Progetto TIW – pure html
Gestione di immagini

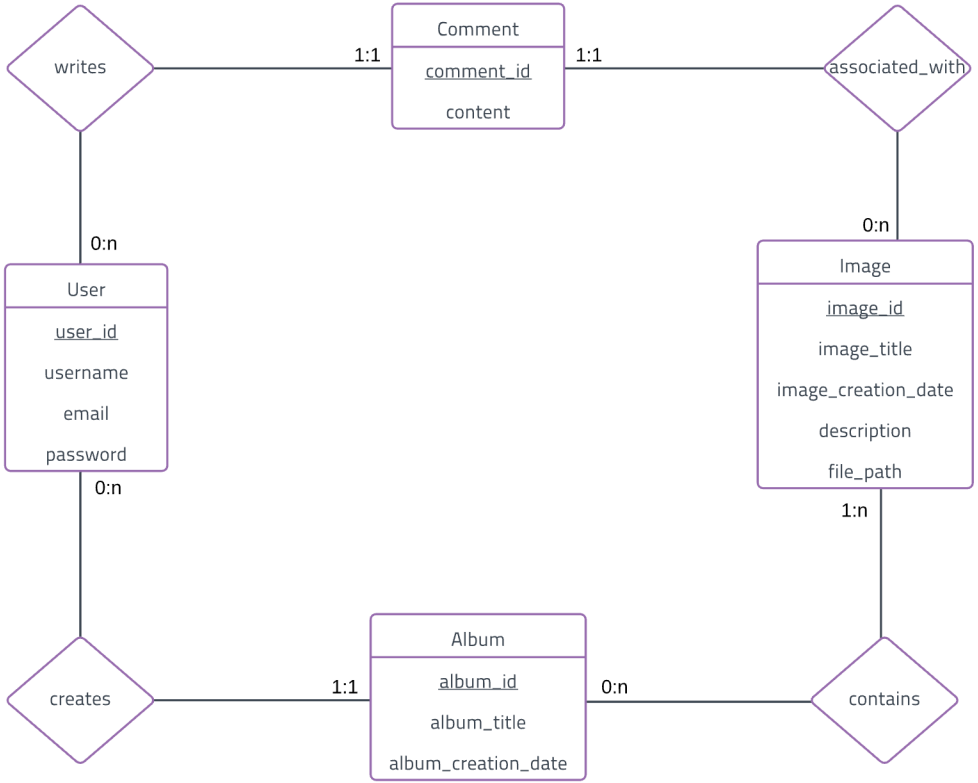
Kevin Ziroldi – Matteo Volpari

Analisi dei dati

Entities, **attributes**, **relationships**

Un'applicazione web consente la gestione di una galleria d'immagini. L'applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password". La registrazione controlla l'unicità dello username.
Ogni immagine è memorizzata come file nel file system del server su cui l'applicazione è rilasciata. Inoltre nella base di dati sono memorizzati i seguenti attributi: un titolo, una data di creazione, un testo descrittivo e il percorso del file dell'immagine nel file system del server. Le immagini sono associate all'utente che le carica.
L'utente può creare album dalla HOME PAGE e associare a questi le proprie immagini. Un album ha un titolo, il creatore e la data di creazione. La stessa immagine può appartenere a più di un album. Le immagini sono associate a uno o più commenti inseriti dagli utenti (dal proprietario o da altri utenti). Un commento ha un testo e il nome dell'utente che lo ha creato.
Quando l'utente accede all'HOME PAGE, questa presenta l'elenco degli album che ha creato e l'elenco degli album creati da altri utenti. Entrambi gli elenchi sono ordinati per data di creazione decrescente.

Schema ER


writes
1:1
Comment
comment_id
content
1:1
associated_with
0:n
User
user_id
username
email
password
0:n
0:n
Image
image_id
image_title
image_creation_date
description
file_path
1:n
creates
1:1
Album
album_id
album_title
album_creation_date
0:n
contains

Schema logico

User(user_id, username, email, password)
Comment(comment_id, content, user_id, image_id)
Comment.user_id -> **User**.user_id
Comment.image_id -> **Image**.image_id
Image(image_id, image_title, image_creation_date, description, file_path)
Album(album_id, album_title, album_creation_date, user_id)
Album.user_id -> **User** -> user_id
Contains(image_id, album_id)
Contains.image_id -> **Image**.image_id
Contains.album_id -> **Album**.album_id

Album

```
CREATE TABLE `Album` (
  `album_id` int NOT NULL AUTO_INCREMENT,
  `album_title` varchar(200) NOT NULL,
  `album_creation_date` date NOT NULL,
  `user_id` int NOT NULL,
  PRIMARY KEY (`album_id`),
  KEY `user_id_fk_idx` (`user_id`),
  CONSTRAINT `user_fk` FOREIGN KEY (`user_id`) REFERENCES `User`
(`user_id`) ON DELETE CASCADE ON UPDATE CASCADE
)
```

Comment

```
CREATE TABLE `Comment` (
  `comment_id` int NOT NULL AUTO_INCREMENT,
  `content` varchar(500) NOT NULL,
  `user_id` int NOT NULL,
  `image_id` int NOT NULL,
  PRIMARY KEY (`comment_id`),
  KEY `user_id_fk_idx` (`user_id`),
  KEY `image_fk_idx` (`image_id`),
  CONSTRAINT `image_fk` FOREIGN KEY (`image_id`) REFERENCES `Image`
(`image_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `user_id_fk` FOREIGN KEY (`user_id`) REFERENCES `User`
(`user_id`) ON DELETE CASCADE ON UPDATE CASCADE
)
```

Contains

```
CREATE TABLE `Contains` (
  `image_id` int NOT NULL,
  `album_id` int NOT NULL,
  PRIMARY KEY (`image_id`,`album_id`),
  KEY `album_id_fk_idx` (`album_id`),
  CONSTRAINT `album_id_fk` FOREIGN KEY (`album_id`) REFERENCES
`Album` (`album_id`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `image_id_fk` FOREIGN KEY (`image_id`) REFERENCES
`Image` (`image_id`) ON DELETE CASCADE ON UPDATE CASCADE
)
```

Image

```
CREATE TABLE `Image` (
  `image_id` int NOT NULL AUTO_INCREMENT,
  `image_title` varchar(45) NOT NULL,
  `image_creation_date` date NOT NULL,
  `description` varchar(500) NOT NULL,
  `file_path` varchar(260) NOT NULL,
  PRIMARY KEY (`image_id`)
)
```

User

```
CREATE TABLE `User` (
  `user_id` int NOT NULL AUTO_INCREMENT,
  `username` varchar(45) NOT NULL,
  `email` varchar(100) NOT NULL,
  `password` varchar(30) NOT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `username` (`username`),
  UNIQUE KEY `email` (`email`)
)
```

Analisi requisiti

Pages (views), view components, events, actions

Un'applicazione web consente la gestione di una galleria d'immagini. L'applicazione supporta registrazione e login mediante una pagina pubblica con opportune form. La registrazione controlla la validità sintattica dell'indirizzo di email e l'uguaglianza tra i campi "password" e "ripeti password". La registrazione controlla l'unicità dello username.
Ogni immagine è memorizzata come file nel file system del server su cui l'applicazione è rilasciata. Inoltre nella base di dati sono memorizzati i seguenti attributi: un titolo, una data di creazione, un testo descrittivo e il percorso del file dell'immagine nel file system del server. Le immagini sono associate all'utente che le carica.
L'utente può creare album dalla HOME PAGE e associare a questi le proprie immagini. Un album ha un titolo, il creatore e la data di creazione. La stessa immagine può appartenere a più di un album. Le immagini sono associate a uno o più commenti inseriti dagli utenti (dal proprietario o da altri utenti). Un commento ha un testo e il nome dell'utente che lo ha creato.

Quando l'utente accede all'HOME PAGE, questa presenta l'elenco degli album che ha creato e l'elenco degli album creati da altri utenti. Entrambi gli elenchi sono ordinati per data di creazione decrescente.
Quando l'utente clicca su un album che appare negli elenchi della HOME PAGE, appare la pagina ALBUM PAGE che contiene inizialmente una tabella di una riga e cinque colonne. Ogni cella contiene una miniatura (*thumbnail*) e il titolo dell'immagine. Se il numero di immagini non è un multiplo di 5 la tabella deve avere sempre 5 celle, lasciando quelle più a destra vuote. Le miniature sono ordinate da sinistra a destra per data decrescente. Se l'album contiene più di cinque immagini, sono disponibili comandi per vedere il precedente e successivo insieme di cinque immagini. Se la pagina ALBUM PAGE mostra il primo blocco d'immagini e ne esistono altre successive nell'ordinamento, compare a destra della riga il bottone SUCCESSIVE, che permette di vedere le successive cinque immagini. Se la pagina ALBUM PAGE mostra l'ultimo blocco d'immagini e ne esistono altre precedenti nell'ordinamento, compare a sinistra della riga il bottone PRECEDENTI, che permette di vedere le cinque immagini precedenti. Se la pagina ALBUM PAGE mostra un blocco d'immagini e ne esistono altre precedenti e successive nell'ordinamento, compare a destra della riga il bottone SUCCESSIVE, che permette di vedere le successive cinque immagini, e a sinistra il bottone PRECEDENTI, che permette di vedere le cinque immagini precedenti.

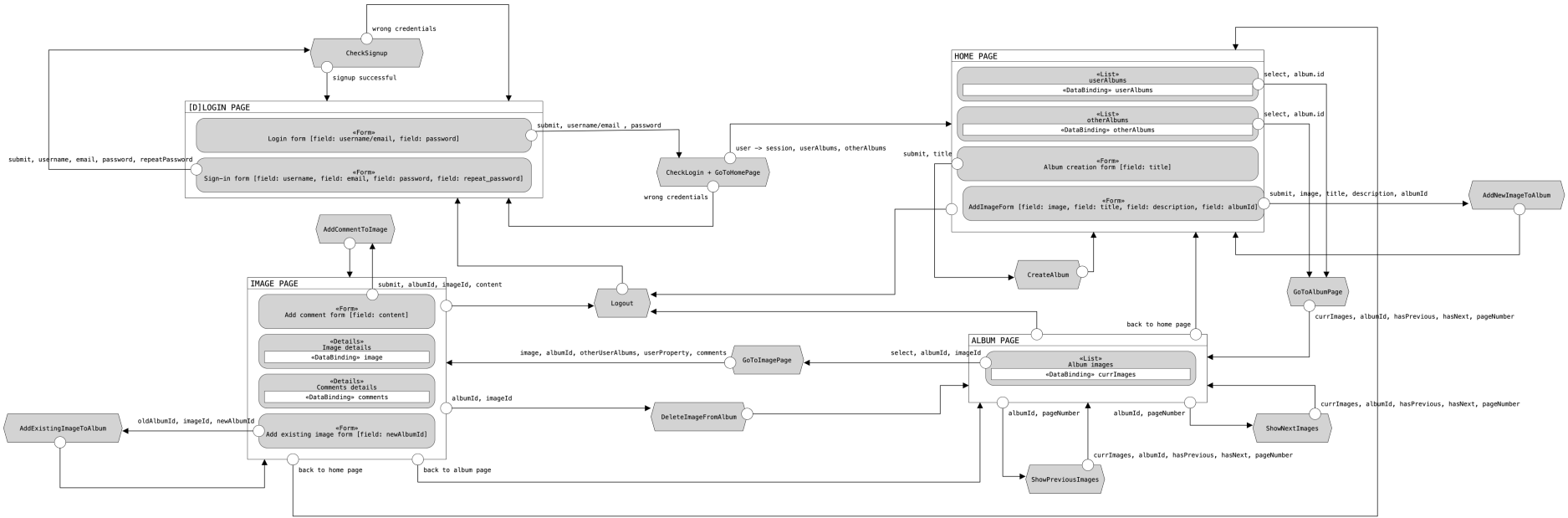
Quando l'utente seleziona una miniatura, una pagina IMAGE PAGE mostra tutti i dati dell'immagine scelta, tra cui la stessa immagine a grandezza naturale e i commenti eventualmente presenti. La pagina mostra anche una form per aggiungere un commento e un bottone per cancellare l'immagine e tutti i commenti ad essa associati. Il bottone di cancellazione appare solo se l'immagine appartiene all'utente. L'invio del commento con un bottone INVIA ripresenta la pagina IMAGE PAGE, con tutti i dati aggiornati della stessa immagine.

La pagina IMAGE PAGE contiene collegamenti per tornare all'HOME PAGE e alla pagina ALBUM PAGE. La pagina ALBUM PAGE contiene un collegamento per tornare all'HOME PAGE. L'applicazione consente il logout dell'utente.

Completamento delle specifiche

- Tutti i campi delle form sono obbligatori.
- La pagina di default è quella che contiene la form di login e di signup.
- Quando un nuovo utente si registra, prima di accedere alla Home page, deve effettuare il login.
- Un utente può loggarsi usando email e password oppure usando username e password.
- È possibile eseguire il logout da ognuna delle pagine.
- Se l'utente non è loggato e tenta di accedere a una pagina diversa da quella di login, viene reindirizzato alla pagina di login.
- Se l'utente è loggato e prova ad eseguire un nuovo login o una nuova registrazione, viene fatto il logout automatico dall'utente precedente.
- Un utente non può creare più album con lo stesso nome. Utenti diversi possono avere album con stesso nome.
- Al momento della creazione, un album può essere vuoto. È possibile aggiungere immagini dal proprio file system all'album dalla home page. È possibile aggiungere immagini già caricate in altri album ad nuovo album dalla image page.
- Un utente può inserire, tra le immagini già presenti sul server, solamente le sue immagini.
- Se l'immagine appartiene a più album, al momento della cancellazione, viene cancellata solo da un album.

Application design



Components

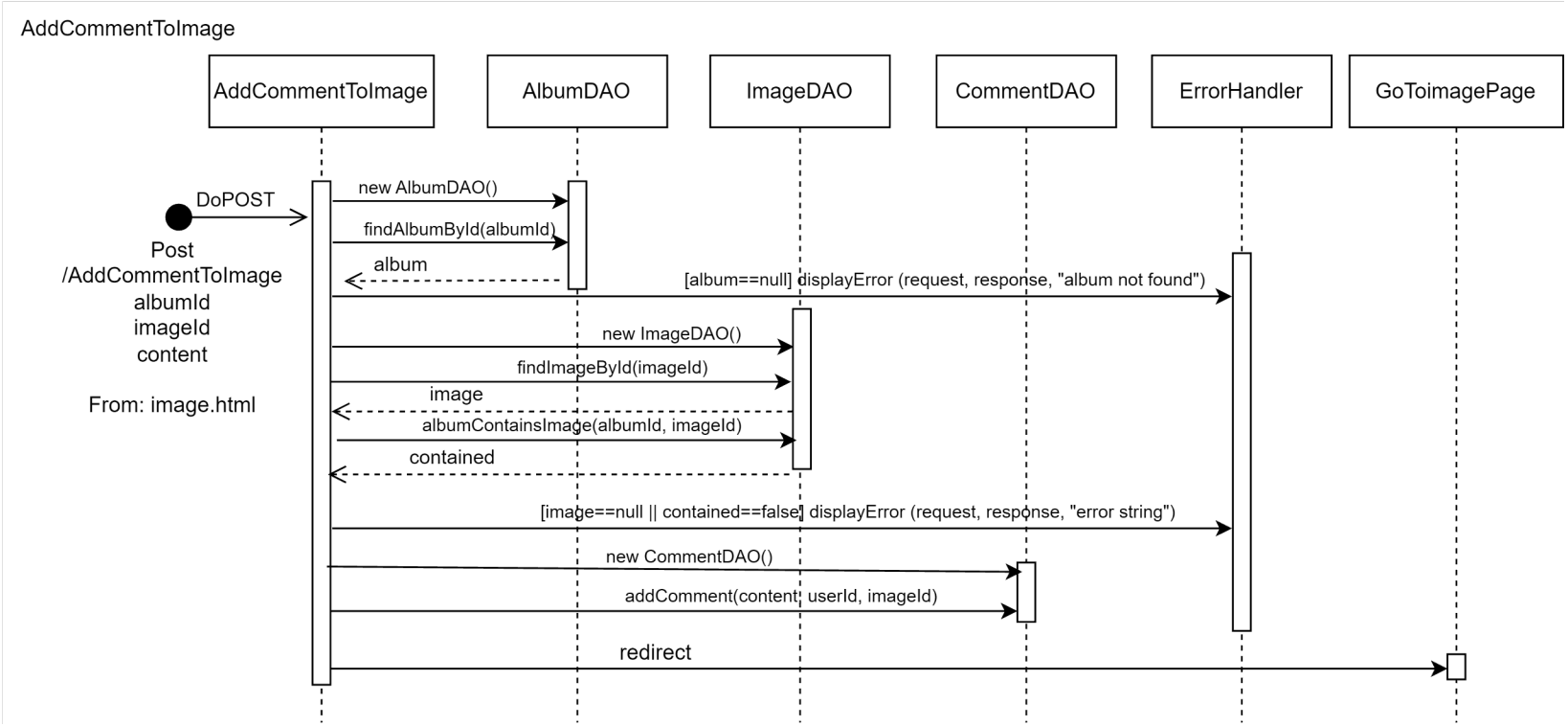
- **Model objects (Beans)**
  - Album
  - Comment
  - Image
  - User

- **Data Access Objects (DAO)**
  - AlbumDAO
    - findAlbumsByUserSorted(int userId)
    - findAlbumsOfOthersSorted(int userId)
    - findAlbumById(int albumId)
    - createAlbum(String albumTitle, int userId)
    - getUserIdByAlbumId(int albumId)
  - CommentDAO
    - findCommentsByImageId(int imageId)
    - addComment(String content, int userId, int imageId)
    - getUsernameByCommentId(int commentId)
  - ImageDAO
    - addNewImageToAlbum(String title, String description, String imagePath, int albumId)
    - findImageById(int imageId)
    - findImagesForAlbum(int albumId)
    - deleteImageFromAlbum(int albumId, int imageId)
    - getUserIdByImageId(int imageId)
    - addExistingImageToAlbum(int albumId, int imageId)
    - albumContainsImage(int albumId, int imageId)
  - UserDAO
    - checkCredentials(String usr, String pwd)
    - getUserByUsername(String username)
    - getUserByEmail(String email)
    - registerUser(String username, String email, String password)

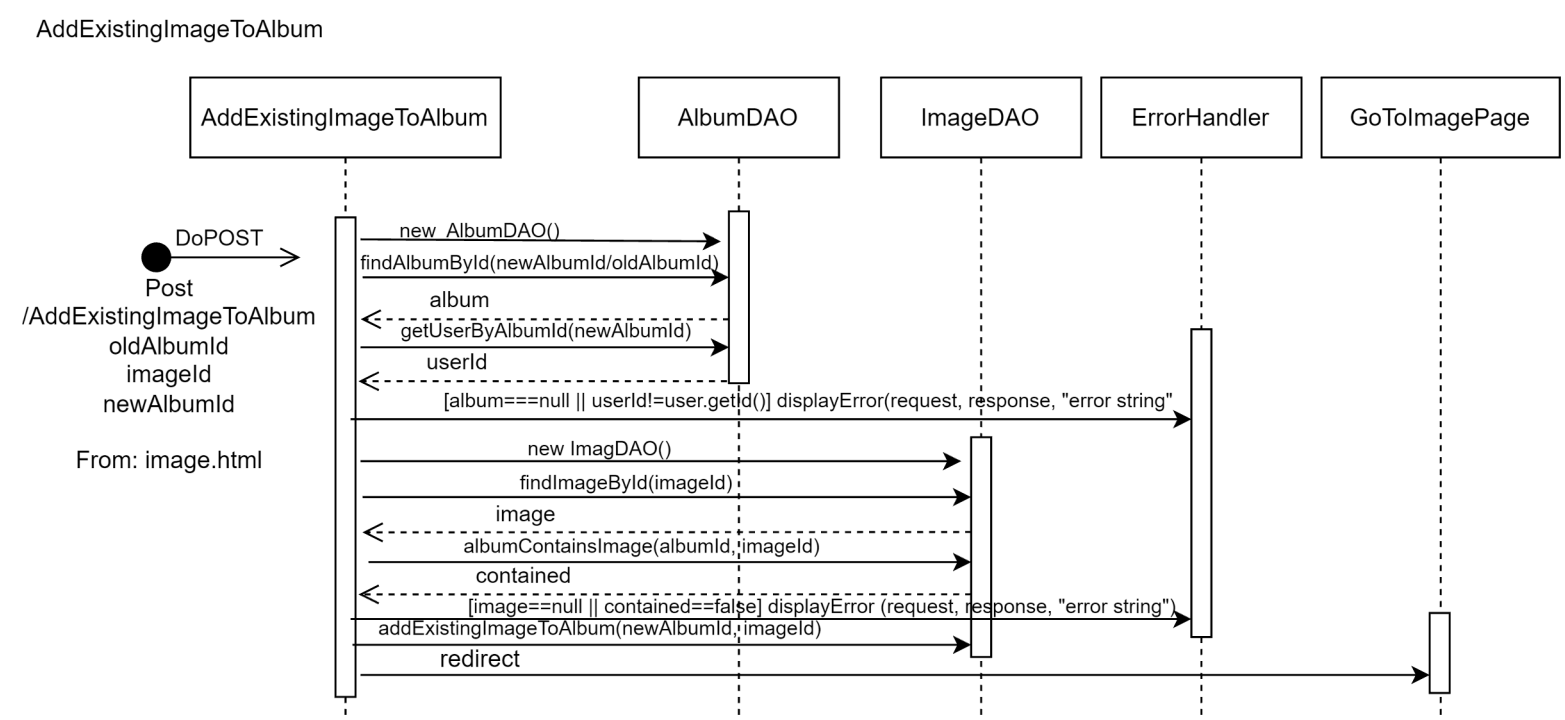
Components

- **Controllers (servlets)**
  - AddCommentToImage
  - AddExistingImageToAlbum
  - AddNewImageToAlbum
  - CheckLogin
  - CheckSignup
  - CreateAlbum
  - DeleteImageFromAlbum
  - DisplayImage
  - GoToAlbumPage
  - GoToHomePage
  - GoToImagePage
  - Logout
  - ShowNextImages
  - ShowPreviousImages
- **Views** (Templates)
  - login.html
  - home.hmtl
  - album.html
  - image.html
  - error.html
- **Filters**
  - CheckUserLogged
- **Utils**
  - ConnectionHandler
    - getConnection
    - closeConnection
  - ErrorHandler
    - displayError
  - TemplateHandler
    - getEngine

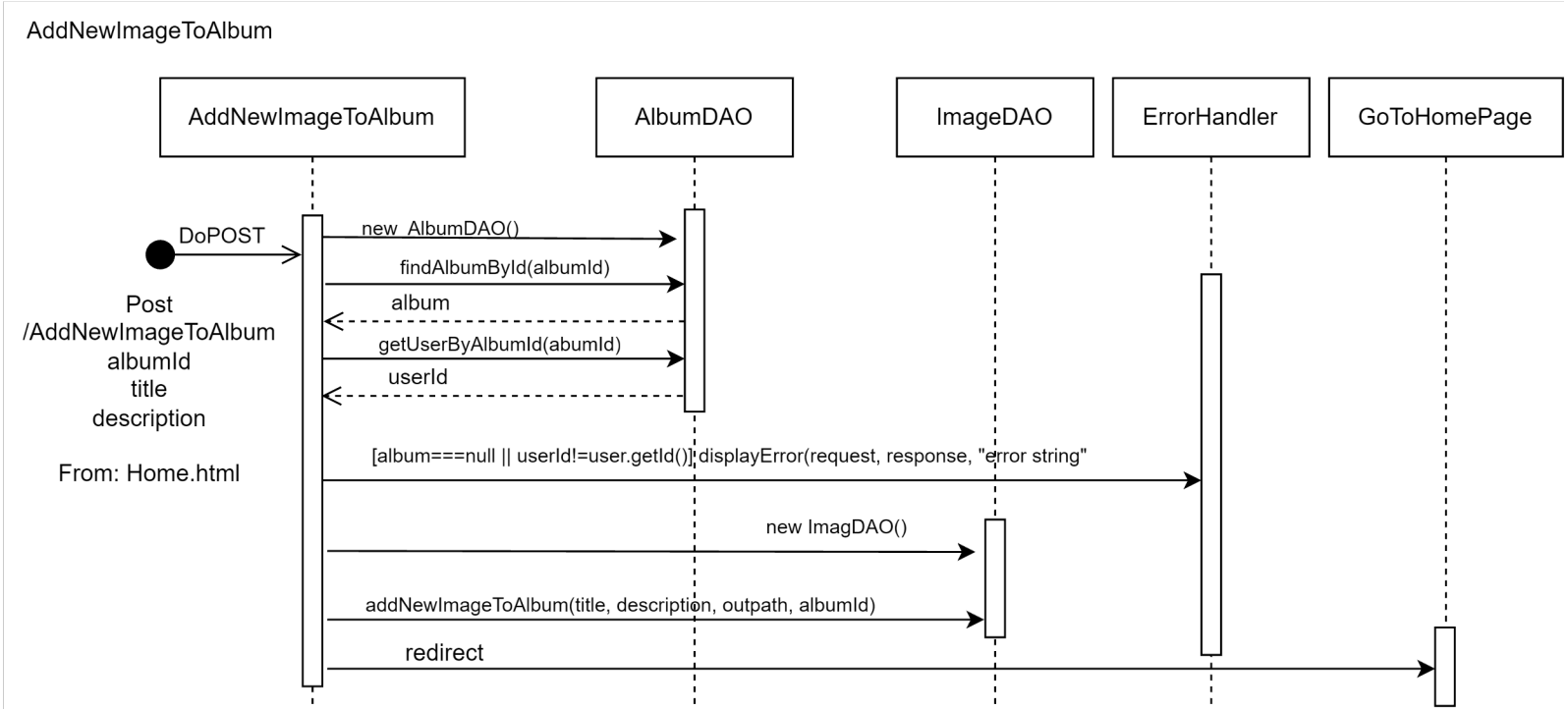
Add comment


AddCommentToImage
AddCommentToImage
AlbumDAO
ImageDAO
CommentDAO
ErrorHandler
GoToimagePage
DoPOST
Post
/AddCommentToImage
albumId
imageId
content
From: image.html
new AlbumDAO()
findAlbumById(albumId)
album
[album==null] displayError (request, response, "album not found")
new ImageDAO()
findImageById(imageId)
image
albumContainsImage(albumId, imageId)
contained
[image==null || contained==false] displayError (request, response, "error string")
new CommentDAO()
addComment(content, userId, imageId)
redirect

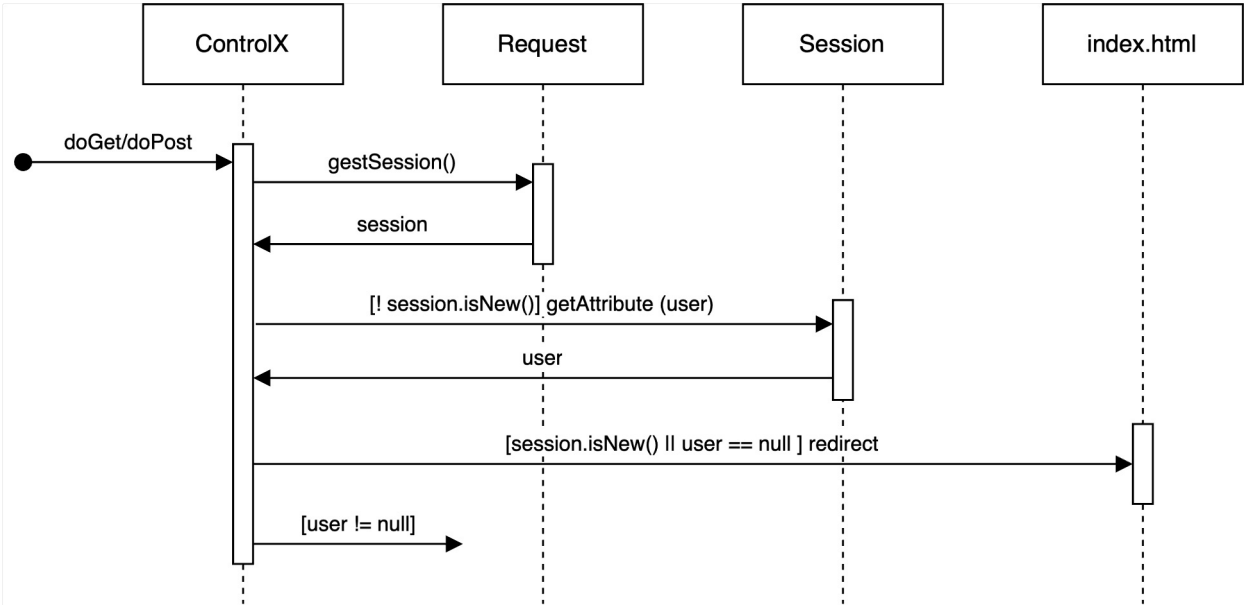
Add existing image to album



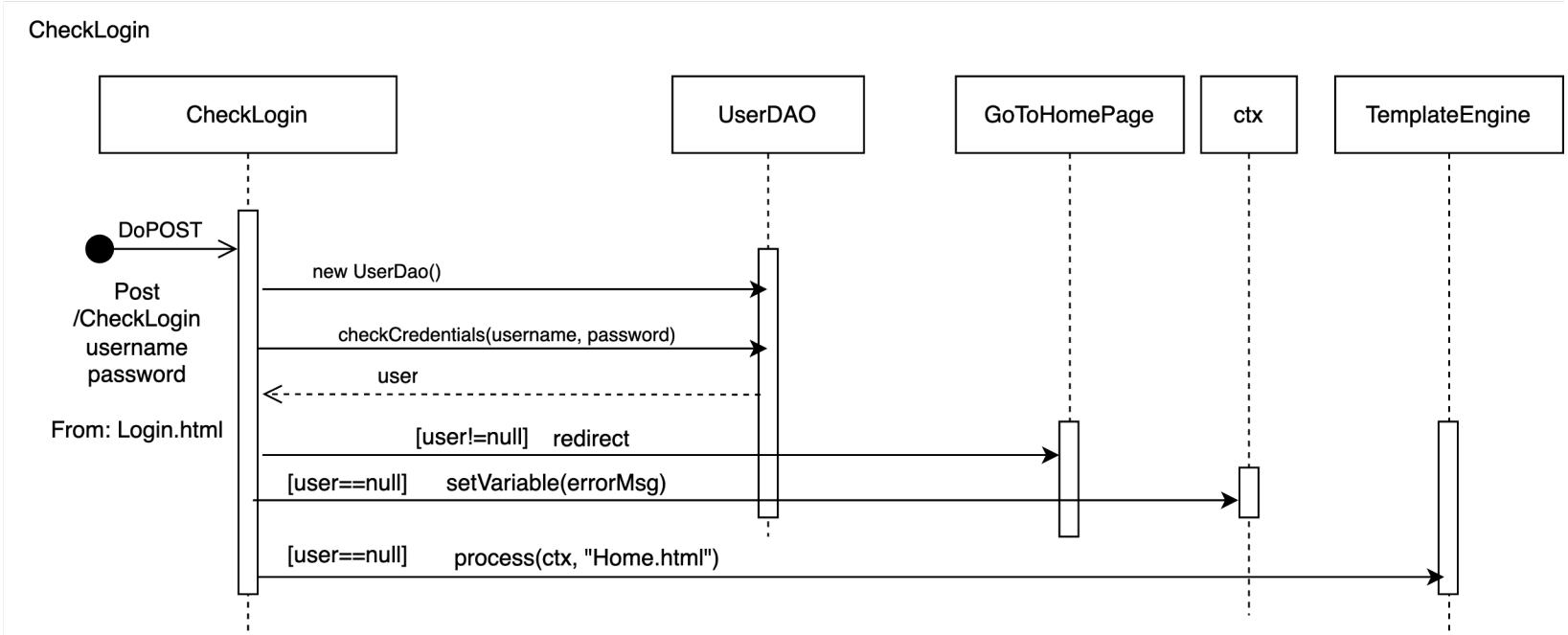
Add new image to album



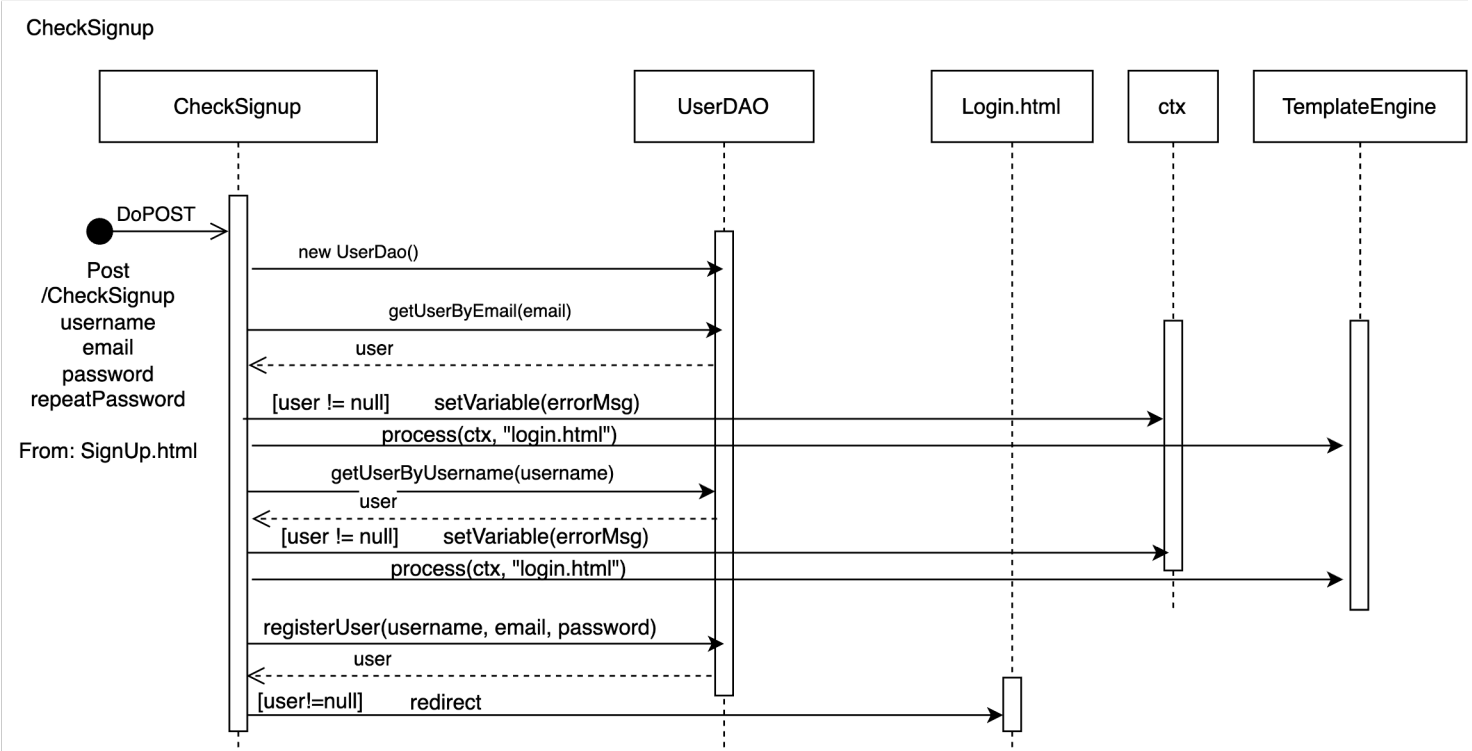
Check if the user is logged



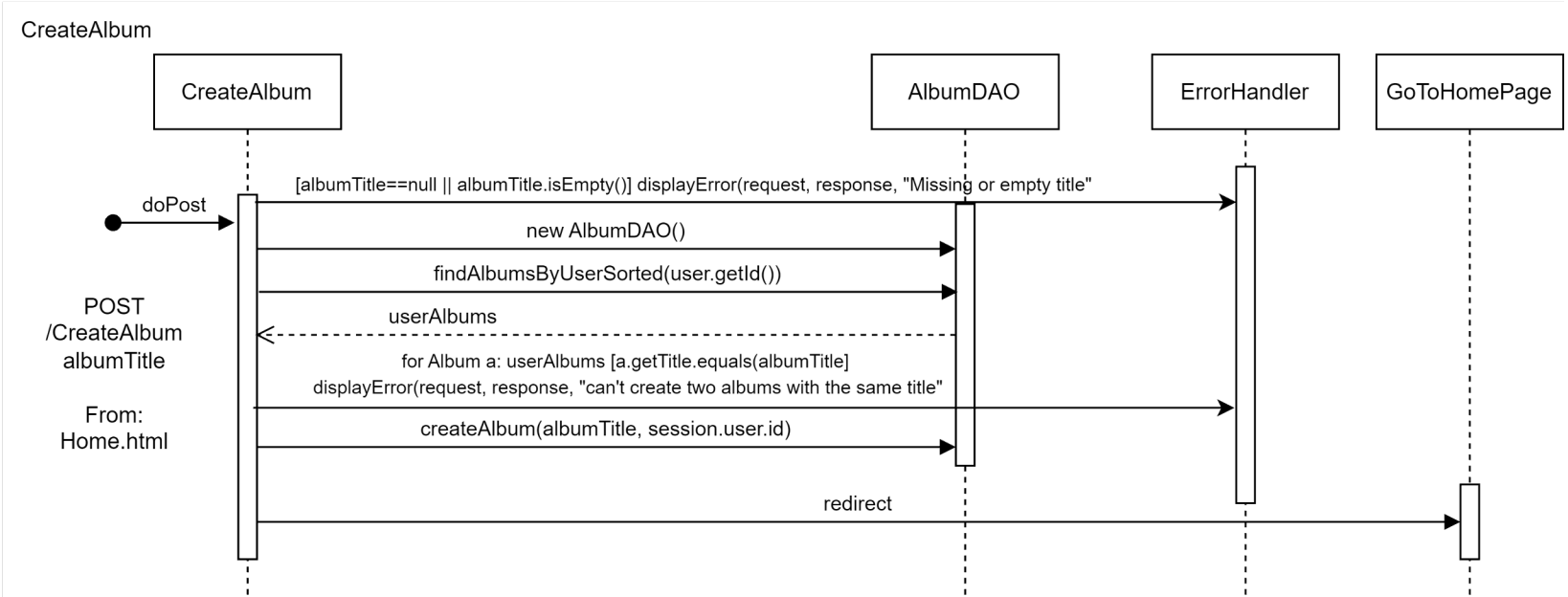
Check login


CheckLogin
CheckLogin
UserDAO
GoToHomePage
ctx
TemplateEngine
DoPOST
Post
/CheckLogin
username
password
From: Login.html
new UserDao()
checkCredentials(username, password)
user
[user!=null] redirect
[user==null] setVariable(errorMsg)
[user==null] process(ctx, "Home.html")

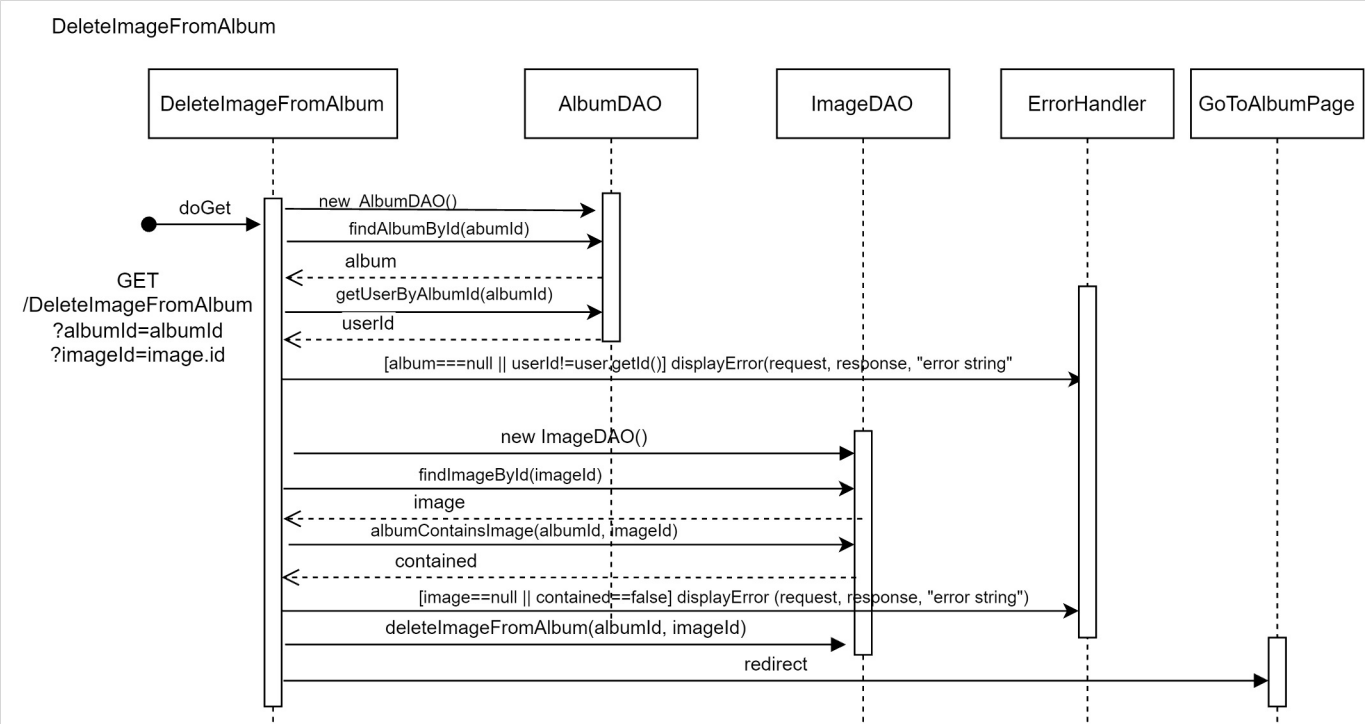
Check signup



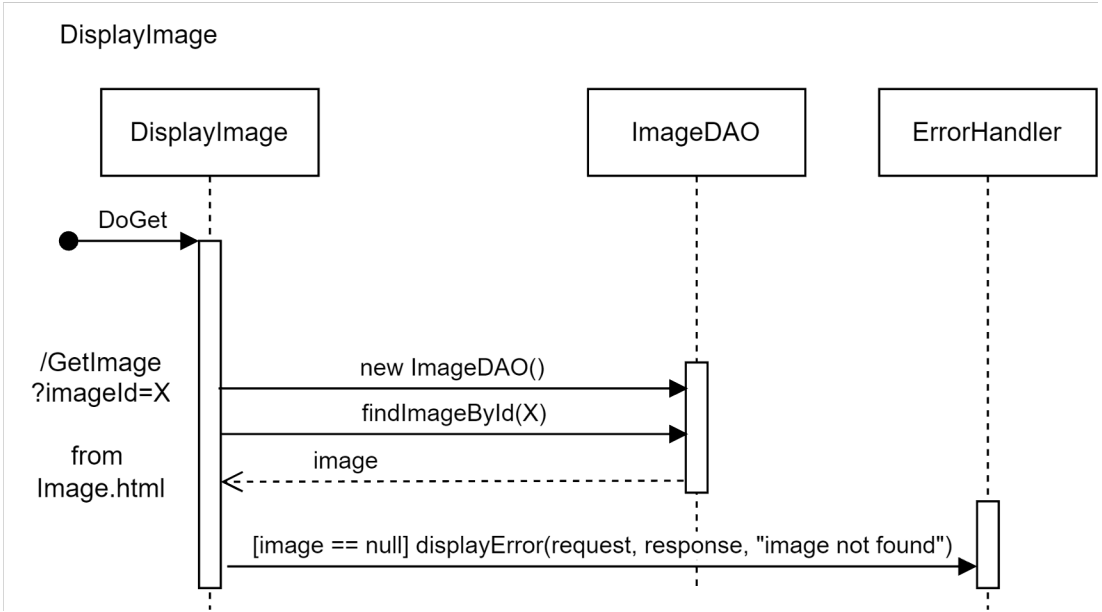
Create album



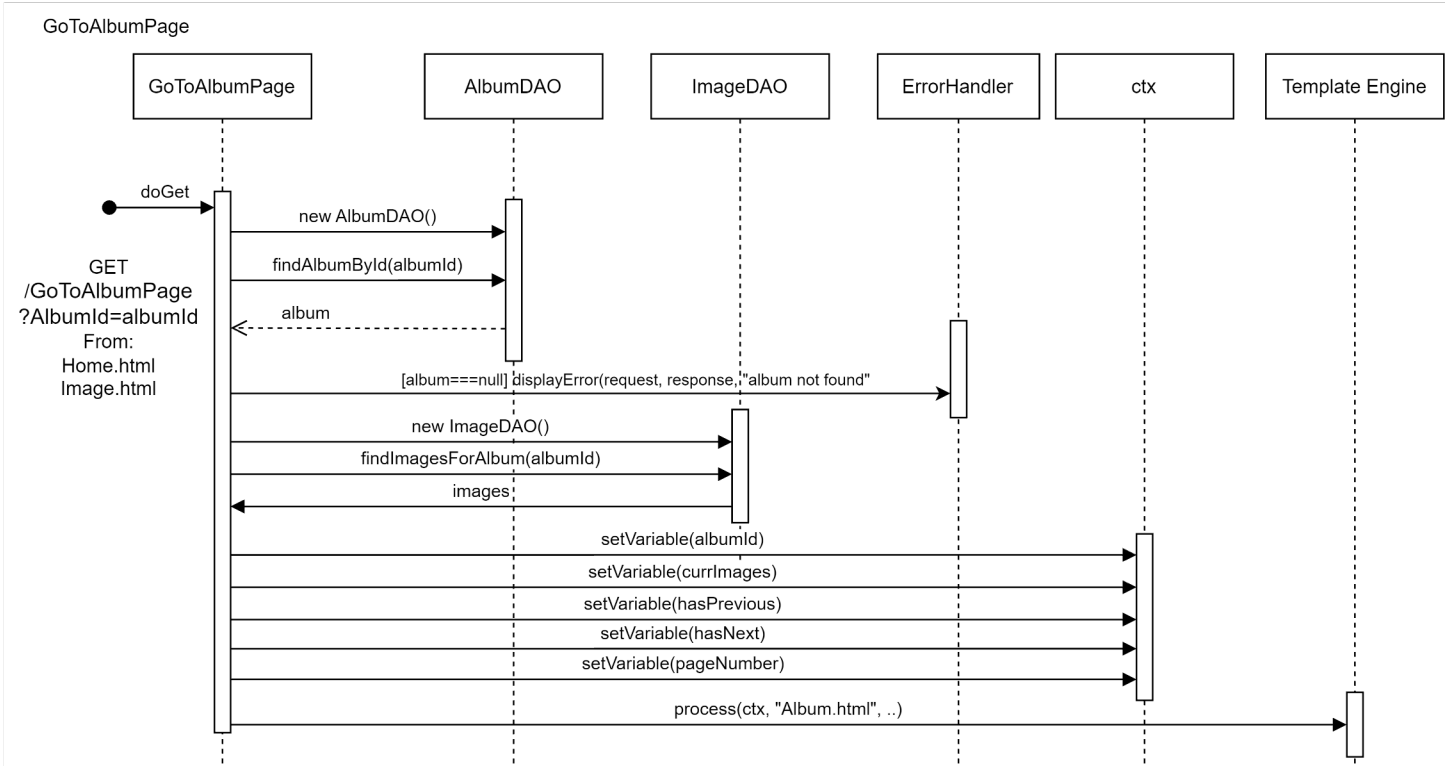
Delete image from album



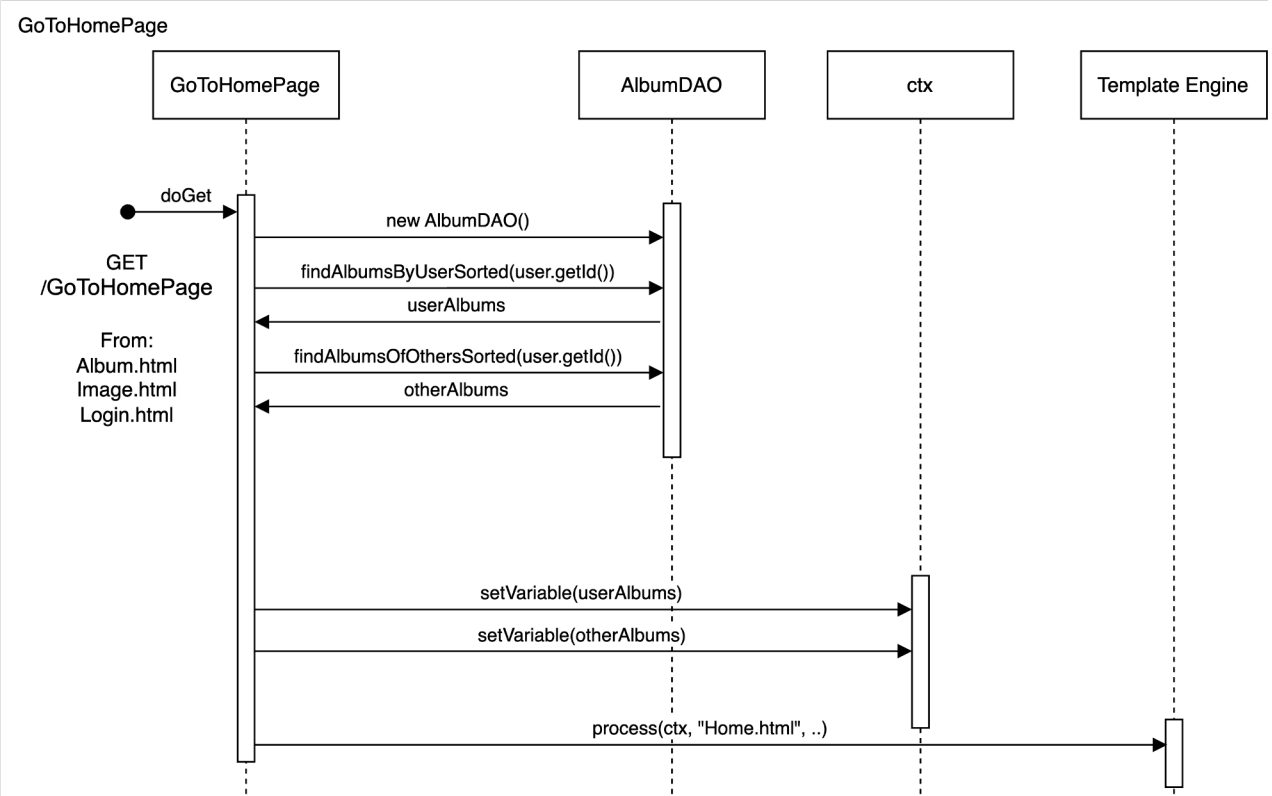
Display image



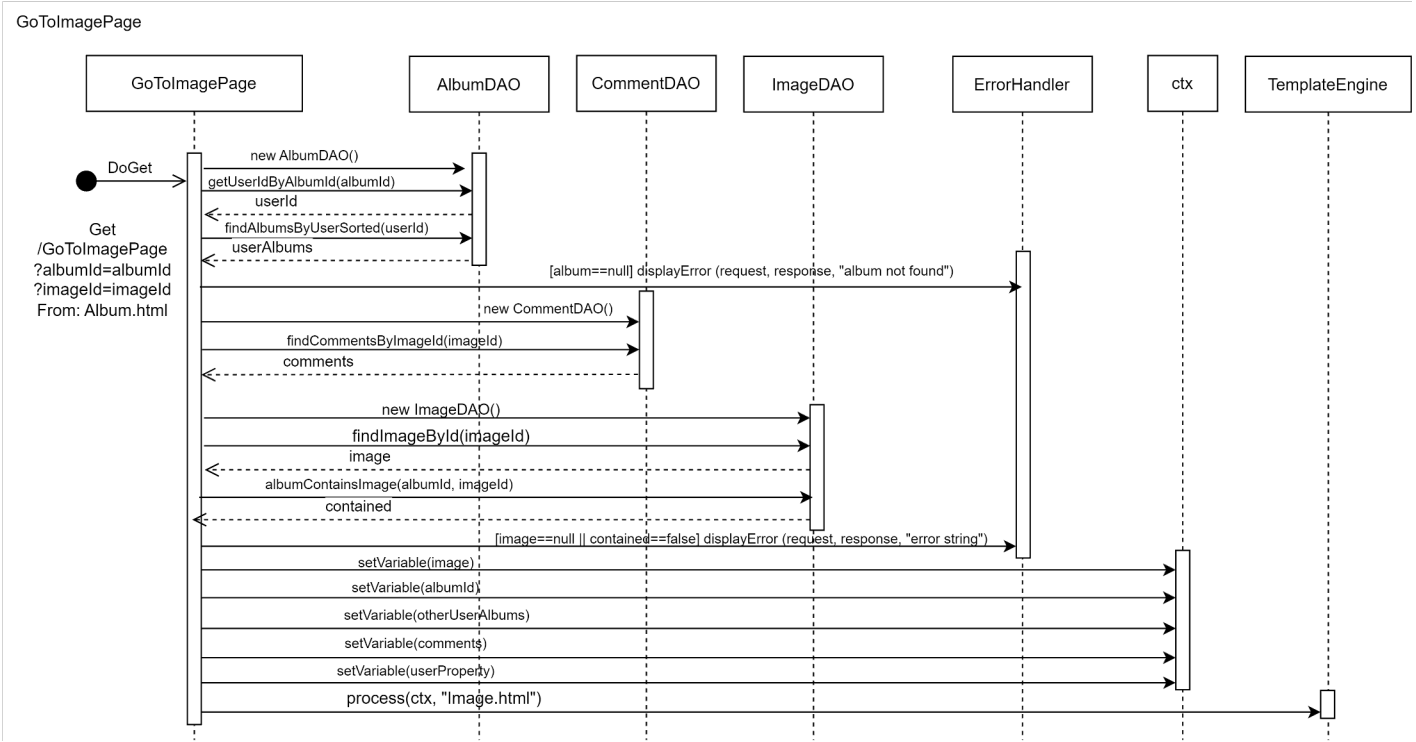
Go to album page



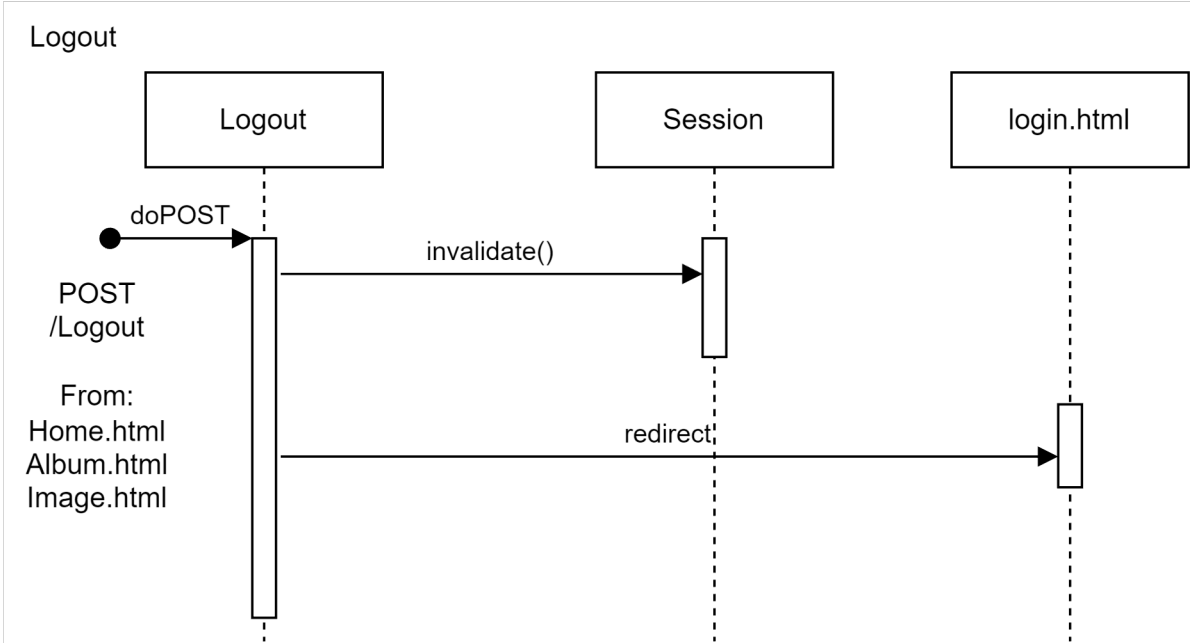
Go to home page



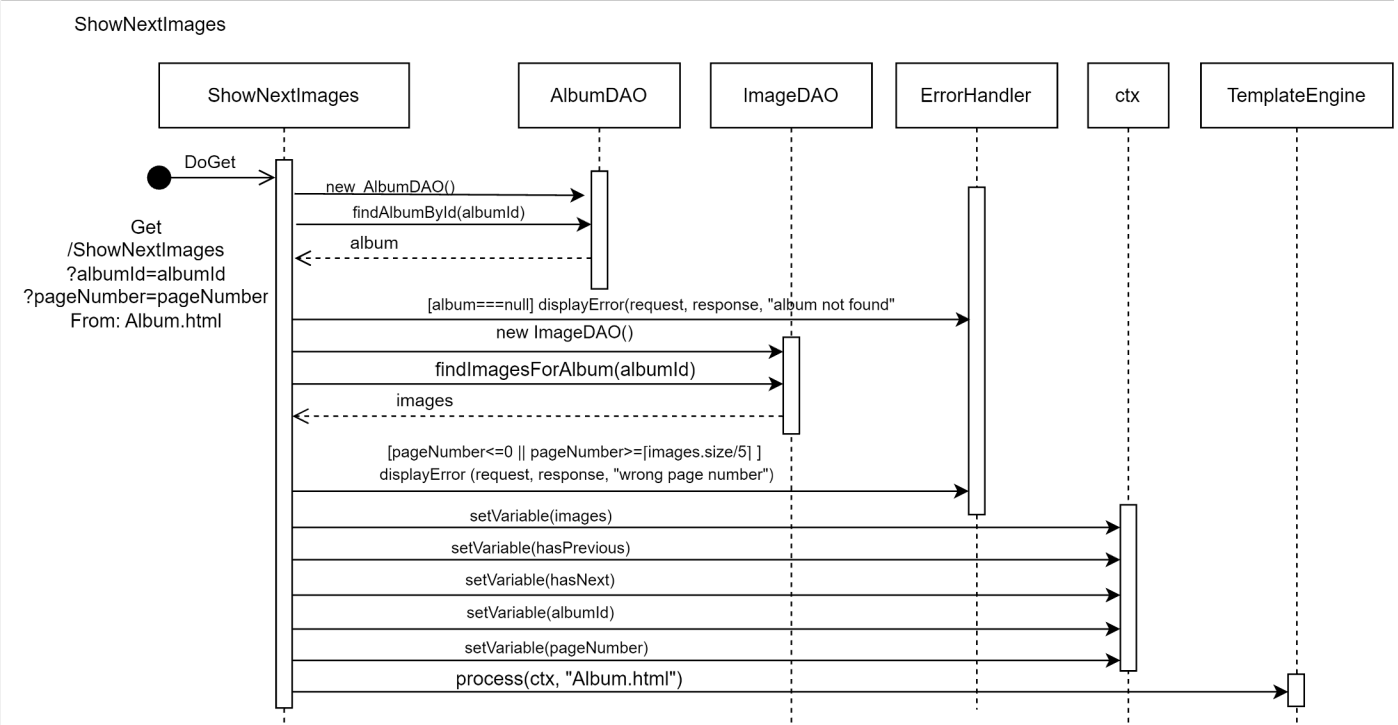
Go to image page



Logout


Logout
Logout
Session
login.html
doPOST
invalidate()
POST
/Logout
From:
Home.html
Album.html
Image.html
redirect

Show next images



Show previous images