert the name of the subfolder.
- The creation of a document is made inside the HOME PAGE through an ADD DOCUMENT button placed next to each subfolder. The pressing of the button lets appear an input form to insert the data of the document.
- It's added a folder named "recycle bin". The drag and drop of a document or of a folder into the recycle bin makes it to be deleted. Before sending the deletion command to the server, the user is shown a modal window for confirmation and he/she can decide whether to cancel or proceed.

2 Requirements analysis

2.1 Data requirements analysis

A web application allows the online management of folders, subfolders and documents. The application supports user registration and login via a public page with appropriate forms. Registration check the uniqueness of the username. A folder has an owner, a name and a creation date and can (only) contains subfolders. A subfolder can (only) contain documents. A document has an owner, name, creation date, a summary and a type. When the uses accesses the application, a HOME PAGE appears containing a tree of its folders and subfolders. In the HOME PAGE the user can select a subfolder and access to a DOCUMENTS page which shows the list of documents of a subfolder. Each document in the list has two links: *access* and *move*. When the user selects the *access* link, a DOCUMENT page appears showing all the data of the selected document. When the user selects the *move* link, the HOME PAGE appears with the tree of folders and subfolders; in this case the page shows the message "You're moving the document X from the subfolder Y. Choose the destination subfolder", the subfolder to which the document to move belongs is NOT selectable and it's name is highlighted (for example with a different colour). When the user selects the destination subfolder, the document is moved from the origin subfolder, to the destination one and the DOCUMENTS page appears showing the updated content of the destination subfolder. Every page, except from the HOME PAGE, contains a link to go back to the previous page. The application allows the user to logout. A DOCUMENT MANAGEMENT page reachable from the HOME PAGE allows the user to create a folder, a subfolder of an existing folder and a document inside a subfolder.

Keys Entities Attributes Relationships

2.2 Application requirements analysis

A web application allows the online management of folders, subfolders and documents. The application supports user registration and login via a public page with appropriate forms. Registration check the uniqueness of the username. A folder has an owner, a name and a creation date and can (only) contains subfolders. A subfolder can (only) contain documents. A document has an owner, name, creation date, a summary and a type. When the uses accesses the application, a HOME PAGE appears containing a tree of its folders and subfolders. In the HOME PAGE the user can select a subfolder and access to a DOCUMENTS page which shows the list of documents of a subfolder. Each document in the list has two links: *access* and *move*. When the user selects the *access* link, a DOCUMENT page appears showing all the data of the selected document. When the user selects the *move* link, the HOME PAGE appears with the tree of folders and subfolders; in this case the page shows the message "You're moving the document X from the subfolder Y. Choose the destination subfolder", the subfolder to which the document to move belongs is NOT selectable and it's name is highlighted (for example with a different colour). When the user selects the destination subfolder, the document is moved from the origin subfolder, to the destination one and the DOCUMENTS page appears showing the updated content of the destination subfolder. Every page, except from the HOME PAGE, contains a link to go back to the previous page. The application allows the user to logout. A DOCUMENT MANAGEMENT page reachable from the HOME PAGE allows the user to create a folder, a subfolder of an existing folder and a document inside a subfolder.

Keys Pages (views) View components Events Actions

3 Completion of specifications

Moreover, we developed our application adding these specifications:

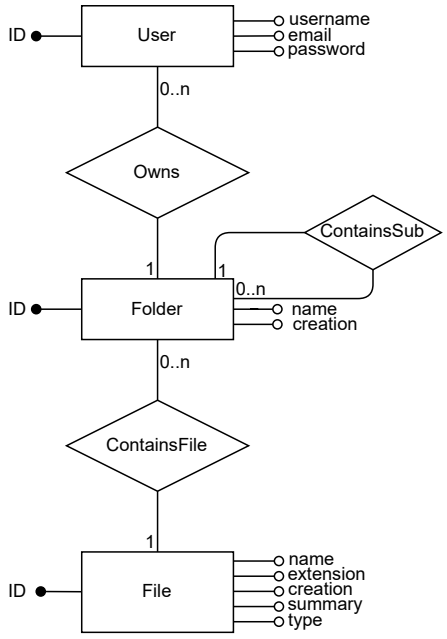
- Login and sign-up are done on two different pages, each one with a link to the other.
- When moving a document, the homepage shows a back button instead of the logout one.
- On document's move, if there is already a document with same name and extension in the destination folder, the action is aborted.
- When the creation date is not specified (only possible by tampering the client), it's set to the current one (according to server time).

4 Design

4.1 Database design

Our goal with the database was to keep it as compact as possible, privileging the number of entities over simplicity. With this approach in mind we put down a diagram where we used the same entity to represent both folders and subfolders, which are set apart by the "parent" attribute. A folder also has a name (unique in scope). The other two entities resemble what was described in the project requirements, meaning that every user has it's own username (unique) and password and every file has a name, an extension (together unique in the same scope), a creation date, a summary and a type. Every entity is then identified by a numeric ID.

4.1.1 ER diagram



4.1.2 Logical schema

```
User (id,           // [PK] User ID
      username,     // Username
      email,        // Email
      password)     // Password

Folder (id,         // [PK] Folder ID
        user,       // [FK] Folder owner
        name,       // Folder name
        creation,   // Folder creation date
        parent*)    // Parent folder (if subfolder)

File (id,           // [PK] File ID
      name,         // File name
      extension,    // File extension
      parent,       // [FK] Parent folder ID
      creation,     // Creation date
      summary,      // File summary
      type)         // File type
```

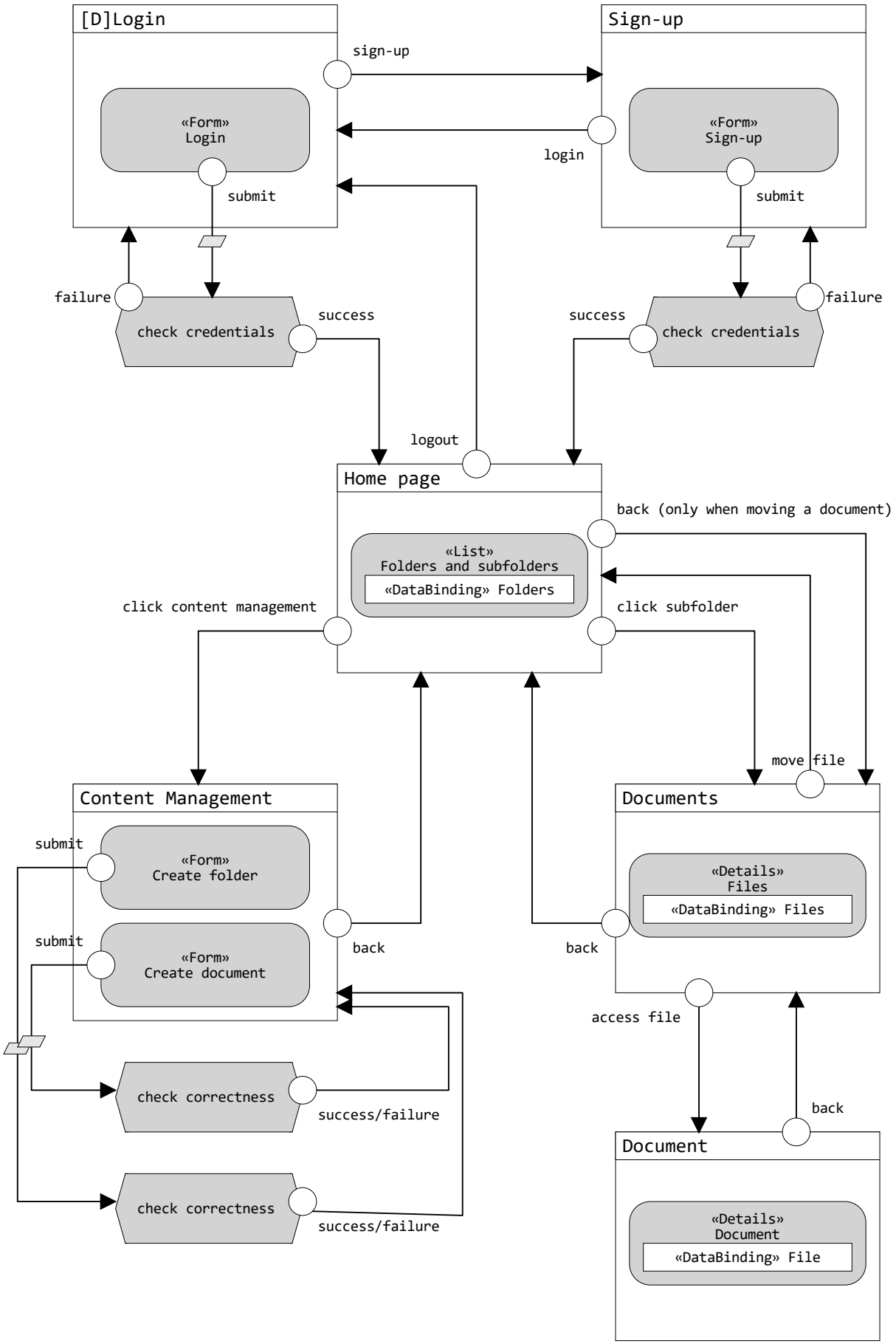
4.1.3 SQL

```sql
CREATE TABLE `User` (
    `id` int PRIMARY KEY,
    `username` varchar(45) NOT NULL,
    `email` varchar(45) NOT NULL,
    `password` varchar(45) NOT NULL
);

CREATE TABLE `Folder` (
    `id` int PRIMARY KEY,
    `user` int NOT NULL
        REFERENCES `User` (`id`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    `name` varchar(45) NOT NULL,
    `creation` date NOT NULL,
    `parent` int DEFAULT NULL
        REFERENCES `Folder` (`id`)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE `File` (
    `id` int PRIMARY KEY,
    `name` varchar(45) NOT NULL,
    `extension` varchar(10) NOT NULL,
    `parent` int NOT NULL
        REFERENCES `Folder` (`id`)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    `creation` date NOT NULL,
    `summary` varchar(250) NOT NULL,
    `type` varchar(15) NOT NULL,
);
```

4.2 Application design



5 Components

5.1 Model objects (beans)

- File
- Folder
- User

5.2 Data Access Objects (classes)

- FileDAO
  - getFilesFromFolder
  - getFile
  - createFile
  - exists
  - moveFile
- FolderDAO
  - getFolder
  - getFoldersFromUser
  - exists
  - createFolder
- UserDAO
  - checkCredentials
  - exists
  - createUser

5.3 Controllers (servlets)

5.3.1 Frontend

- File
- Home
- Login
- MoveFile
- NewItem
- Signup
- SubFolder

5.3.2 Backend

- Controller
- CreateFile
- CreateFolder
- Logout
- MoveFile

5.4 Views (templates)

- create.html
- document.html
- error.html
- folder.html
- home.html
- login.html
- moveFile.html
- signup.html

6 JavaScript

Client side			Server side		
Event	**Action**	**Controller**	**Event**	**Action**	**Controller**
login > form > submit	Data check	checkLogin()	POST username, password	Check validity	Login (servlet)
signup > form > submit	Data check	checkSignup()	POST username, mail, password, repeat password	Check validity	Signup (servlet)
home > load #1	View update with folders	home()	POST	Folders and subfolders search	Home (servlet)
home > load #2	View update with files	subfolder()	POST id	Check ownership and get files	SubFolder (servlet)
home > file > drop	Drop file into folder	dropFolder()	POST folderId, fileId	Check ownership and file movement	MoveFile (servlet)
home > file > drop	Drop file into recycle bin	dropBin()	×	×	×
home > file deletion modal > confirmation > click	File deletion modal confirmation	removeFile()	POST id	Check ownership and deletion	Deletion (servlet)
home > (sub)folder > drop	Drop folder into recycle bin	dropBin()	×	×	×
home > (sub)folder deletion modal > confirmation > click	Folder deletion modal confirmation	removeFolder()	POST id	Check ownership and deletion	Deletion (servlet)
home > file > click	Click on file	showFileDetails Modal()	POST id	Check ownership	File (servlet)
home > file add > click	Click on file creation	showFileModal()	×	×	×
home > file add modal > confirmation > click	Check data presence	createFile()	POST name, extension, parent, creation, summary, type	Check folder ownership, data validity and file creation	CreateFile (servlet)
home > subfolder add > click	Click on subfolder creation	showSubFolder Modal()	×	×	×
home > subfolder add modal > confirmation > click	Check data presence	createSubFolder()	POST name, parent, creation	Check data validity and file creation	CreateFolder (servlet)
home > folder add > click	Click on folder creation	showFolder Modal()	×	×	×
home > subfolder add modal > confirmation > click	Check data presence	createFolder()	POST name, parent, creation	Check folder ownership, data validity and file creation	CreateFolder (servlet)
home > logout > click	Logout	logout()	GET	User session deletion	Logout (servlet)

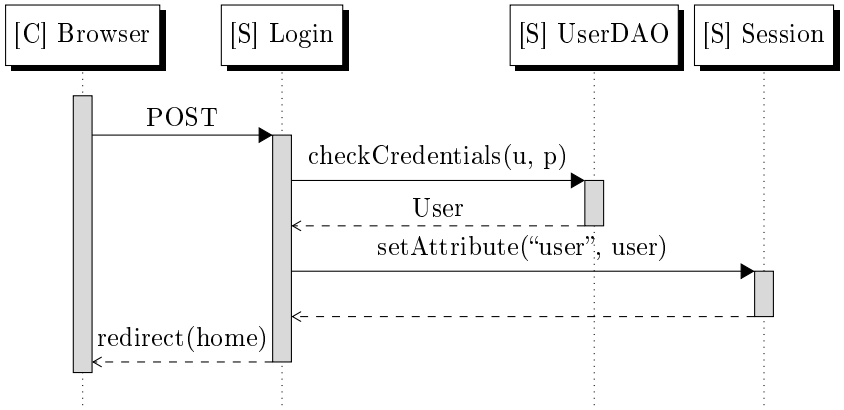
7 Events

Disclaimer For simplicity, all the following diagrams represent only the case of well formatted requests and correct value, without errors. In reality, of course, everything is checked, at least server-side, and a proper error message is sent back to the client if anything goes wrong.

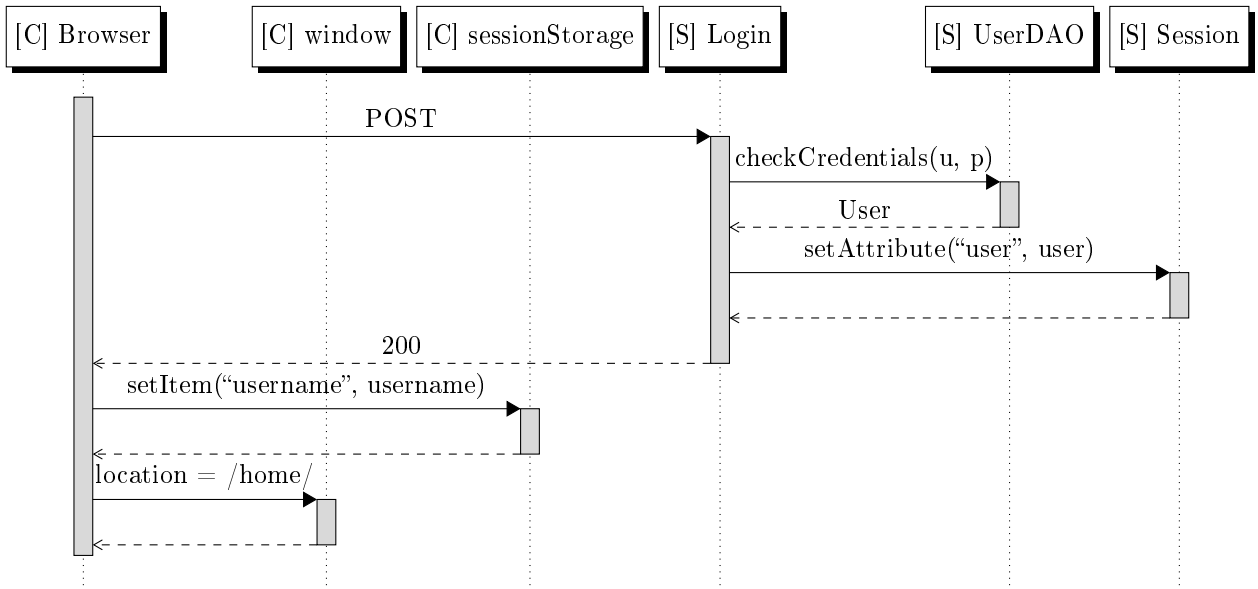
Note Items marked with `[C]` are client-sided, while items marked with `[S]` are server-sided.

7.1 Login

Pure HTML version

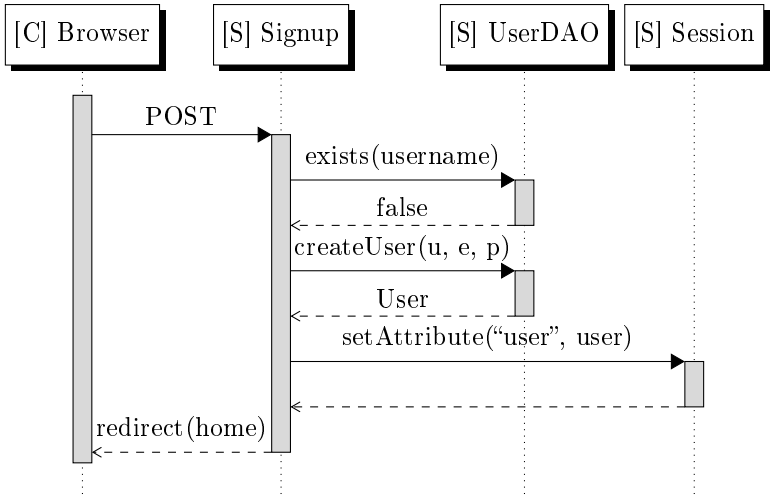


RIA version

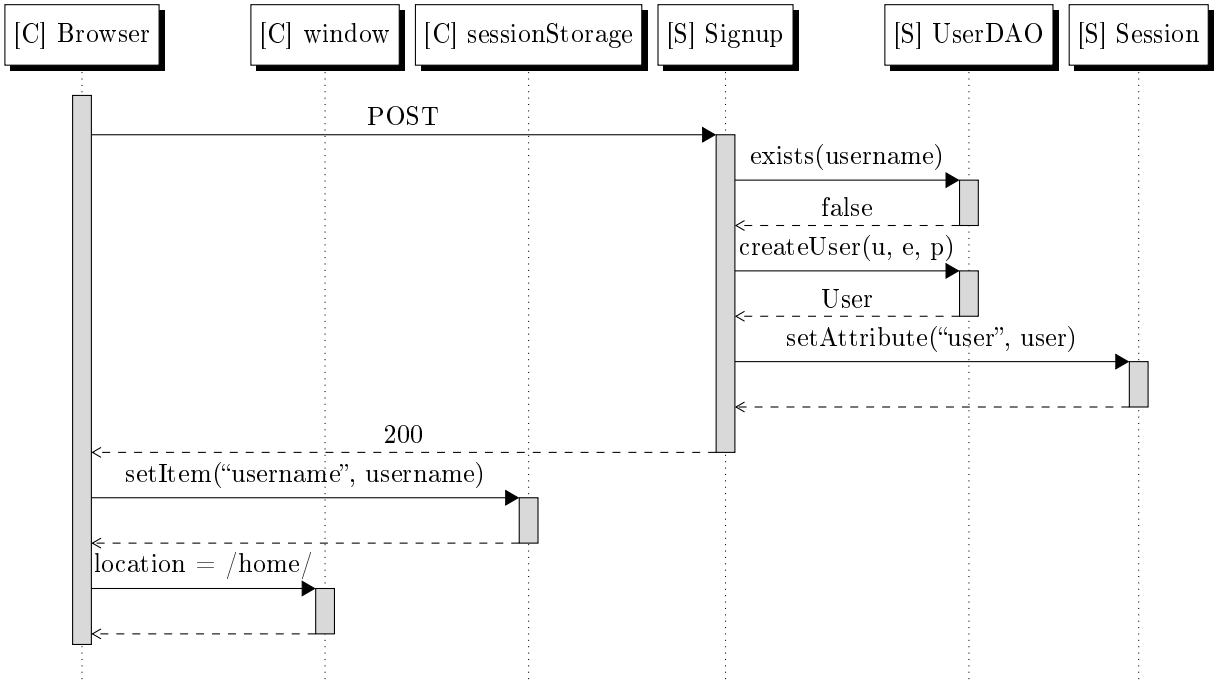


7.2 Sign-up

Pure HTML version

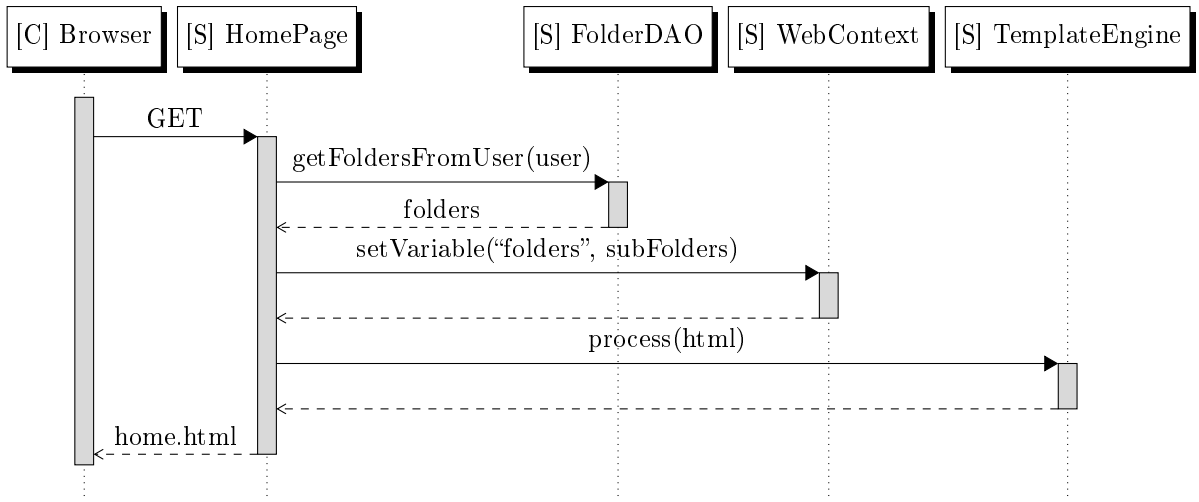


RIA version

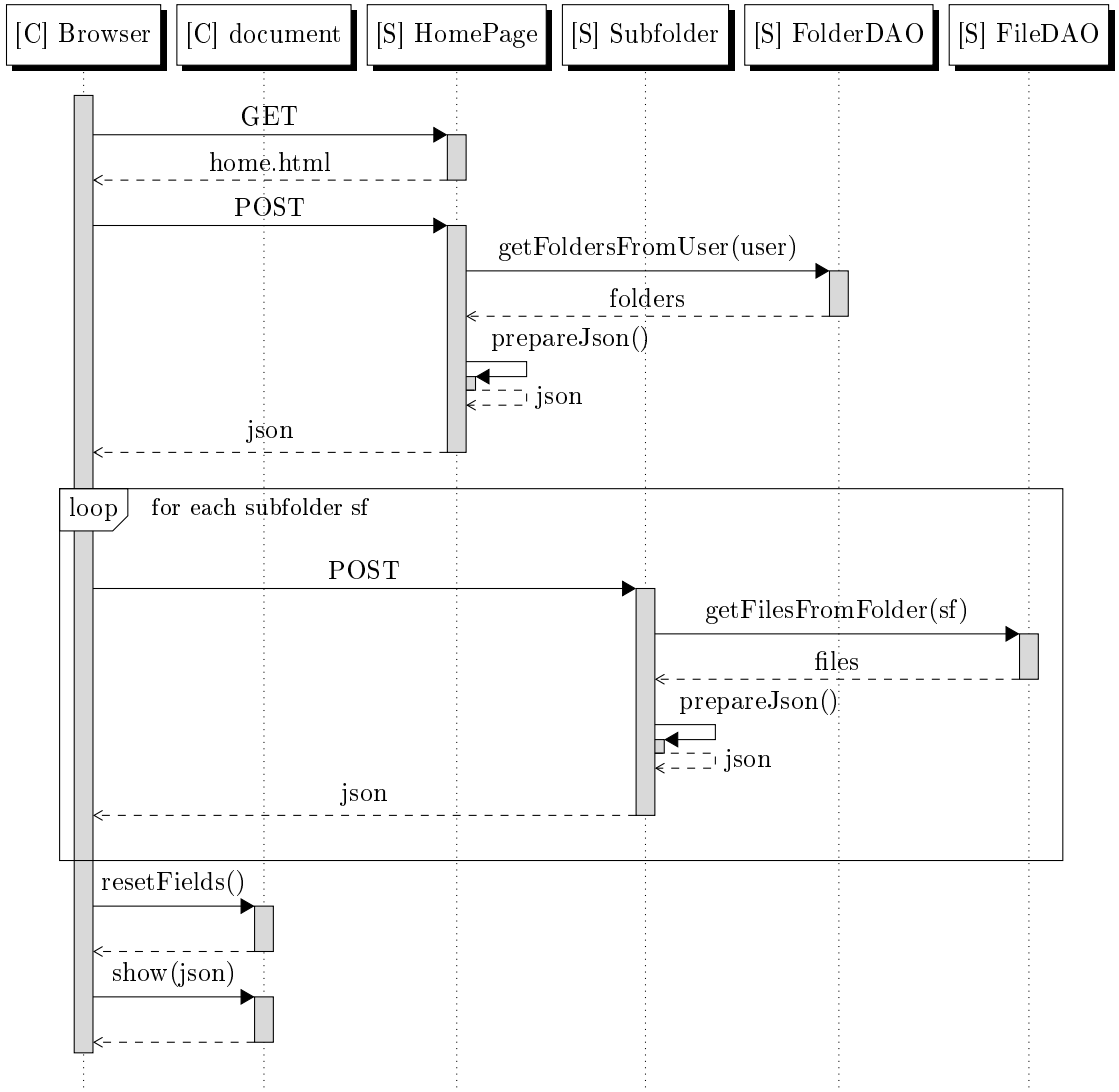


7.3 Home page access

Pure HTML version

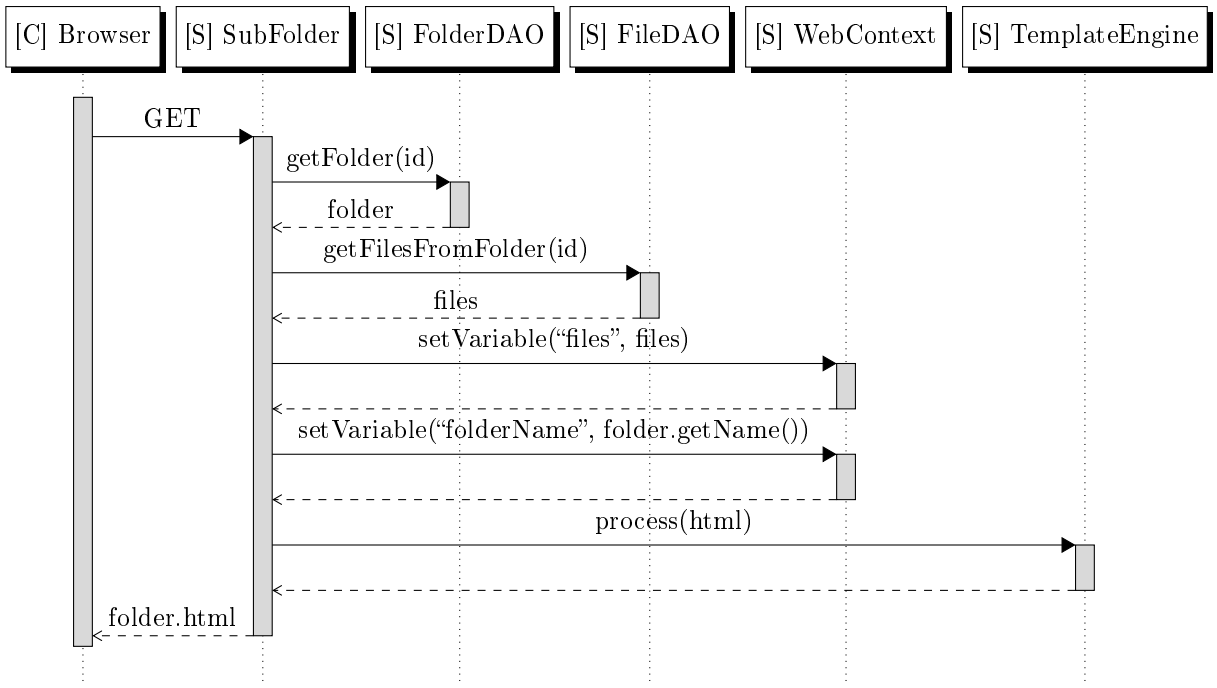


RIA version



7.4 Subfolder access

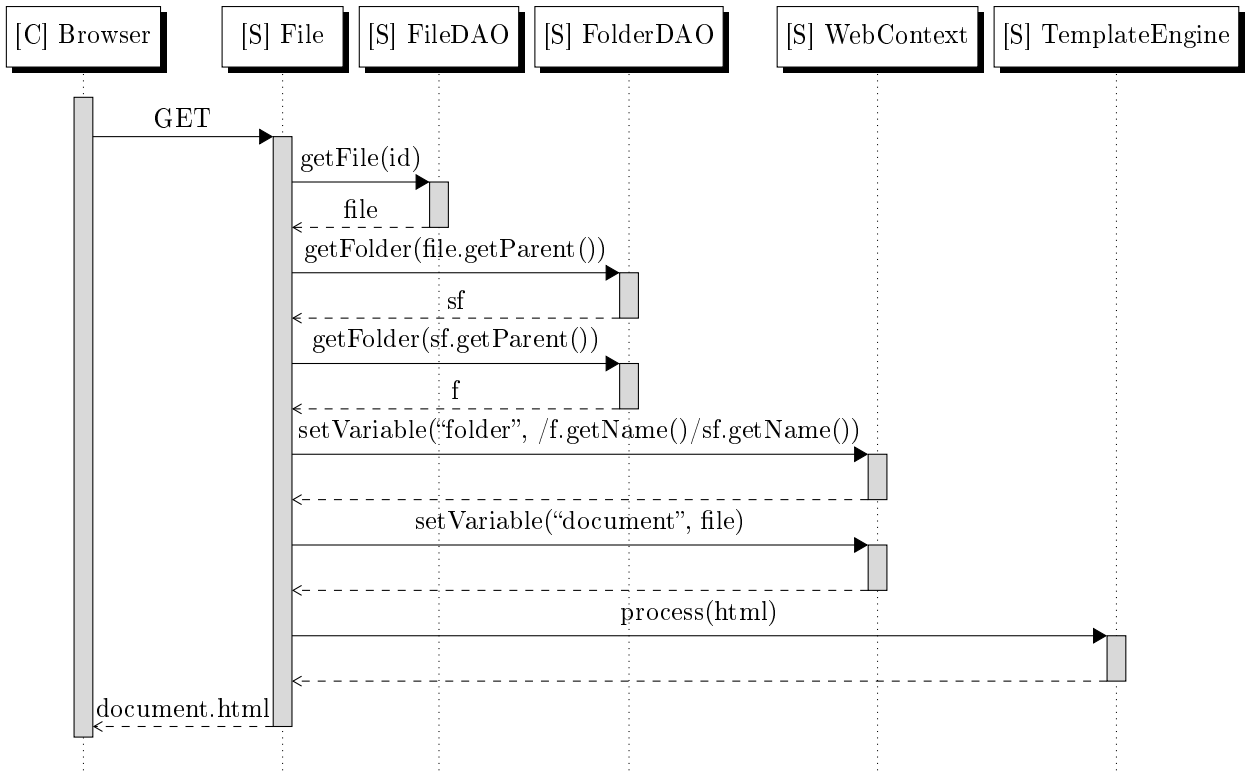
Pure HTML version



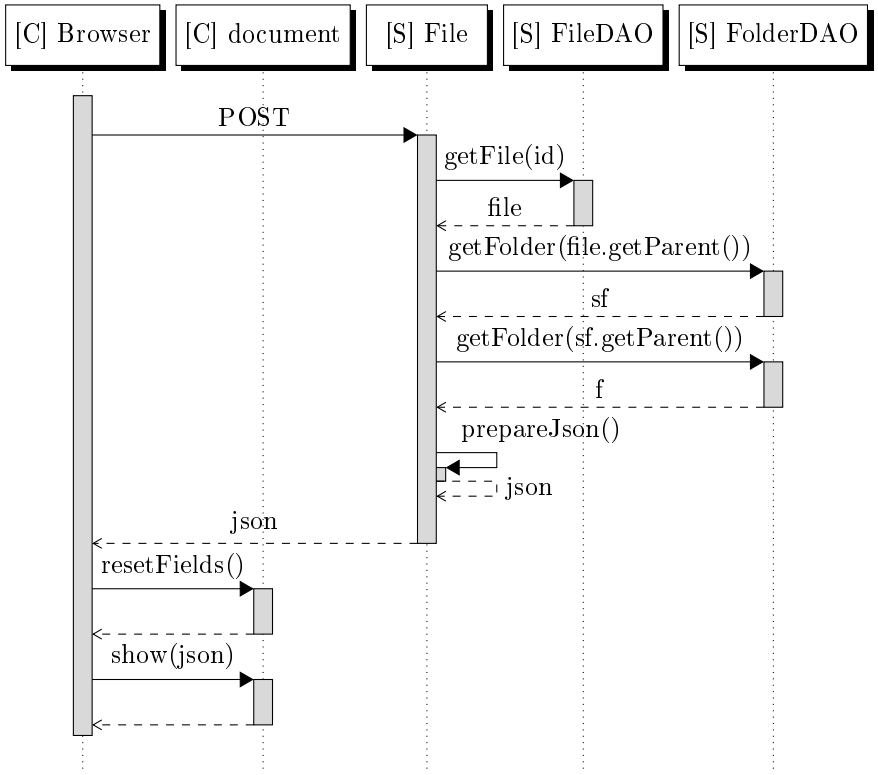
RIA version There is no correspondence between the pure HTML version and the RIA version, because in this case it's merged into the home page displaying process.

7.5 Document access

Pure HTML version

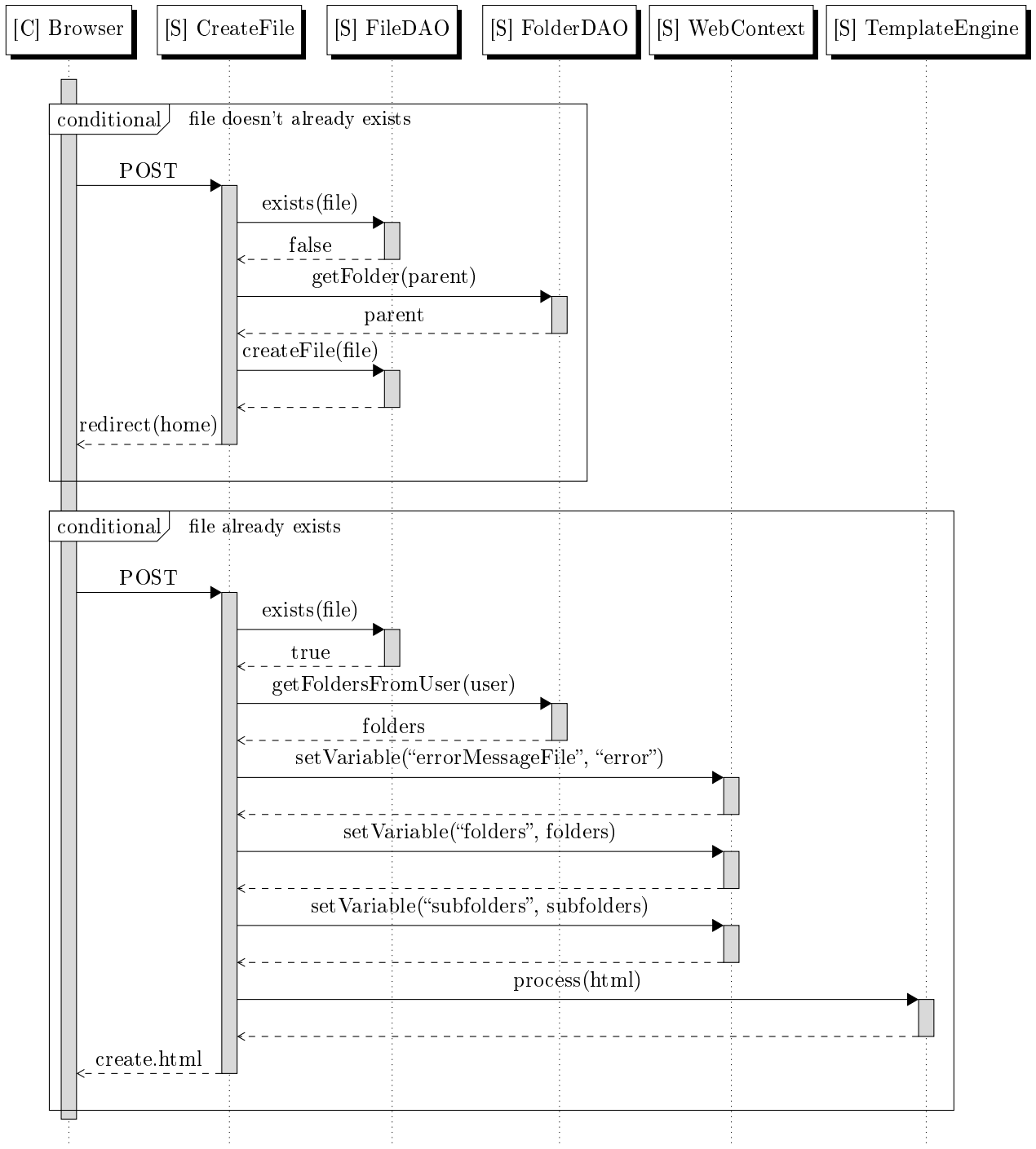


RIA version

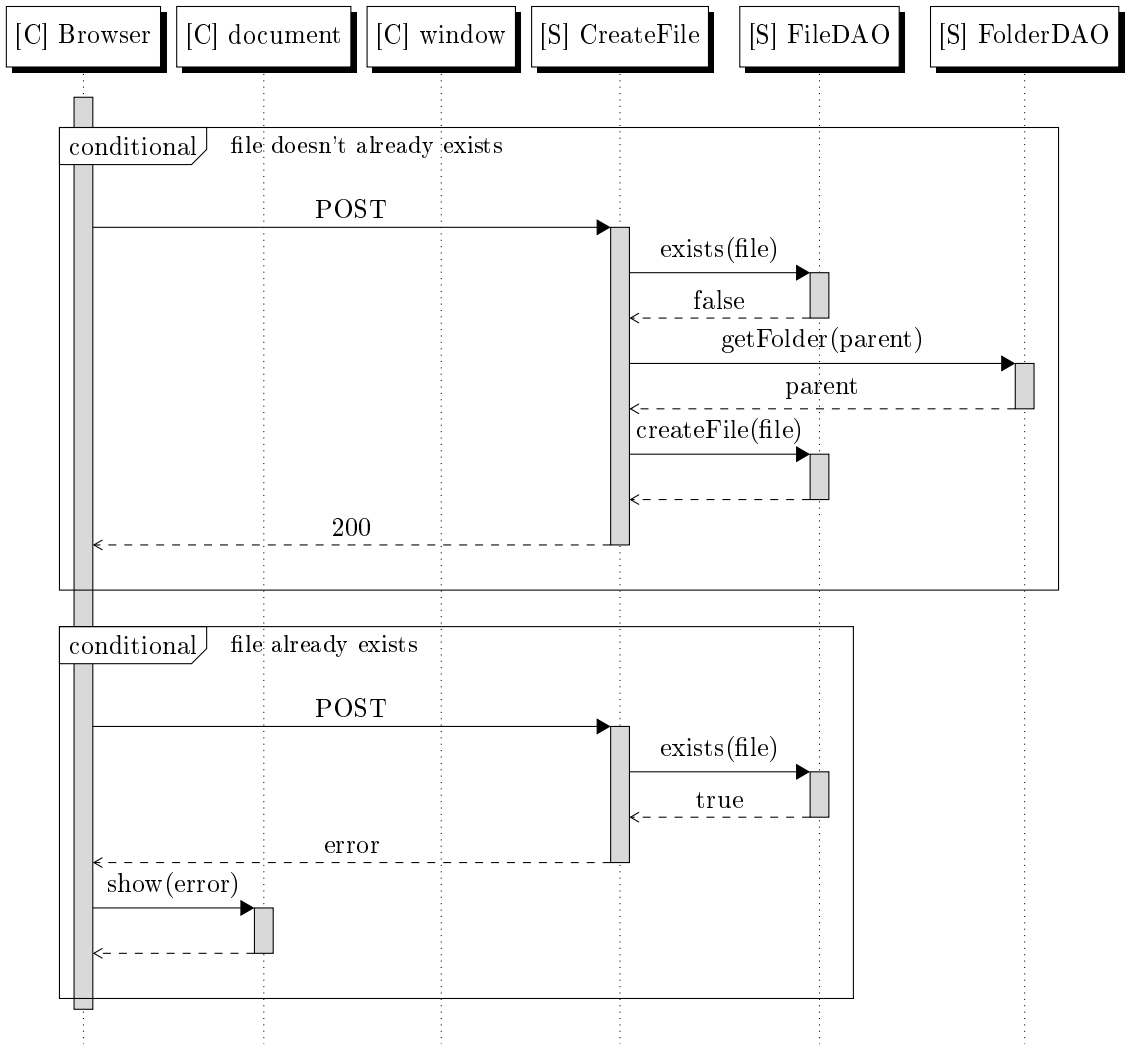


7.6 File creation

Pure HTML version

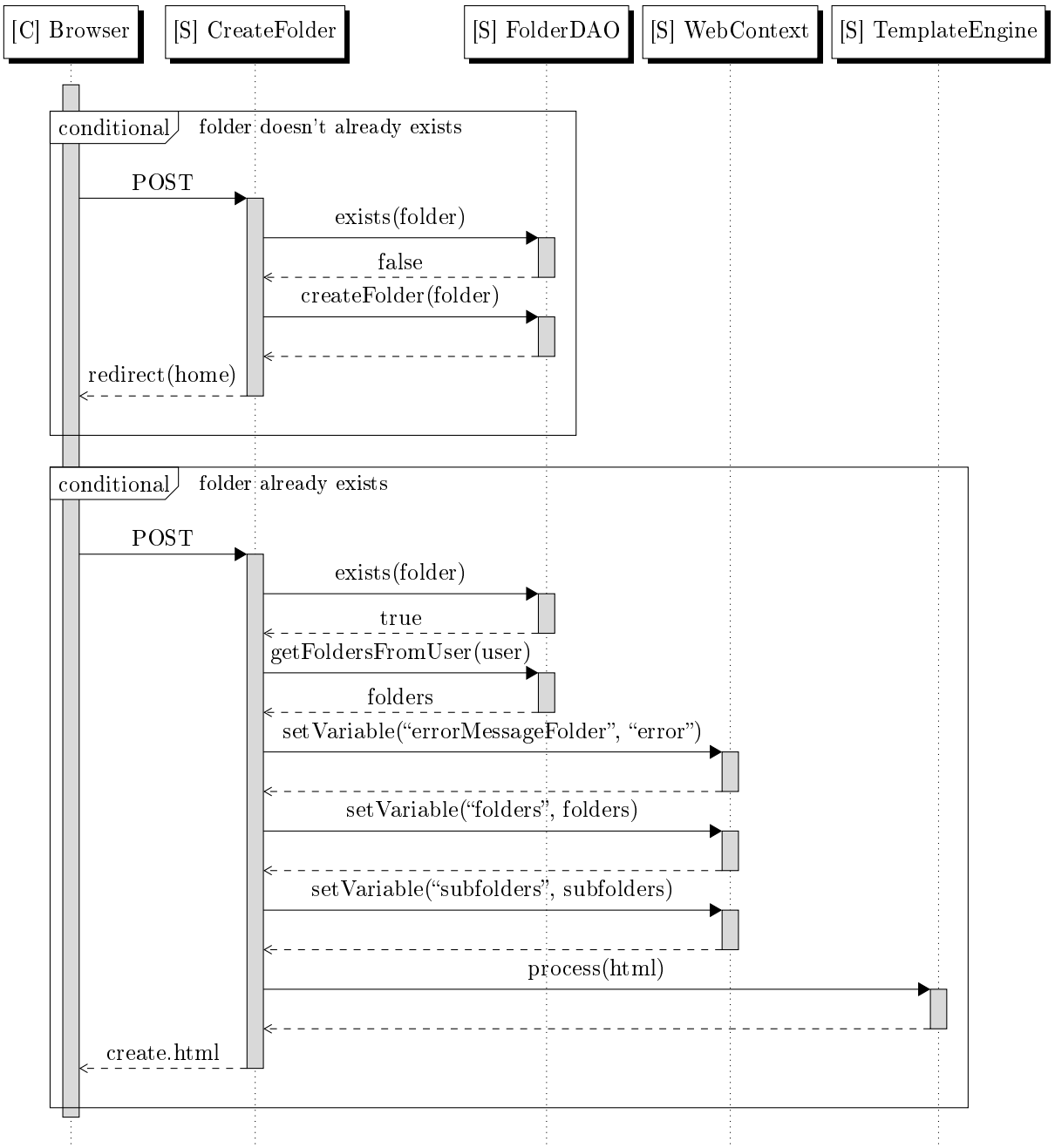


RIA version

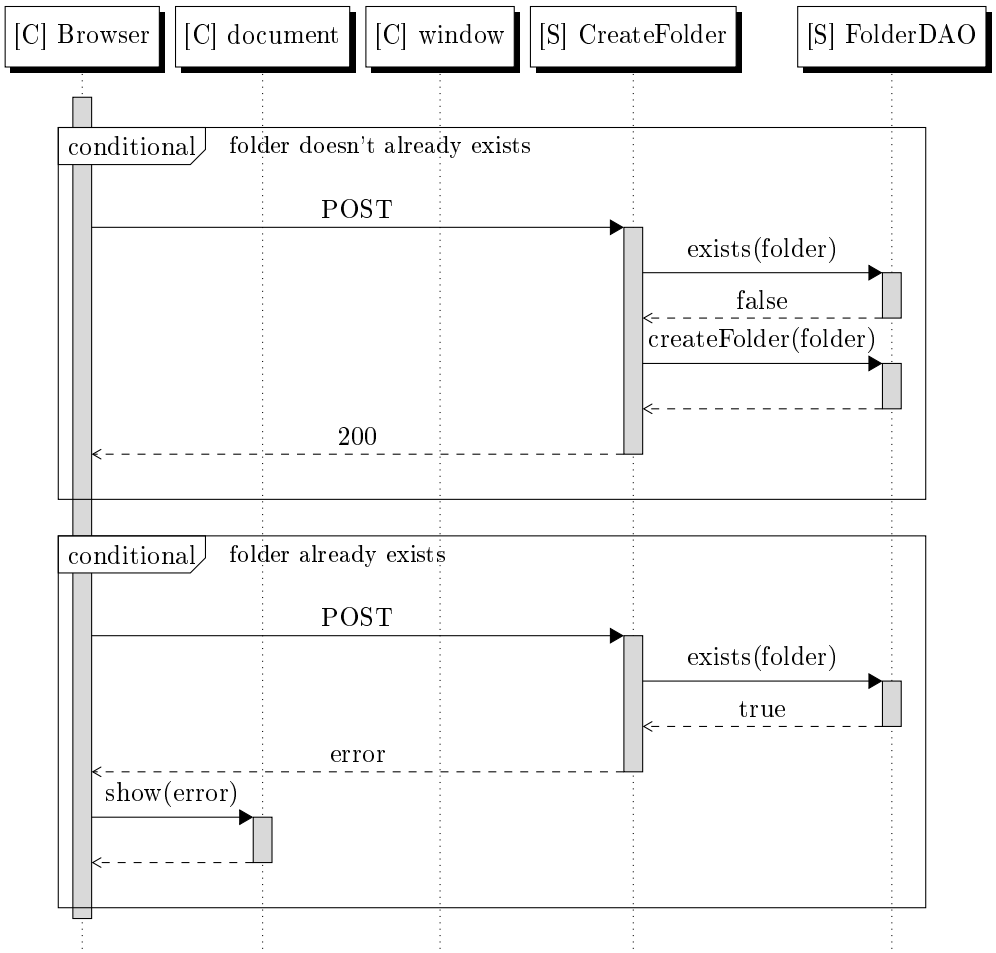

[C] Browser
[C] document
[C] window
[S] CreateFile
[S] FileDAO
[S] FolderDAO
conditional
file doesn't already exists
POST
exists(file)
false
getFolder(parent)
parent
createFile(file)
200
conditional
file already exists
POST
exists(file)
true
error
show(error)

7.7 Folder creation

Pure HTML version

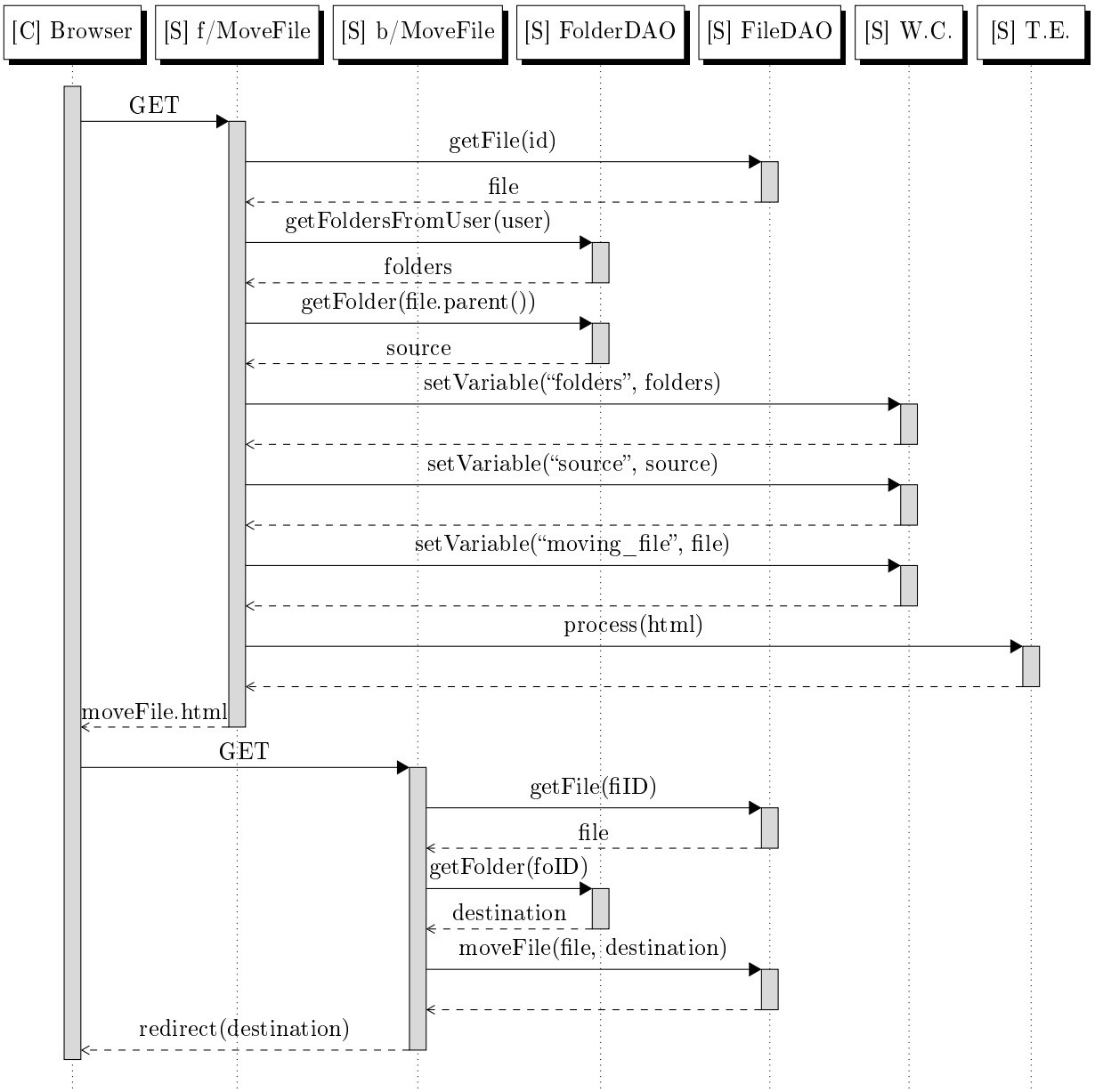


RIA version

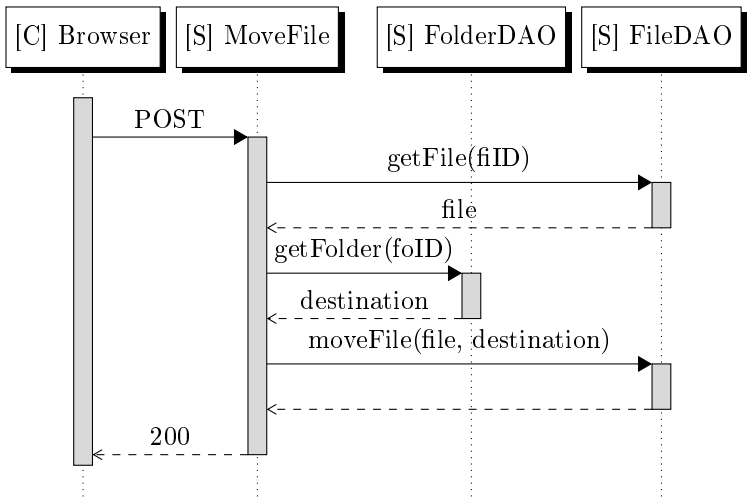

[C] Browser
[C] document
[C] window
[S] CreateFolder
[S] FolderDAO
conditional
folder doesn't already exists
POST
exists(folder)
false
createFolder(folder)
200
conditional
folder already exists
POST
exists(folder)
true
error
show(error)

7.8 File movement

Pure HTML version



RIA version



7.9 Deletion

Pure HTML version In this version there is no file or folder deletion.

RIA version