

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
Τμήμα Πληροφορικής



Εργασία Μαθήματος «Προγραμματισμός στο διαδίκτυο και στον
παγκόσμιο ιστό»

Τρίτη εργασία	<i>Ολοκλήρωση 3-tier εφαρμογής για τη διαχείριση κρατήσεων εισιτηρίων</i>
---------------	---



ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1	Γενική Περιγραφή της λύσης.....	6
2	Admins.....	7
2.1	Admin.jsp	7
2.2	UserAdmin.jsp	7
2.3	UserContentAdmin.jsp	7
2.4	UserAddContentAdmin.jsp	8
2.5	UserCustomer.jsp.....	8
2.6	UserAddCustomer.jsp	8
2.7	AdminAdminController.java extends HttpServlet (servlet)	9
2.8	AdminContentAdminController.java extends HttpServlet (servlet)	9
2.9	AdminCustomerController.java extends HttpServlet (servlet)	10
3	Content_Admin.....	10
3.1	Logincontentadmin.jsp.....	10
3.2	listMovie.jsp	10
3.3	editMovie.jsp.....	11
3.4	addMovie.jsp.....	11
3.5	listProv.jsp	12
3.6	editProv.jsp	12
3.7	addProv.jsp.....	12
3.8	addProv2.jsp.....	13
3.9	MovieController.java extends HttpServlet (servlet)	13
3.10	ProvolesController.java extends HttpServlet (servlet).....	14
4	Customers.....	14
4.1	Customer.jsp	14
4.2	listMovieCustomer.jsp	14
4.3	reservationCustomer.jsp.....	15
4.4	Reservation.jsp.....	15
4.5	HistoryCustomer.jsp.....	15



4.6	ReservationController.java extends HttpServlet (servlet)	16
4.7	CustomerController.java extends HttpServlet (servlet).....	16
4.8	HistoryController.java extends HttpServlet (servlet).....	16
5	SystemeDao.java	17
5.1	Login	17
5.2	Admin	17
5.3	Content admin.....	18
5.4	Customer	19
6	LoginServlet.java (servlet)	19
7	Class	20
7.1	Κλάση User	20
7.2	Κλάση ContentAdmin extends User.....	20
7.3	Κλάση Customer extends User.....	20
7.4	Κλάση Admin extends User.....	20
7.5	Κλάση Cinemas.....	20
7.6	Κλάση Movies.....	20
7.7	Κλάση Reservation	21
7.8	Κλάση Provoles	21
8	Βιβλιογραφικές Πηγές.....	21



1 Γενική Περιγραφή της λύσης

Σκοπός της εργασίας είναι η δημιουργία μίας διαδικτυακής εφαρμογής (η επικοινωνία γίνεται μέσα από web περιβάλλον) για τη κράτηση θέσεων/εισιτηρίων σε έναν κινηματογράφο. Η εφαρμογή περιλαμβάνει τρεις βασικές κατηγορίες χρηστών: (1) πελάτες, (2) διαχειριστές περιεχομένου και (3) διαχειριστή της εφαρμογής, με διαφορετικά δικαιώματα και λειτουργίες για κάθε κατηγορία χρηστών.

(1) Ένας πελάτης (customer) θα μπορεί να κάνει τις εξής λειτουργίες:

- i. Σύνδεση (login).
- ii. Προβολή προγράμματος προβολών ταινιών για συγκεκριμένο χρονικό διάστημα.
- iii. Διαθεσιμότητα για κάποια προβολή.
- iv. Ηλεκτρονική κράτηση θέσεων για μία προβολή ταινίας κάποια συγκεκριμένη ημέρα, ώρα και αίθουσα.
- v. Προβολή ιστορικού κρατήσεων.
- vi. Αποσύνδεση (logout).

(2) Ένας διαχειριστής περιεχομένου (content admin) θα μπορεί να κάνει τις εξής λειτουργίες:

- i. Σύνδεση (login).
- ii. Προβολή, τροποποίηση και διαγραφή διαθέσιμων ταινιών.
- iii. Εισαγωγή νέας ταινίας.
- iv. Προβολή, τροποποίηση και διαγραφή διαθέσιμων προβολών.
- v. Εισαγωγή νέας προβολής.
- vi. Αποσύνδεση (logout).

(3) Ένας διαχειριστής της εφαρμογής (admin) θα μπορεί να κάνει τις εξής λειτουργίες:

- i. Σύνδεση (login).
- ii. Προβολή όλων των διαχειριστών της εφαρμογής.
- iii. Προβολή και διαγραφή όλων των διαχειριστών περιεχομένου.
- iv. Εισαγωγή νέου διαχειριστή περιεχομένου.
- v. Προβολή και διαγραφή όλων των πελατών.
- vi. Εισαγωγή νέου πελάτη.
- vii. Αποσύνδεση (logout).

Η εφαρμογή έχει λειτουργικότητα 3-tier, χρήση server-side τεχνολογιών (servlets και jsp) και επικοινωνία με βάση δεδομένων.



2 Admins

Ακολουθεί η αναλυτική περιγραφή των αρχείων του προγράμματος που αναφέρονται στις λειτουργίες του διαχειριστεί εφαρμογής (admin).

2.1 Admin.jsp

- Κάνει forward στο AdminAdminController.java ώστε να εμφανιστεί το UserAdmin.jsp

2.2 UserAdmin.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head> συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου
- Μέσα στο <body> :
Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του χρήστη.
Υπάρχει ακόμα ένα table στο οποίο εμφανίζονται όλοι οι διαχειριστές εφαρμογής μέσω του AdminAdminController με action = listAdmin.
Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

2.3 UserContentAdmin.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head> συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου
- Μέσα στο <body> :
Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του χρήστη.
Υπάρχει ακόμα ένα table στο οποίο εμφανίζονται όλοι οι διαχειριστές εφαρμογής μέσω του AdminContentAdminController με action = listContentAdmin και υπάρχει η δυνατότητα διαγραφής κάποιου χρήστη μέσω του AdminContentAdminController με action = delete.
Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.
Υπάρχει ένα script για την εμφάνιση κατάλληλων μηνυμάτων κατά την διαγραφή διαχειριστή περιεχομένου.



2.4 UserAddContentAdmin.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head>:
Συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου
Υπάρχει ένα script για την εμφάνιση κατάλληλων μηνυμάτων κατά την προσθήκη διαχειριστή περιεχομένου.
- Μέσα στο <body> :
Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του χρήστη.
Υπάρχει μια φόρμα συμπλήρωσης με υποχρεωτικά πεδία για την προσθήκη διαχειριστή περιεχομένου.
Η φόρμα προωθεί τα πεδία στο AdminContentAdminController με action = insert.
Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

2.5 UserCustomer.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head> συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου
- Μέσα στο <body> :
Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του χρήστη.
Υπάρχει ακόμα ένα table στο οποίο εμφανίζονται όλοι οι διαχειριστές εφαρμογής μέσω του AdminCustomerController με action = listCustomer και υπάρχει η δυνατότητα διαγραφής κάποιου χρήστη μέσω του AdminCustomerController με action = delete.
Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.
Υπάρχει ένα script για την εμφάνιση κατάλληλων μηνυμάτων κατά την διαγραφή πελάτη.

2.6 UserAddCustomer.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head>:
Συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου
Υπάρχει ένα script για την εμφάνιση κατάλληλων μηνυμάτων κατά την προσθήκη πελάτη.



- Μέσα στο `<body>` :
Υπάρχει ένα `navbar` το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του χρήστη.
Υπάρχει μια φόρμα συμπλήρωσης με υποχρεωτικά πεδία για την προσθήκη πελάτη.
Η φόρμα προωθεί τα πεδία στο `AdminController` με `action = insert`.
Τέλος συνδέεται ένα αρχείο `.js` που αφορά την λειτουργία του `navbar`.

2.7 AdminAdminController.java extends HttpServlet (servlet)

- Έχουν γίνει τα κατάλληλα `import` στην αρχή.
- Δημιουργείται νέα `SystemeDao` μέσω της συνάρτησης `public AdminAdminController()`.
- Η μέθοδος `doGet`:
Δέχεται το `action` και μέσω της κλάσης `dao` ανακτά τα απαραίτητα δεδομένα.
Ανάλογα με το `action` που δέχεται κάνει `forward` στο αντίστοιχο `.jsp`
- Η μέθοδος `doPost`:
Δημιουργείται αντικείμενο τύπου `Admin` για την αποθήκευση των δεδομένων διαχειριστή εφαρμογής.
Λήψη των δεδομένων του διαχειριστή εφαρμογής από τα πεδία της φόρμας του αιτήματος και αποθήκευση τους στα αντίστοιχα πεδία του αντικειμένου `Admin`.
Μέσω της `dao` ανακτά τα απαραίτητα δεδομένα και κάνει `forward` στο αντίστοιχο `.jsp`.

2.8 AdminContentAdminController.java extends HttpServlet (servlet)

- Έχουν γίνει τα κατάλληλα `import` στην αρχή.
- Δημιουργείται νέα `SystemeDao` μέσω της συνάρτησης `public AdminContentAdminController()`.
- Η μέθοδος `doGet`:
Δέχεται το `action` και μέσω της κλάσης `dao` ανακτά τα απαραίτητα δεδομένα.
Ανάλογα με το `action` που δέχεται κάνει `forward` στο αντίστοιχο `.jsp`
- Η μέθοδος `doPost`:
Δημιουργείται αντικείμενο τύπου `ContentAdmin` για την αποθήκευση των δεδομένων διαχειριστή περιεχομένου.
Λήψη των δεδομένων του διαχειριστή περιεχομένου από τα πεδία της φόρμας του αιτήματος και αποθήκευση τους στα αντίστοιχα πεδία του αντικειμένου `ContentAdmin`.
Δέχεται το `action` και μέσω της κλάσης `dao` ανακτά τα απαραίτητα δεδομένα.



Μέσω της dao ανακτά τα απαραίτητα δεδομένα και κάνει forward στο αντίστοιχο .jsp.

2.9 AdminCustomerController.java extends HttpServlet (servlet)

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Δημιουργείται νέα SystemDao μέσω της συνάρτησης public AdminCustomerController()
- Η μέθοδος doGet:
Δέχεται το action και μέσω της κλάσης dao ανακτά τα απαραίτητα δεδομένα.
Ανάλογα με το action που δέχεται κάνει forward στο αντίστοιχο .jsp
- Η μέθοδος doPost:
Δημιουργείται αντικείμενο τύπου Customer για την αποθήκευση των δεδομένων πελάτη.
Λήψη των δεδομένων του πελάτη από τα πεδία της φόρμας του αιτήματος και αποθήκευση τους στα αντίστοιχα πεδία του αντικειμένου Customer.
Δέχεται το action και μέσω της κλάσης dao ανακτά τα απαραίτητα δεδομένα.
Μέσω της dao ανακτά τα απαραίτητα δεδομένα και κάνει forward στο αντίστοιχο .jsp.

3 Content Admins

Ακολουθεί η αναλυτική περιγραφή των αρχείων του προγράμματος που αναφέρονται στις λειτουργίες του διαχειριστεί εφαρμογής (admin).

3.1 Logincontentadmin.jsp

- Κάνει forward στο MovieController.java ώστε να εμφανιστεί το listMovie.jsp

3.2 listMovie.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head> συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου
- Μέσα στο <body> :



Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του content admin.

Υπάρχει ακόμα ένα table στο οποίο εμφανίζονται όλες οι ταινίες μέσω του MovieController με action = listMovie, υπάρχει η δυνατότητα ενημέρωσης κάποιας ταινίας μέσω του MovieController με action = update το οποίο σε προωθεί στο editMovie.jsp και υπάρχει η δυνατότητα διαγραφής κάποιας ταινίας μέσω του MovieController με action = delete.

Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

Υπάρχει ένα script για την εμφάνιση κατάλληλων μηνυμάτων κατά την διαγραφή ταινιών.

3.3 editMovie.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.

- Μέσα στο <head>:

Συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου

Υπάρχει ένα script για την εμφάνιση κατάλληλων μηνυμάτων κατά την ενημέρωση ταινιών.

- Μέσα στο <body> :

Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του content admin.

Υπάρχει μια φόρμα συμπλήρωσης πεδίων για την προσθήκη ταινιών.

Η φόρμα προωθεί τα πεδία στο MovieController με action = edit.

Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

3.4 addMovie.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.

- Μέσα στο <head>:

Συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου

Υπάρχει ένα script για την εμφάνιση κατάλληλων μηνυμάτων κατά την προσθήκη ταινιών.

- Μέσα στο <body> :

Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του content admin.

Υπάρχει μια φόρμα συμπλήρωσης πεδίων για την προσθήκη ταινιών.

Η φόρμα προωθεί τα πεδία στο MovieController με action = insert.



Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

3.5 listProv.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head> συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου
- Μέσα στο <body> :

Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του content admin.

Υπάρχει ακόμα ένα table στο οποίο εμφανίζονται όλες οι προβολές μέσω του ProvolesController με action = listProv, υπάρχει η δυνατότητα ενημέρωσης κάποιας προβολής μέσω του ProvolesController με action = update το οποίο σε προωθεί στο editProv.jsp και υπάρχει η δυνατότητα διαγραφής καποιας προβολής μέσω του ProvolesController με action = delete.

Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

Υπάρχει ένα script για την εμφάνιση κατάλληλων μηνυμάτων κατά την διαγραφή προβολών.

3.6 editProv.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head>:

Συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου

Υπάρχει ένα script για την εμφάνιση κατάλληλων μηνυμάτων κατά την ενημέρωση προβολών.

- Μέσα στο <body> :

Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του content admin.

Υπάρχει μια φόρμα συμπλήρωσης πεδίων για την προσθήκη προβολών.

Η φόρμα προωθεί τα πεδία στο ProvolesController με action = edit.

Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

3.7 addProv.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head> συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου



- Μέσα στο `<body>` :

Υπάρχει ένα `navbar` το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του `content admin`.

Υπάρχει ακόμα ένα `table` στο οποίο εμφανίζονται όλες οι ταινίες μέσω του `ProvolesController` με `action = movies`, υπάρχει η δυνατότητα επιλογής συγκεκριμένης ταινίας για την εμφάνιση των προβολών της μέσω του `ProvolesController` με `action = add` το οποίο σε προωθεί στο `editProv2.jsp`.

Τέλος συνδέεται ένα αρχείο `.js` που αφορά την λειτουργία του `navbar`.

3.8 `addProv2.jsp`

- Έχουν γίνει τα κατάλληλα `import` στην αρχή.

- Μέσα στο `<head>`:

Συνδέονται δυο αρχεία `.css` τα οποία αφορούν την μορφοποίηση του αρχείου

Υπάρχει ένα `script` για την εμφάνιση κατάλληλων μηνυμάτων κατά την προσθήκη προβολών.

- Μέσα στο `<body>` :

Υπάρχει ένα `navbar` το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του `content admin`.

Υπάρχει μια φόρμα συμπλήρωσης πεδίων για την προσθήκη προβολών.

Η φόρμα προωθεί τα πεδία στο `ProvolesController` με `action = insert`.

Τέλος συνδέεται ένα αρχείο `.js` που αφορά την λειτουργία του `navbar`.

3.9 `MovieController.java extends HttpServlet (servlet)`

- Έχουν γίνει τα κατάλληλα `import` στην αρχή.
- Δημιουργείται νέα `SystemeDao` μέσω της συνάρτησης `public MovieController()`.
- Η μέθοδος `doGet`:

Δέχεται το `action` και μέσω της κλάσης `dao` ανακτά τα απαραίτητα δεδομένα.

Ανάλογα με το `action` που δέχεται κάνει `forward` στο αντίστοιχο `.jsp`

- Η μέθοδος `doPost`:

Δημιουργείται αντικείμενο τύπου `Movies` για την αποθήκευση των δεδομένων ταινιών.

Λήψη των δεδομένων των ταινιών από τα πεδία της φόρμας του αιτήματος και αποθήκευση τους στα αντίστοιχα πεδία του αντικειμένου `Movies`.

Δέχεται το `action` και μέσω της κλάσης `dao` ανακτά τα απαραίτητα δεδομένα.

Μέσω της `dao` ανακτά τα απαραίτητα δεδομένα και κάνει `forward` στο αντίστοιχο `.jsp`.



3.10 ProvolesController.java extends HttpServlet (servlet)

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Δημιουργείται νέα SystemeDao μέσω της συνάρτησης public ProvolesController().
- Η μέθοδος doGet:
Δέχεται το action και μέσω της κλάσης dao ανακτά τα απαραίτητα δεδομένα.
Ανάλογα με το action που δέχεται κάνει forward στο αντίστοιχο .jsp
- Η μέθοδος doPost:
Δημιουργείται αντικείμενο τύπου Provoles για την αποθήκευση των δεδομένων προβολών.
Λήψη των δεδομένων των προβολών από τα πεδία της φόρμας του αιτήματος και αποθήκευση τους στα αντίστοιχα πεδία του αντικειμένου Provoles.
Δέχεται το action και μέσω της κλάσης dao ανακτά τα απαραίτητα δεδομένα.
Μέσω της dao ανακτά τα απαραίτητα δεδομένα και κάνει forward στο αντίστοιχο .jsp.

4 Customers

Ακολουθεί η αναλυτική περιγραφή των αρχείων του προγράμματος που αναφέρονται στις λειτουργίες του πελάτη (customer).

4.1 Customer.jsp

- Κάνει forward στο CustomerController.java ώστε να εμφανιστεί το listMovieCustomer.jsp

4.2 listMovieCustomer.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head> συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου listMovieCustomer.jsp
- Μέσα στο <body> :
Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του χρήστη.
Υπάρχει ακόμα ένα table στο οποίο εμφανίζονται όλες οι ταινίες μέσω του CustomerController με action = listMovieCustomer και υπάρχει η δυνατότητα προβολής διαθεσιμότητας προβολών μέσω του CustomerController με action = reservation.



Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

4.3 reservationCustomer.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head> συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου reservationCustomer.jsp
- Μέσα στο <body> :

Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του χρήστη.

Υπάρχει ακόμα ένα table στο οποίο εμφανίζονται όλες οι ταινίες μέσω του CustomerController με action = reservation και υπάρχει η δυνατότητα κράτησης προβολής για μια ταινία μέσω του ReservationController με action = reservation2.

Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

4.4 Reservation.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head>:

Συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου Reservation.jsp
- Μέσα στο <body> :

Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του χρήστη.

Υπάρχει μια φόρμα συμπλήρωσης πεδίων για την κράτηση θέσεων προβολής για μια ταινία.

Η φόρμα προωθεί τα πεδία στο ReservationController με action = reservation2 .

Υπάρχει ένα script για τον έλεγχο διαθέσιμων θέσεων και για την εμφάνιση κατάλληλων μηνυμάτων κατά την κράτηση.

Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

4.5 HistoryCustomer.jsp

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Μέσα στο <head> συνδέονται δυο αρχεία .css τα οποία αφορούν την μορφοποίηση του αρχείου HistoryCustomer.jsp
- Μέσα στο <body> :



Υπάρχει ένα navbar το οποίο είναι υπεύθυνο για την πλοήγηση στις διάφορες λειτουργίες του χρήστη.

Υπάρχει ακόμα ένα table στο οποίο εμφανίζεται το ιστορικό κρατήσεων μέσω του HistoryController με action = History.

Τέλος συνδέεται ένα αρχείο .js που αφορά την λειτουργία του navbar.

4.6 ReservationController.java extends HttpServlet (servlet)

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Δημιουργείται νέα SystemDao μέσω της συνάρτησης public ReservationController ().
- Η μέθοδος doGet:
Δέχεται το action και μέσω της κλάσης dao ανακτά τα απαραίτητα δεδομένα.
Ανάλογα με το action που δέχεται κάνει forward στο αντίστοιχο .jsp
- Η μέθοδος doPost:
Δημιουργείται αντικείμενο τύπου Reservation για την αποθήκευση των δεδομένων κρατήσεων.
Λήψη των δεδομένων των κρατήσεων από τα πεδία της φόρμας του αιτήματος και αποθήκευση τους στα αντίστοιχα πεδία του αντικειμένου Reservation.
Δέχεται το action και μέσω της κλάσης dao ανακτά τα απαραίτητα δεδομένα.
Μέσω της dao ανακτά τα απαραίτητα δεδομένα και κάνει forward στο αντίστοιχο .jsp.

4.7 CustomerController.java extends HttpServlet (servlet)

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Δημιουργείται νέα SystemDao μέσω της συνάρτησης public CustomerController ().
- Η μέθοδος doGet:
Δέχεται το action και μέσω της κλάσης dao ανακτά τα απαραίτητα δεδομένα.
Ανάλογα με το action που δέχεται κάνει forward στο αντίστοιχο .jsp
- Η μέθοδος doPost:
Μέσω της dao ανακτά τα απαραίτητα δεδομένα και κάνει forward στο αντίστοιχο .jsp.

4.8 HistoryController.java extends HttpServlet (servlet)

- Έχουν γίνει τα κατάλληλα import στην αρχή.
- Δημιουργείται νέα SystemDao μέσω της συνάρτησης public HistoryController ().
- Η μέθοδος doGet:



Δέχεται το action και μέσω της κλάσης dao ανακτά τα απαραίτητα δεδομένα.

Ανάλογα με το action που δέχεται κάνει forward στο αντίστοιχο .jsp

- Η μέθοδος doPost:

Δημιουργείται αντικείμενο τύπου Reservation για την αποθήκευση των δεδομένων κρατήσεων.

Λήψη των δεδομένων των κρατήσεων από τα πεδία της φόρμας του αιτήματος και αποθήκευση τους στα αντίστοιχα πεδία του αντικειμένου Reservation.

Δέχεται το action και μέσω της κλάσης dao ανακτά τα απαραίτητα δεδομένα.

Μέσω της dao ανακτά τα απαραίτητα δεδομένα και κάνει forward στο αντίστοιχο .jsp.

5 SystemeDao.java

Ακολουθεί η αναλυτική περιγραφή των αρχείων του προγράμματος που αναφέρονται στο αρχείο SystemeDao.java

Το αρχείο dao χωρίζεται σε 4 μέρη :

5.1 Login

1. loginUsernameCheck(): Μέσω της βάσης ελέγχει εάν υπάρχει το όνομα χρήστη και επιστρέφει μια boolean τιμή.
2. passwordCheck(): Μέσω της βάσης επαληθεύει τον κωδικό για τον αντίστοιχο χρήστη και επιστρέφει boolean τιμή.
3. getRole(): Μέσω της βάσης βρίσκει και επιστρέφει την ιδιότητα του χρήστη που συνδέθηκε.
4. getAdminId(): Μέσω της βάσης βρίσκει και επιστρέφει το ID του admin.
5. getContentAdminId(): Μέσω της βάσης βρίσκει και επιστρέφει το ID του content admin.
6. getCustomerId(): Μέσω της βάσης βρίσκει και επιστρέφει το ID του customer.Admin

5.2 Admin

1. deleteUser(): Διαγραφή από την βάση του πίνακα user που έχει το συγκεκριμένο username.
2. getAllAdmins(): Δημιουργεί αντικείμενα για κάθε admin στα οποία αποθηκεύει τα πεδία του πίνακα admins και user (join) και τα καταχωρεί σε μια λίστα η οποία επιστρέφεται.



3. `getAllContentAdmins()`: Δημιουργεί αντικείμενα για κάθε `content_admin` στα οποία αποθηκεύει τα πεδία του πίνακα `content_admins` και `user` (join) και τα καταχωρεί σε μια λίστα η οποία επιστρέφεται.
4. `deleteContentAdmin()`: Διαγράφη από την βάση του πίνακα `content_admins` που έχει το συγκεκριμένο ID.
5. `getLatestContentAdminId()`: Παίρνει το μεγαλύτερο ID από τον πίνακα `content_admin` και το επιστρέφει.
6. `addUserContentAdmin()`: Παίρνει τα περιεχόμενα από το αντικείμενο `content_admin`, τα οποία εκχωρούνται στον πίνακα `user` με `role = 'content_admin'`.
7. `addContentAdmin()`: Παίρνει τα περιεχόμενα από το αντικείμενο `content_admin`, το οποίο εκχωρείτε.
8. `getAllCustomers()`: Δημιουργεί αντικείμενα για κάθε `Customer` στα οποία αποθηκεύει τα πεδία του πίνακα `customers` και `user` (join) και τα καταχωρεί σε μια λίστα η οποία επιστρέφεται.
9. `deleteCustomer()`: Διαγράφη από την βάση του πίνακα `customers` που έχει το συγκεκριμένο ID.
10. `getLatestCustomerId()`: Παίρνει το μεγαλύτερο ID από τον πίνακα `customers` και το επιστρέφει.
11. `addUserCustomer()`: Παίρνει τα περιεχόμενα από το αντικείμενο `Customer`, τα οποία εκχωρούνται στον πίνακα `user` με `role = 'customer'`.
12. `addCustomer()`: Παίρνει τα περιεχόμενα από το αντικείμενο `Customer`, το οποίο εκχωρείτε.

5.3 Content admin

13. `getAllMovies()`: Δημιουργεί αντικείμενα για κάθε ταινία στα οποία αποθηκεύει τα πεδία του πίνακα `movies` και τα καταχωρεί σε μια λίστα η οποία επιστρέφεται.
14. `getMovieById()`: Δημιουργεί ένα αντικείμενο `movies` με βάση το ID στο οποίο αποθηκεύει τα πεδία του πίνακα `movies` και επιστρέφει το αντικείμενο αυτό.
15. `updateMovie()`: Παίρνει τα περιεχόμενα από το αντικείμενο `movies`, το οποίο ενημερώθηκε και τα τοποθετεί στην βάση με βάση το ID.
16. `deleteMovies()`: Διαγράφη από την βάση την ταινία που έχει το συγκεκριμένο ID.
17. `getLatestMovieById()`: Παίρνει το μεγαλύτερο ID από τον πίνακα `movies` και το επιστρέφει.
18. `addMovies()`: Παίρνει τα περιεχόμενα από το αντικείμενο `movies`, το οποίο εκχωρείτε.
19. `getAllProvoles()`: Δημιουργεί αντικείμενα για κάθε προβολή στα οποία αποθηκεύει τα πεδία του πίνακα `provoles` και τα καταχωρεί σε μια λίστα η οποία επιστρέφεται.



20. `getProvById()`: Δημιουργεί ένα αντικείμενο `Provoles` με βάση το ID στο οποίο αποθηκεύει τα πεδία ID και NAME του πίνακα `movies` και επιστρέφει το αντικείμενο αυτό.
21. `getProvByIdEdit()`: Δημιουργεί ένα αντικείμενο `Provoles` με βάση το ID στο οποίο αποθηκεύει τα πεδία του πίνακα `pron` και επιστρέφει το αντικείμενο αυτό.
22. `updateProvoles()`: Παίρνει τα περιεχόμενα από το αντικείμενο `provoles`, το οποίο ενημερώθηκε και τα τοποθετεί στην βάση με βάση το ID.
23. `deleteProvoles()`: Διαγράφει από την βάση την προβολή που έχει το συγκεκριμένο ID.
24. `getLatestProvId()`: Παίρνει το μεγαλύτερο ID από τον πίνακα `provoles` και το επιστρέφει.
25. `addProvoles()`: Παίρνει τα περιεχόμενα από το αντικείμενο `provoles`, το οποίο εκχωρείτε.

5.4 Customer

1. `getProvolesById()`: Δημιουργεί αντικείμενα για κάθε προβολή στα οποία αποθηκεύει τα στοιχεία του πίνακα `pron`. Τα αντικείμενα αυτά είναι οι προβολές της ταινίας για το συγκεκριμένο ID που έχει δοθεί στην συνάρτηση.
2. `getReservById()`: Δημιουργεί ένα αντικείμενο `Reservation` με βάση το `MOVIES_ID` και το `TIME` στο οποίο αποθηκεύει τα πεδία του πίνακα `pron` και επιστρέφει το αντικείμενο αυτό.
3. `getSeats()`: Παίρνει από την βάση τον συνολικό αριθμό των καθισμάτων που έχουν κρατηθεί για την συγκεκριμένη προβολή ταινίας και χρονική στιγμή για το συγκεκριμένο `PROVOLES_MOVIES_ID` και το `PR_TIME` και το επιστρέφει.
4. `getCinemaSeats()`: Παίρνει από την βάση τον συνολικό αριθμό των καθισμάτων που υπάρχουν σε αυτό το σινεμά (`CINEMAS_ID`) και το επιστρέφει.
5. `addReservation()`: Παίρνει τα περιεχόμενα από το αντικείμενο `Reservation`, το οποίο εκχωρείτε.
6. `getCustomerHistory()`: Δημιουργεί αντικείμενα για κάθε κράτηση στα οποία αποθηκεύει τα πεδία του πίνακα `reser` για το συγκεκριμένο ID και τα καταχωρεί σε μια λίστα η οποία επιστρέφεται.

6 LoginServlet.java (servlet)

- Έχουν γίνει τα κατάλληλα `import` στην αρχή.
- Δημιουργείται νέα `SystemeDao` μέσω της συνάρτησης `public LoginServlet ()`.
- Η μέθοδος `doPost`:



Ελέγχει εάν το username και το password που έδωσε ο χρήστης υπάρχουν στην βάση. Αν το username και το password είναι σωστά, τότε δημιουργεί ένα νέο session και αποθηκεύει το username και το role του στο session. Ανάλογα με το role του χρήστη, καθορίζει την σελίδα που θα εμφανιστεί μετά τη σύνδεση. Εάν το username ή το password είναι λάθος δεν δημιουργείται νέο session και ο χρήστης καλείται να ξανά συμπληρώσει την φόρμα.

7 Class

7.1 Κλάση User

- Έχουν δημιουργηθεί τα attributes: username, email, password, create_time, salt, role.
- Έχουν δημιουργηθεί μέθοδοι setters getters για όλα τα attributes της κλάσης.

7.2 Κλάση ContentAdmin extends User

- Έχουν δημιουργηθεί τα attributes: ID, NAME, user_username=getUsername().
- Έχουν δημιουργηθεί μέθοδοι setters getters για όλα τα attributes της κλάσης

7.3 Κλάση Customer extends User

- Έχουν δημιουργηθεί τα attributes: ID, NAME, user_username=getUsername().
- Έχουν δημιουργηθεί μέθοδοι setters getters για όλα τα attributes της κλάσης

7.4 Κλάση Admin extends User

- Έχουν δημιουργηθεί τα attributes: ID, NAME, user_username=getUsername().
- Έχουν δημιουργηθεί μέθοδοι setters getters για όλα τα attributes της κλάσης

7.5 Κλάση Cinemas

- Έχουν δημιουργηθεί τα attributes: ID, NAME, SEATS, D3.
- Έχουν δημιουργηθεί μέθοδοι setters getters για όλα τα attributes της κλάσης.

7.6 Κλάση Movies

- Έχουν δημιουργηθεί τα attributes: ID, NAME, CONTENT, LENGTH, TYPE, SUMMARY, DIRECTOR, CONTENT_ADMIN_ID.



- Έχουν δημιουργηθεί μέθοδοι setters getters για όλα τα attributes της κλάσης.

7.7 Κλάση Reservation

- Έχουν δημιουργηθεί τα attributes: PROVOLES_MOVIES_ID, PROVOLES_MOVIES_NAME, PROVOLES_CINEMAS_ID, CUSTOMERS_ID, NUMBER_OF_SEATS, PR_TIME.
- Έχουν δημιουργηθεί μέθοδοι setters getters για όλα τα attributes της κλάσης.

7.8 Κλάση Provoles

- Έχουν δημιουργηθεί τα attributes: MOVIES_ID, MOVIES_NAME, CINEMA_ID, CONTENT_ADMIN_ID, TIME, ID.
- Έχουν δημιουργηθεί μέθοδοι setters getters για όλα τα attributes της κλάσης.

8 Βιβλιογραφικές Πηγές

1. *Menezes, Vandstone. Handbook of Applied Cryptography. New York : CRC PRes, 1996.*
2. [09 mvc-example](#)
3. [09 mvc-example-session-db](#)
4. [09 MVC-Bootstrap-example](#)
5. [06b-ExamplePost](#)
6. [06d 3-tier example](#)
7. <https://codepen.io/>
8. <https://www.w3schools.com/>