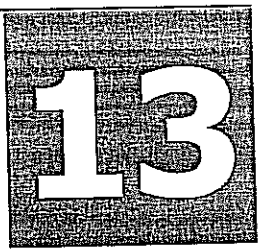
ΣΥΜΠΕΡΙΛΗΨΗ ΑΡΧΕΙΩΝ ΚΑΙ ΜΙΚΡΟΕΦΑΡΜΟΓΩΝ ΣΕ ΣΕΛΙΔΕΣ

Θέματα σε αυτό το Κεφάλαιο

- Χρήση της `jsp:include` για τη συμπερίληψη σελίδων κατά το χρόνο αίτησης
- Χρήση της `<%@ include ...%>` (η οδηγία `include`) για τη συμπερίληψη αρχείων κατά το χρόνο μετάφρασης
- Γιατί η `jsp:include` είναι συνήθως καλύτερη από την οδηγία `include`
- Χρήση της `jsp:plugin` για τη συμπερίληψη μικροεφαρμογών για το Java Plug-in

Κεφάλαιο 

Η JSP έχει τρεις βασικές δυνατότητες για τη συμπερίληψη (include) εξωτερικών τμημάτων σε ένα έγγραφο JSP:

- **Την ενέργεια `jsp:include`.** Η ενέργεια (action) `jsp:include` σας επιτρέπει να συμπεριλάβετε την έξοδο μιας σελίδας κατά το χρόνο αίτησης. Το βασικό της πλεονέκτημα είναι ότι δεν χρειάζεται να τροποποιήσετε την κύρια σελίδα όταν αλλάζουν οι συμπεριλαμβανόμενες σελίδες. Το κύριο μειονέκτημα είναι ότι, επειδή συμπεριλαμβάνεται η *έξοδος* της δευτερεύουσας σελίδας και όχι ο πραγματικός της *κώδικας* (όπως στην περίπτωση της οδηγίας `include`), οι συμπεριλαμβανόμενες σελίδες δεν μπορούν να χρησιμοποιήσουν εντολές JSP που να επηρεάζουν συνολικά την κύρια σελίδα. Γενικά τα πλεονεκτήματα ξεπερνούν κατά πολύ τα μειονεκτήματα, και σχεδόν σίγουρα θα χρησιμοποιείτε αυτό το μηχανισμό συμπερίληψης πολύ περισσότερο από τους υπόλοιπους. Η χρήση της ενέργειας `jsp:include` περιγράφεται στην Ενότητα 13.1.
- **Η οδηγία `include`.** Αυτή η εντολή σας επιτρέπει να εισάγετε κώδικα JSP στην κύρια σελίδα πριν αυτή μεταφραστεί σε μικροϋπηρεσία. Το βασικό της πλεονέκτημα είναι η παρεχόμενη ισχύς: ο κώδικας που συμπεριλαμβάνεται μπορεί να περιέχει εντολές JSP, όπως ορισμούς πεδίων και ρυθμίσεις τύπου περιεχομένου, που επηρεάζουν συνολικά την κύρια σελίδα. Το βασικό της μειονέκτημα είναι η δυσκολία στη συντήρηση: θα πρέπει να ενημερώνετε την κύρια σελίδα κάθε φορά που αλλάζουν οι συμπεριλαμβανόμενες σελίδες. Η χρήση της οδηγίας `include` περιγράφεται στην Ενότητα 13.2.

- Η ενέργεια `jsp:plugin`. Αν και το βιβλίο ασχολείται κυρίως με τη Java όταν εκτελείται στην πλευρά του διακομιστή, η Java στην πλευρά του πελάτη, με τη μορφή των ενσωματωμένων μικροεφαρμογών (applets), συνεχίζει να παίζει κάποιο ρόλο, ιδιαίτερα στα εταιρικά ενδοδίκτυα. Το στοιχείο `jsp:plugin` χρησιμοποιείται για την εισαγωγή μικροεφαρμογών, που χρησιμοποιούν το Java Plug-in, σε σελίδες JSP. Το βασικό της πλεονέκτημα είναι ότι γλυτώνετε από τη συγγραφή μεγάλων, επίπονων, και επιρρεπών σε σφάλματα ετικετών OBJECT και EMBED στον κώδικα HTML. Το κύριο μειονέκτημα είναι ότι ισχύει για μικροεφαρμογές, και οι μικροεφαρμογές δεν χρησιμοποιούνται και τόσο συχνά. Η χρήση του στοιχείου `jsp:plugin` περιγράφεται στην Ενότητα 13.4.

13.1 Συμπερίληψη σελίδων κατά το χρόνο αίτησης: Η ενέργεια jsp:include

Υποθέστε ότι έχετε μια σειρά από σελίδες όπου όλες οι σελίδες έχουν την ίδια γραμμή πλοήγησης, τις ίδιες πληροφορίες επικοινωνίας, ή την ίδια υποσημείωση. Τι μπορείτε να κάνετε; Μια συνηθισμένη "λύση" είναι η αποκοπή και επικόλληση των ίδιων τμημάτων κώδικα HTML σε όλες τις σελίδες. Πρόκειται για κακή ιδέα, επειδή όταν αλλάξετε το κοινό τμήμα θα πρέπει να τροποποιήσετε κάθε σελίδα που το χρησιμοποιεί. Μια άλλη συνηθισμένη λύση είναι η χρήση κάποιου είδους μηχανισμού συμπερίληψης στην πλευρά του διακομιστή, όπου το κοινό τμήμα θα εισάγεται κατά την αίτηση της σελίδας. Η γενική προσέγγιση είναι καλή, αλλά οι τυπικοί μηχανισμοί εξαρτώνται από το διακομιστή. Καλωσορίσατε στη `jsp:include`, ένα φορητό μηχανισμό που σας επιτρέπει να εισάγετε στην έξοδο JSP οποιοδήποτε από τα ακόλουθα στοιχεία:

- Το περιεχόμενο μιας σελίδας HTML
- Το περιεχόμενο ενός εγγράφου απλού κειμένου
- Την έξοδο μιας σελίδας JSP
- Την έξοδο μιας μικροϋπηρεσίας

Η ενέργεια `jsp:include` ενσωματώνει την έξοδο μιας δευτερεύουσας σελίδας κατά το χρόνο που γίνεται η αίτηση της κύριας σελίδας. Αν και η *έξοδος* των συμπεριλαμβανόμενων σελίδων δεν μπορεί να περιέχει κώδικα JSP, οι σελίδες μπορούν να είναι το αποτέλεσμα πόρων που χρησιμοποιούν μικροϋπηρεσίες ή σελίδες JSP για τη *δημιουργία* της εξόδου. Με άλλα λόγια, το URL που αναφέρεται στους συμπεριλαμβανόμενους πόρους ερμηνεύεται κανονικά από το διακομιστή, και έτσι μπορεί να είναι μικροϋπηρεσία ή σελίδα JSP. Ο διακομιστής εκτελεί τη συμπεριλαμβανόμενη σελίδα με το συνηθισμένο τρόπο και τοποθετεί την έξοδο στην κύρια σελίδα. Αυτή ακριβώς είναι η συμπεριφορά της μεθόδου `include` της κλάσης `RequestDispatcher` — δείτε το Κεφάλαιο 15, "Ολοκλήρωση μικροϋπηρεσιών και JSP: Η αρχιτεκτονική Model View Controller (MVC)" — και είναι αυτό που χρησιμοποιούν οι μικροϋπηρεσίες όταν θέλουν να πραγματοποιήσουν αυτόν τον τύπο συμπερίληψης.

Η ιδιότητα page: Καθορισμός της συμπεριλαμβανόμενης σελίδας

Με την ιδιότητα page καθορίζετε τη σελίδα που θα συμπεριληφθεί, με τον τρόπο που φαίνεται στη συνέχεια. Αυτή η ιδιότητα είναι απαραίτητη: θα πρέπει να είναι ένα σχετικό URL που αναφέρει τον πόρο του οποίου η έξοδος θα πρέπει να συμπεριληφθεί.

```
<jsp:include page="σχετική-διαδρομή-του-πόρου" />
```

Τα σχετικά URL που δεν ξεκινούν με κάθετο ερμηνεύονται σε σχέση με τη θέση της κύριας σελίδας. Τα σχετικά URL που ξεκινούν με κάθετο ερμηνεύονται σε σχέση με το βασικό κατάλογο της εφαρμογής Ιστού, και *όχι* σε σχέση με τη ρίζα του διακομιστή. Για παράδειγμα, ας υποθέσουμε ότι έχουμε μια σελίδα JSP στην εφαρμογή Ιστού `headlines`, την οποία προσπελάζουμε με το URL `http://ΥπολογιστήςΥπηρεσίας/headlines/sports/table-tennis.jsp`. Το αρχείο table-tennis.jsp βρίσκεται στον υποκατάλογο sports του καταλόγου που χρησιμοποιείται από την εφαρμογή Ιστού `headlines`. Δείτε τώρα τις παρακάτω δύο εντολές συμπερίληψης.

```
<jsp:include page="bios/cheng-yinghua.jsp" />
<jsp:include page="/templates/footer.jsp" />
```

Στην πρώτη περίπτωση το σύστημα θα αναζητήσει το αρχείο cheng-yinghua.jsp στον υποκατάλογο bios του καταλόγου sports (δηλαδή, στον υπο-υποκατάλογο sports/bios του κύριου καταλόγου της εφαρμογής `headlines`). Στη δεύτερη περίπτωση το σύστημα θα αναζητήσει το αρχείο footer.jsp στον υποκατάλογο templates της εφαρμογής `headlines`, και *όχι* στον υποκατάλογο templates της ρίζας του διακομιστή. Η ενέργεια `jsp:include` *ποτέ* δεν κάνει το σύστημα να αναζητήσει αρχεία έξω από την τρέχουσα εφαρμογή Ιστού. Αν έχετε δυσκολία να θυμάστε πώς ερμηνεύει το σύστημα τις διευθύνσεις URL που ξεκινούν με καθέτους, να θυμάστε τον εξής κανόνα: αυτές ερμηνεύονται σε σχέση με την τρέχουσα εφαρμογή Ιστού όταν ο χειρισμός γίνεται από το διακομιστή, ενώ ερμηνεύονται σε σχέση με τη ρίζα του διακομιστή μόνο όταν ο χειρισμός γίνεται από τον πελάτη (το φυλλομετρητή). Για παράδειγμα, η διεύθυνση URL στην εντολή

```
<jsp:include page="/διαδρομή/αρχείο" />
```

ερμηνεύεται στο πλαίσιο της τρέχουσας εφαρμογής Ιστού επειδή η διεύθυνση URL αναλύεται από το διακομιστή· ο φυλλομετρητής ποτέ δεν τη βλέπει. Όμως η διεύθυνση URL στην εντολή HTML

```
<IMG SRC="/διαδρομή/αρχείο" ...>
```

ερμηνεύεται σε σχέση με το βασικό κατάλογο του διακομιστή, επειδή η διεύθυνση URL αναλύεται από το φυλλομετρητή ο οποίος δεν γνωρίζει τίποτε για εφαρμογές Ιστού. Για πληροφορίες σχετικά με τις εφαρμογές Ιστού, δείτε την Ενότητα 2.11.

Σημείωση

Οι διευθύνσεις URL που ξεκινούν με καθέτους ερμηνεύονται διαφορετικά από το διακομιστή και διαφορετικά από το φυλλομετρητή. Ο διακομιστής πάντοτε τις ερμηνεύει σε σχέση με την τρέχουσα εφαρμογή Ιστού. Ο φυλλομετρητής τις ερμηνεύει πάντοτε σε σχέση με τη ρίζα του διακομιστή.

Τέλος, σημειώστε ότι επιτρέπεται να τοποθετήσετε τις σελίδες σας στον κατάλογο WEB-INF. Αν και ο πελάτης δεν επιτρέπεται να προσπελάσει άμεσα τα αρχεία αυτού του καταλόγου, είναι ο διακομιστής, και όχι ο πελάτης, αυτός που προσπελάζει τα αρχεία τα οποία αναφέρονται από την ιδιότητα `page` της εντολής `jsp:include`. Στην πραγματικότητα η τοποθέτηση των συμπεριλαμβανόμενων σελίδων στον κατάλογο WEB-INF είναι μια πρακτική την οποία συνιστούμε: έτσι εμποδίζετε την τυχαία προσπέλασή τους από τον πελάτη (κάτι που είναι κακό, επειδή συνήθως πρόκειται για ημιτελή έγγραφα HTML).

Η προσέγγιση του βιβλίου

Για να εμποδίσετε την ανεξάρτητη προσπέλαση των συμπεριλαμβανόμενων αρχείων, τοποθετήστε τα στον κατάλογο WEB-INF ή σε κάποιον υποκατάλογό του.

Σύνταξη XML και jsp:include

Η ενέργεια `jsp:include` είναι μια από τις πρώτες εντολές JSP που έχουμε δει ότι έχει μόνο σύνταξη XML, χωρίς ισοδύναμη "κλασική" σύνταξη. Αν δεν είστε εξοικειωμένοι με την XML, προσέξτε τρία πράγματα:

- **Τα ονόματα των στοιχείων XML μπορούν να περιέχουν άνω και κάτω τελεία.** Έτσι, μην παραπλανηθείτε από το γεγονός ότι το όνομα του στοιχείου είναι `jsp:include`. Στην πραγματικότητα, η συμβατή με XML εκδοχή όλων των καθιερωμένων στοιχείων JSP ξεκινά με το πρόθεμα (ή χώρο ονομάτων) `jsp`.
- **Στις ετικέτες XML γίνεται διάκριση πεζών και κεφαλαίων χαρακτήρων.** Στην τυπική HTML δεν έχει διαφορά αν πείτε `BODY`, `body`, ή `Body`. Στην XML έχει σημασία. Γι' αυτό, βεβαιωθείτε ότι το `jsp:include` είναι γραμμένο με πεζούς χαρακτήρες.
- **Οι ετικέτες XML πρέπει να κλείνουν ρητά.** Στην HTML υπάρχουν στοιχεία αποδεκτών, όπως το `H1`, που έχουν ετικέτες και αρχής και τέλους (`<H1>...</H1>`), καθώς και μεμονωμένα στοιχεία, όπως τα `IMG` ή `HR`, που δεν έχουν ετικέτες τέλους (`<HR>`). Επιπρόσθετα, οι προδιαγραφές της HTML ορίζουν ότι οι ετικέτες τέλους ορισμένων στοιχείων αποδεκτών (π.χ., `TR`, `P`) είναι προαιρετικές. Στην XML όλα τα στοιχεία είναι στοιχεία αποδεκτών, και οι ετικέτες τέλους δεν είναι ποτέ προαιρετικές. Για λόγους ευκολίας, όμως, μπορείτε να αντικαταστήσετε ένα τμήμα χωρίς σώμα, όπως το

`<blah></blah>`, με το `<blah/>`. Έτσι, όταν χρησιμοποιείτε την εντολή `jsp:include`, φροντίστε να συμπεριλάβετε την κάθετο στο τέλος.

Η ιδιότητα flush

Εκτός από την απαραίτητη ιδιότητα `page`, η `jsp:include` διαθέτει και μια δεύτερη ιδιότητα, τη `flush`, όπως φαίνεται παρακάτω. Αυτή η ιδιότητα είναι προαιρετική· προσδιορίζει αν το ρεύμα εξόδου της κύριας σελίδας θα πρέπει να αδειάζει (flush) πριν από τη συμπερίληψη κάποιας σελίδας (η προεπιλεγμένη τιμή είναι `false`). Προσέξτε όμως ότι στη JSP 1.1 η ιδιότητα `flush` ήταν υποχρεωτική ιδιότητα, και η μόνη έγκυρη τιμή ήταν η `true`.

```
<jsp:include page="σχετική-διαδρομή-του-πόρου" flush="true" />
```

Μια σελίδα με τίτλους ειδήσεων

Ως παράδειγμα μιας τυπικής χρήσης της εντολής `jsp:include`, δείτε την απλή σελίδα με τίτλους ειδήσεων που παρουσιάζεται στη Λίστα 13.1. Οι δημιουργοί της σελίδας μπορούν να αλλάξουν τους τίτλους των ειδήσεων στα αρχεία Item1.html μέχρι Item3.html (Λίστες 13.2 μέχρι 13.4) χωρίς να χρειάζεται να ανανεώσουν την κύρια σελίδα με τις ειδήσεις. Η Εικόνα 13-1 δείχνει το αποτέλεσμα.

Παρατηρήστε ότι τα κομμάτια που συμπεριλαμβάνονται δεν είναι ολοκληρωμένες ιστοσελίδες. Οι συμπεριλαμβανόμενες σελίδες μπορεί να είναι αρχεία HTML, απλά αρχεία κειμένου, σελίδες JSP, ή μικροϋπηρεσίες (όμως με τις σελίδες JSP και τις μικροϋπηρεσίες, συμπεριλαμβάνεται μόνο η έξοδος της σελίδας, και όχι ο πραγματικός κώδικας). Σε όλες τις περιπτώσεις ο πελάτης βλέπει το *συνδυασμένο* αποτέλεσμα. Έτσι, αν τόσο η κύρια σελίδα όσο και οι συμπεριλαμβανόμενες σελίδες περιέχουν ετικέτες όπως `DOCTYPE`, `BODY`, κ.λπ., το αποτέλεσμα θα είναι μη έγκυρη HTML επειδή αυτές οι ετικέτες θα εμφανίζονται δύο φορές στο αποτέλεσμα που βλέπει ο πελάτης. Με τις μικροϋπηρεσίες και τις σελίδες JSP, είναι πάντοτε καλή συνήθεια να προβάλλετε τον πηγαίο κώδικα HTML και να υποβάλλετε τη διεύθυνση URL σε ένα πρόγραμμα επικύρωσης HTML (δείτε την Ενότητα 3.5, "Απλές βοηθητικές κλάσεις δόμησης HTML")./Οταν χρησιμοποιείτε την εντολή `jsp:include`, η συμβουλή αυτή είναι ακόμη πιο σημαντική επειδή οι αρχάριοι συνήθως σχεδιάζουν τόσο την κύρια σελίδα όσο και τις συμπεριλαμβανόμενες σελίδες ως ολοκληρωμένα έγγραφα HTML.

Η προσέγγιση του βιβλίου

Μη χρησιμοποιείτε ολοκληρωμένα έγγραφα HTML για τις συμπεριλαμβανόμενες σελίδες. Συμπεριλάβετε μόνο τις ετικέτες HTML που είναι απαραίτητες στη θέση όπου θα γίνει εισαγωγή των συμπεριλαμβανόμενων σελίδων.

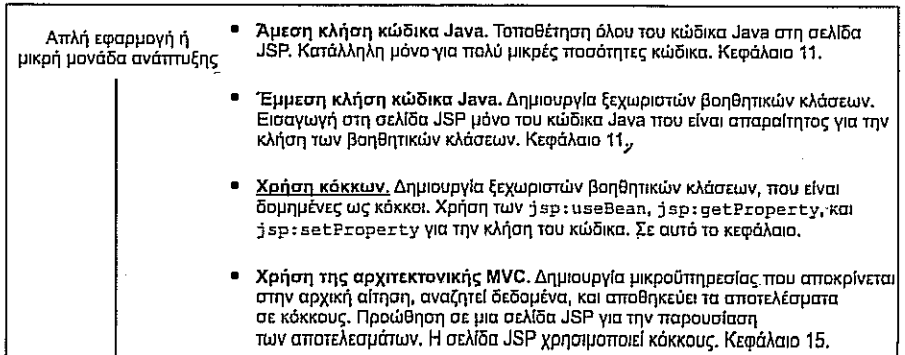
ΧΡΗΣΗ ΣΤΟΙΧΕΙΩΝ JAVABEANS ΣΕ ΕΓΓΡΑΦΑ JSP

Θέματα σε αυτό το Κεφάλαιο

- Κατανόηση των πλεονεκτημάτων των κόκκων
- Δημιουργία κόκκων
- Εγκατάσταση κλάσεων κόκκων στο διακομιστή σας
- Προσπέλαση ιδιοτήτων κόκκων
- Ρητή ρύθμιση των ιδιοτήτων κόκκων
- Αυτόματη ρύθμιση των ιδιοτήτων κόκκων από παραμέτρους αιτήσεων
- Κοινοχρησία κόκκων μεταξύ πολλών διακομιστών και σελίδων JSP

Κεφάλαιο 14

Αυτό το κεφάλαιο εξετάζει την τρίτη γενική στρατηγική για την εισαγωγή δυναμικού περιεχομένου σε σελίδες JSP (δείτε την Εικόνα 14-1): τη χρήση στοιχείων JavaBeans.



Εικόνα 14-1 Στρατηγικές κλήσης κώδικα Java από σελίδες JSP.

14.1 Γιατί να χρησιμοποιήσετε κόκκους;

Πολύ ωραία. Έχετε ήδη κατανοήσει τα πλεονεκτήματα της χρήσης ξεχωριστών κλάσεων Java, σε σύγκριση με την ενσωμάτωση μεγάλων ποσοτήτων κώδικα κατευθείαν στις σελίδες JSP. Όπως εξηγήσαμε στην Ενότητα 11.3 (Περιορισμός της ποσότητας κώδικα Java σε σελίδες JSP), οι ξεχωριστές κλάσεις διευκολύνουν τη συγγραφή, τη μεταγλώττιση, τον έλεγχο, την αποσφαλμάτωση, και την επαναχρησιμοποίηση κώδικα. Τι μπορούν λοιπόν να προσφέρουν οι κόκκοι (beans) το οποίο δεν μπορούν να προσφέρουν οι άλλες κλάσεις; Έτσι κι αλλιώς, οι κόκκοι είναι απλώς κανονικές κλάσεις Java που ακολουθούν μερικές απλές συμβάσεις οι οποίες ορίζονται από τις προδιαγραφές της JavaBeans· οι κόκκοι δεν επεκτείνουν κάποια συγκεκριμένη κλάση, δεν βρίσκονται σε κάποιο ιδιαίτερο πακέτο, και δεν χρησιμοποιούν κάποια ιδιαίτερη διασύνδεση.

Αν και είναι αλήθεια ότι οι κόκκοι είναι απλώς κλάσεις Java που έχουν γραφτεί σε μια τυποποιημένη μορφή, υπάρχουν αρκετά πλεονεκτήματα από τη χρήση τους. Με τους κόκκους γενικότερα, τα εργαλεία οπτικοποιημένου χειρισμού και τα άλλα προγράμματα μπορούν αυτόματα να ανακαλύψουν πληροφορίες σχετικά με τις κλάσεις που ακολουθούν αυτή τη μορφή, και μπορούν να δημιουργήσουν και να διαχειριστούν τις κλάσεις χωρίς να χρειάζεται να γράψει ο χρήστης δικό του κώδικα. Ιδιαίτερα στις σελίδες JSP, η χρήση των στοιχείων JavaBeans παρέχει τρία πλεονεκτήματα σε σχέση με τα μικροσενάρια και τις παραστάσεις JSP που αναφέρονται σε κανονικές κλάσεις Java.

1. **Καθόλου σύνταξη Java.** Με τη χρήση των κόκκων, οι συγγραφείς σελίδων μπορούν να χειρίζονται αντικείμενα Java χρησιμοποιώντας μόνο σύνταξη τύπου XML: χωρίς παρενθέσεις, ελληνικά ερωτηματικά, και αγκύλες. Αυτό προωθεί έναν ισχυρότερο διαχωρισμό μεταξύ περιεχομένου και παρουσίασης, και είναι ιδιαίτερα χρήσιμο στις μεγάλες ομάδες ανάπτυξης που έχουν ξεχωριστούς προγραμματιστές Ιστού και Java.
2. **Απλούστερη κοινοχρησία αντικειμένων.** Όταν χρησιμοποιείτε κόκκους JSP, μπορείτε ευκολότερα να έχετε κοινόχρηστα αντικείμενα μεταξύ πολλών σελίδων ή μεταξύ αιτήσεων, παρά αν χρησιμοποιούσατε τον αντίστοιχο ρητό κώδικα Java.
3. **Εύχρηστη αντιστοίχιση μεταξύ παραμέτρων αιτήσεων και ιδιοτήτων αντικειμένων.** Οι κόκκοι JSP απλοποιούν σε μεγάλο βαθμό τη διαδικασία ανάγνωσης των παραμέτρων αιτήσεων, τη μετατροπή τους από αλφαριθμητικά, και την τοποθέτηση των αποτελεσμάτων μέσα σε αντικείμενα.

14.2 Τι είναι οι κόκκοι;

Όπως αναφέραμε, οι κόκκοι είναι απλώς κλάσεις Java που έχουν γραφτεί σε μια τυποποιημένη μορφή. Η πλήρης εξέταση των JavaBeans είναι έξω από τους στόχους αυτού του βιβλίου, αλλά για το σκοπό της χρήσης τους σε σελίδες JSP αυτά που θα πρέπει να γνωρίζετε για τους κόκκους είναι τρία απλά σημεία τα οποία περιγράφονται στον κατάλογο που ακολουθεί. Αν θέλετε να μάθετε περισσότερες πληροφορίες γενικά για τους κόκκους, επιλέξτε κάποιο από τα πολλά

βιβλία που κυκλοφορούν σχετικά με το θέμα ή δείτε την τεκμηρίωση και τα εκπαιδευτικά μαθήματα στη διεύθυνση http://java.sun.com/products/javabeans/docs/.

- **Η κλάση του κόκκου πρέπει να έχει (προεπιλεγμένη) συνάρτηση δόμησης χωρίς ορίσματα.** Μπορείτε να ικανοποιήσετε αυτή την απαίτηση είτε ορίζοντας ρητά μία τέτοια συνάρτηση δόμησης (constructor) είτε παραλείποντας όλες τις συναρτήσεις δόμησης, οπότε θα δημιουργηθεί αυτόματα μια συνάρτηση δόμησης χωρίς ορίσματα. Η προεπιλεγμένη συνάρτηση δόμησης θα κληθεί όταν τα στοιχεία JSP δημιουργούν κόκκους. Στην πραγματικότητα, όπως θα δούμε στο Κεφάλαιο 15 "Ολοκλήρωση μικροϋπηρεσιών και σελίδων JSP: Η αρχιτεκτονική Model View Controller (MVC)", είναι αρκετά συνηθισμένο για μια μικροϋπηρεσία να δημιουργεί έναν κόκκο από τον οποίο η σελίδα JSP θα αναζητεί απλώς δεδομένα. Σε αυτή την περίπτωση δεν λαμβάνεται υπόψη η απαίτηση ότι ο κόκκος πρέπει να έχει συνάρτηση δόμησης χωρίς ορίσματα.
- **Η κλάση κόκκου δεν πρέπει να έχει δημόσιες μεταβλητές στιγμιοτύπου (πεδία).** Για να είναι μια κλάση κόκκος προσπελάσιμος από σελίδες JSP, θα πρέπει να χρησιμοποιεί μεθόδους προσπέλασης (accessor methods) και να μην επιτρέπει την άμεση πρόσβαση στις μεταβλητές στιγμιοτύπου. Ελπίζουμε ότι ήδη ακολουθείτε αυτή την πρακτική, αφού πρόκειται για μια σημαντική σχεδιαστική στρατηγική στον αντικειμενοστρεφή προγραμματισμό. Γενικά, η χρήση μεθόδων προσπέλασης σας επιτρέπει τρία πράγματα, χωρίς να χρειάζεται να αλλάξουν τον κώδικά τους οι χρήστες της κλάσης σας: (α) επιβολή περιορισμών στις τιμές των μεταβλητών (π.χ., να μην επιτρέπει η μέθοδος `setSpeed` της κλάσης `Car` αρνητικές τιμές)· (β) αλλαγή των εσωτερικών δομών δεδομένων (π.χ., εσωτερική αλλαγή των μονάδων μέτρησης από το Αγγλικό σύστημα στο μετρικό, αλλά με παράλληλη διατήρηση των μεθόδων `getSpeedInMPH` και `getSpeedInKPH`)· (γ) αυτόματη εκτέλεση παράπλευρων ενεργειών όταν αλλάζουν οι τιμές (π.χ., ενημέρωση της διασύνδεσης του χρήστη όταν καλείται η μέθοδος `setPosition`).
- **Οι διατηρούμενες τιμές θα πρέπει να προσπελάζονται μέσω μεθόδων με ονόματα `getXxx` και `setXxx`.** Για παράδειγμα, αν η κλάση σας `Car` αποθηκεύει τον τρέχοντα αριθμό των επιβατών, μπορεί να έχετε τις μεθόδους `getNumPassengers` (για τη λήψη του αριθμού των επιβατών, η οποία δεν δέχεται ορίσματα και επιστρέφει μια τιμή `int`) και τη `setNumPassengers` (για τον καθορισμό του αριθμού των επιβατών, η οποία θα δέχεται μια τιμή `int` και θα επιστρέφει τύπο `void`). Σε μια τέτοια περίπτωση η κλάση `Car` λέγεται ότι έχει μια *ιδιότητα* (property) με το όνομα `numPassengers` (προσέξτε το πεζό n στο όνομα της ιδιότητας και το κεφαλαίο `N` στο όνομα των μεθόδων). Αν η κλάση έχει μια μέθοδο `getXxx` αλλά όχι αντίστοιχη μέθοδο `setXxx`, τότε η κλάση λέμε ότι έχει μια ιδιότητα "μόνο ανάγνωσης", την `xxx`.
  Η μόνη εξαίρεση σε αυτή τη σύμβαση ονομασίας των μεθόδων είναι στις ιδιότητες τύπου `boolean`: σε αυτές επιτρέπεται να χρησιμοποιήσουν μια μέθοδο με όνομα `isXxx` για την αναζήτηση των τιμών τους. Έτσι, για παράδειγμα, η κλάση `Car` μπορεί να διαθέτει τη μέθοδο `isLeased` (η οποία δεν δέχεται ορίσματα και επιστρέφει τιμή `boolean`) και τη μέθοδο `setLeased` (η οποία δέχεται ένα όρισμα `boolean` και έχει τύπο

επιστρεφόμενης τιμής `void`), και θα μπορούσαμε να πούμε ότι έχει μια ιδιότητα `boolean` με το όνομα `leased` (και πάλι, προσέξτε τον πεζό χαρακτήρα που βρίσκεται στην αρχή του ονόματος της ιδιότητας).

Αν και μπορείτε να χρησιμοποιήσετε μικροσενάρια ή παραστάσεις JSP για να προσπελάσετε τις διάφορες μεθόδους μιας κλάσης, οι τυπικές ενέργειες JSP για την προσπέλαση κόκκων μπορούν μόνο να κάνουν χρήση των συμβάσεων ονομασίας `getXxx`/`setXxx` ή `isXxx`/`setXxx`.

14.3 Χρήση κόκκων: Βασικές εργασίες

Για τη δόμηση και το χειρισμό στοιχείων JavaBeans σε σελίδες JSP θα χρησιμοποιείτε τρεις κύριες εντολές:

- **`jsp:useBean`**. Στην απλούστερη περίπτωση, αυτό το στοιχείο δομεί ένα νέο κόκκο. Κανονικά χρησιμοποιείται με τον εξής τρόπο:

```
<jsp:useBean id="όνομαΚόκκου"
class="πακέτο.Class" />
```

  Αν συμπεριλάβετε την ιδιότητα `scope` (δείτε την Ενότητα 14.6, "Κοινοχρησία κόκκων"), το στοιχείο `jsp:useBean` θα μπορεί είτε να δομήσει ένα νέο κόκκο, είτε να προσπελάσει έναν κόκκο που υπάρχει ήδη.
- **`jsp:getProperty`**. Αυτό το στοιχείο διαβάζει και εμφανίζει στην έξοδο την τιμή μιας ιδιότητας κόκκου. Η ανάγνωση μιας ιδιότητας αποτελεί ένα συντομευμένο τρόπο κλήσης της μεθόδου που έχει το όνομα `getXxx`. Αυτό το στοιχείο χρησιμοποιείται με τον εξής τρόπο:

```
<jsp:getProperty name="όνομαΚόκκου"
property="όνομαΙδιότητας" />
```

- **`jsp:setProperty`**. Αυτό το στοιχείο τροποποιεί μια ιδιότητα κόκκου (με άλλα λόγια, καλεί μια μέθοδο που έχει όνομα `setXxx`). Κανονικά χρησιμοποιείται με τον εξής τρόπο:

```
<jsp:getProperty name="όνομαΚόκκου"
property="όνομαΙδιότητας"
value="τιμήΙδιότητας" />
```

Οι υποενότητες που ακολουθούν παρέχουν λεπτομέρειες γι' αυτά τα στοιχεία.

Δόμηση κόκκων: jsp:useBean

Η ενέργεια `jsp:useBean` σάς επιτρέπει να φορτώσετε έναν κόκκο για να τον χρησιμοποιήσετε σε μια σελίδα JSP. Οι κόκκοι παρέχουν μια πολύ χρήσιμη δυνατότητα, επειδή σας βοηθούν να εκμεταλλευτείτε τη δυνατότητα επαναχρησιμοποίησης των κλάσεων Java χωρίς να θυσιάζετε την ευκολία που προσθέτει η JSP σε σύγκριση με τις μικροϋπηρεσίες.

Η απλούστερη σύνταξη για να καθορίσετε ότι θα πρέπει να χρησιμοποιηθεί ένας κόκκος, είναι η ακόλουθη:

```
<jsp:useBean id="όνομα" class="πακέτο.Class" />
```

Αυτή η εντολή συνήθως σημαίνει "δημιούργησε ένα στιγμιότυπο αντικειμένου της κλάσης που καθορίζεται από την παράμετρο `Class`, και σύνδεσέ το με μια μεταβλητή της `_jspService` το όνομα της οποίας καθορίζεται από το `id`." Σημειώστε όμως ότι χρησιμοποιείτε το πλήρως αναπτυγμένο όνομα της κλάσης — το όνομα κλάσης μαζί με τα συμπεριλαμβανόμενα πακέτα. Αυτή η απαίτηση ισχύει ανεξάρτητα από το αν χρησιμοποιείτε την παράσταση `<%@ page import...%>` για να εισαγάγετε πακέτα.

Προειδοποίηση

Θα πρέπει να χρησιμοποιείτε το πλήρως αναπτυγμένο όνομα κλάσης για την ιδιότητα `class` του στοιχείου `jsp:useBean`.

Έτσι, για παράδειγμα, η ενέργεια JSP

```
<jsp:useBean id="book1" class="coreservlets.Book" />
```

μπορεί κανονικά να θεωρηθεί ισοδύναμη με το μικροσενάριο

```
<% coreservlets.Book book1 = new coreservlets.Book(); %>
```

Εγκατάσταση κλάσεων κόκκων

Ο ορισμός μιας κλάσης κόκκου πρέπει να τοποθετηθεί στους ίδιους καταλόγους όπου μπορούν να εγκαθίστανται μικροϋπηρεσίες, και *όχι* στον κατάλογο που περιέχει το αρχείο JSP. Να θυμάστε απλώς ότι πρέπει να χρησιμοποιείτε πακέτα (δείτε την Ενότητα 11.3 για λεπτομέρειες). Έτσι η κατάλληλη τοποθεσία για μεμονωμένες κλάσεις κόκκων είναι η WEB-INF/classes/*υποκατάλογοςΠουΤαιριάζειΜεΤοΌνομαΤουΠακέτου*, όπως περιγράφεται στις Ενότητες 2.10 (Κατάλογοι εκδίπλωσης για την προεπιλεγμένη εφαρμογή Ιστού: Περίληψη) και 2.11 (Εφαρμογές Ιστού: Μια προεπισκόπηση). Τα αρχεία JAR που περιέχουν κλάσεις κόκκων θα πρέπει να τοποθετούνται στον κατάλογο WEB-INF/lib.

ΟΛΟΚΛΗΡΩΣΗ ΜΙΚΡΟΫΠΗΡΕΣΙΩΝ ΚΑΙ JSP: Η ΑΡΧΙΤΕΚΤΟΝΙΚΗ MODEL VIEW CONTROLLER (MVC)

Κεφάλαιο 15

Θέματα σε αυτό το Κεφάλαιο

- Κατανόηση των ωφελειών από την αρχιτεκτονική MVC
- Χρήση του αντικειμένου `RequestDispatcher` για την υλοποίηση της αρχιτεκτονικής MVC
- Προώθηση αιτήσεων από μικροϋπηρεσίες σε σελίδες JSP
- Χειρισμός σχετικών URL
- Επιλογή μεταξύ των διαφόρων επιλογών εμφάνισης
- Σύγκριση στρατηγικών κοινοχρησίας δεδομένων
- Προώθηση αιτήσεων από σελίδες JSP
- Συμπερίληψη σελίδων αντί για την προώθησή τους

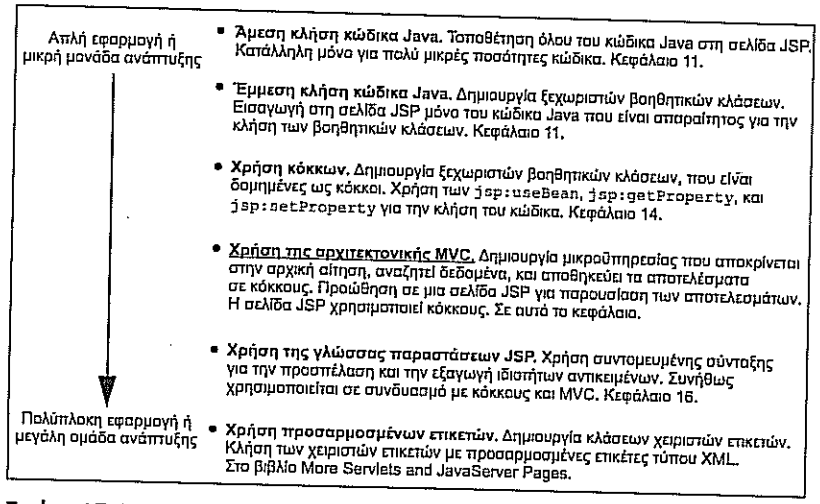
Οι μικροϋπηρεσίες είναι καλές στην *επεξεργασία* δεδομένων: ανάγνωση και έλεγχος δεδομένων, επικοινωνία με βάσεις δεδομένων, επιβολή επιχειρηματικής λογικής, κ.ο.κ. Οι σελίδες JSP είναι καλές στην *παρουσίαση*: δόμηση του κώδικα HTML για την αναπαράσταση των αποτελεσμάτων των αιτήσεων. Αυτό το κεφάλαιο σας δείχνει πώς να συνδυάσετε τις μικροϋπηρεσίες και τις σελίδες JSP έτσι ώστε να εκμεταλλευτείτε καλύτερα τα ισχυρά στοιχεία της κάθε τεχνολογίας.

15.1 Κατανόηση της ανάγκης για την αρχιτεκτονική MVC

Οι μικροϋπηρεσίες είναι σπουδαίες όταν η εφαρμογή σας απαιτεί πολύ προγραμματισμό για την εκτέλεση κάποιας εργασίας. Όπως δείξαμε σε αυτό το βιβλίο, οι μικροϋπηρεσίες μπορούν να χειρίζονται κωδικούς κατάστασης και κεφαλίδες HTTP, να χρησιμοποιούν μπισκότα, να παρακολουθούν συνεδρίες, να αποθηκεύουν πληροφορίες μεταξύ των αιτήσεων, να συμπιέζουν σελίδες, να προσπελάζουν βάσεις δεδομένων, να παράγουν εικόνες JPEG "στον αέρα", και να εκτελούν πολλές άλλες εργασίες με ευελιξία και αποτελεσματικότητα. Όμως η παραγωγή κώδικα HTML με μικροϋπηρεσίες είναι επίπονη, και μπορεί να δώσει ένα αποτέλεσμα το οποίο δύσκολα θα μπορεί να τροποποιηθεί.

Εκεί είναι που έρχονται οι σελίδες JSP: όπως φαίνεται στην Εικόνα 15-1, η τεχνολογία JSP σας επιτρέπει σε μεγάλο βαθμό να διαχωρίσετε την παρουσίαση από το δυναμικό περιεχόμενο. Έτσι μπορείτε να γράψετε κώδικα HTML με τον κανονικό τρόπο, ή ακόμα και να χρησιμοποιήσετε ειδικά εργαλεία για τη γλώσσα HTML και να βάλετε τους κατασκευαστές περιεχομένου Ιστού να δουλέψουν με τα έγγραφα JSP. Οι παραστάσεις, τα μικροσενάρια, και οι δηλώσεις JSP σάς επιτρέπουν να εισάγετε απλό κώδικα Java στη μικροϋπηρεσία που προκύπτει από τη σελίδα

JSP, ενώ οι οδηγίες σάς επιτρέπουν να ρυθμίζετε τη συνολική διάταξη της σελίδας. Για πιο περίπλοκες απαιτήσεις μπορείτε να "περικλείσετε" τον κώδικα Java σε κόκκους, ή ακόμα και να ορίσετε τις δικές σας ετικέτες JSP.



Εικόνα 15-1 Στρατηγικές κλήσης δυναμικού κώδικα από σελίδες JSP.

Τέλεια. Διαθέτουμε όλα όσα έχουμε ανάγκη, έτσι δεν είναι; Μάλλον όχι. Η υπόθεση πίσω από ένα έγγραφο JSP είναι ότι παρέχει *μία μοναδική* συνολική παρουσίαση. Τι θα συμβεί αν θέλετε να δώσετε τελείως διαφορετικά αποτελέσματα με βάση τα δεδομένα που λαμβάνετε; Οι παραστάσεις σεναρίων, οι κόκκοι, και οι προσαρμοσμένες ετικέτες — αν και εξαιρετικά ισχυρά και ευέλικτα εργαλεία — δεν μπορούν να ξεπεράσουν τον περιορισμό ότι η σελίδα JSP ορίζει μια σχετικά σταθερή και ανώτατου επιπέδου παρουσίαση σελίδας. Παρομοίως, τι θα κάνετε αν θέλετε μια περίπλοκη λογική απλώς και μόνο για να προσδιορίσετε τον τύπο δεδομένων που θα ισχύουν στην τρέχουσα περίπτωση; Η JSP δεν είναι πολύ δυνατή σε αυτού του είδους την επιχειρηματική λογική.

Η λύση είναι να χρησιμοποιήσετε *και* μικροϋπηρεσίες *και* σελίδες JSP. Σε αυτή την προσέγγιση, που είναι γνωστή ως Model View Controller (Μοντέλο ελεγκτή προβολής, MVC) ή αρχιτεκτονική Model 2, αφήνετε την κάθε τεχνολογία να επικεντρωθεί σε αυτό στο οποίο υπερέχει. Ο χειρισμός της αρχικής αίτησης γίνεται από μια μικροϋπηρεσία. Η μικροϋπηρεσία ενεργοποιεί τον κώδικα της επιχειρηματικής λογικής και της προσπέλασης δεδομένων και δημιουργεί κόκκους για την αναπαράσταση των αποτελεσμάτων (αυτό είναι το *μοντέλο*). Κατόπιν η μικροϋπηρεσία αποφασίζει ποιες σελίδες JSP είναι κατάλληλες για την παρουσίαση των συγκεκριμένων αποτελεσμάτων και προωθεί την αίτηση εκεί (η σελίδα JSP είναι η *προβολή*). Η μικροϋπηρεσία αποφασίζει ποιος κώδικας επιχειρηματικής λογικής ισχύει και ποια σελίδα JSP θα πρέπει να παρουσιάσει τα αποτελέσματα (η μικροϋπηρεσία είναι ο *ελεγκτής*).

Πλαίσια εργασιών MVC

Η κινητήρια δύναμη πίσω από την προσέγγιση MVC είναι η επιθυμία διαχωρισμού του κώδικα που δημιουργεί και χειρίζεται τα δεδομένα από τον κώδικα που τα παρουσιάζει. Τα βασικά εργαλεία που είναι απαραίτητα για την υλοποίηση του διαχωρισμού σε επίπεδο παρουσίασης είναι τυποποιημένα στην API των μικροϋπηρεσιών, και αποτελούν το θέμα αυτού του κεφαλαίου. Ωστόσο, σε ιδιαίτερα περίπλοκες εφαρμογές, είναι μερικές φορές χρήσιμη η χρήση ενός πιο "περίτεχνου" πλαισίου εργασιών MVC. Το πιο δημοφιλές από αυτά τα πλαίσια εργασιών είναι το Apache Struts· αυτό εξετάζεται λεπτομερώς στο βιβλίο More Servlets and JavaServer Pages. Αν και το Struts είναι χρήσιμο και χρησιμοποιείται ευρέως, δεν θα πρέπει να θεωρήσετε ότι θα πρέπει να χρησιμοποιήσετε το Struts για να εφαρμόσετε την προσέγγιση MVC. Για εφαρμογές με μικρή ή μέτρια πολυπλοκότητα, η υλοποίηση του MVC από το μηδέν με την κλάση `RequestDispatcher` είναι απλή και ευέλικτη. Μην πτοείστε: ξεκινήστε με τη βασική προσέγγιση. Σε πολλές περιπτώσεις θα παραμείνετε σε αυτή τη βασική προσέγγιση για ολόκληρη τη διάρκεια ζωής της εφαρμογής σας. Ακόμα και αν στη συνέχεια αποφασίσετε να χρησιμοποιήσετε το Struts ή κάποιο άλλο πλαίσιο εργασιών MVC, θα αποζημιωθείτε επειδή μεγάλο τμήμα της δουλειάς σας θα ισχύει και στο πλαίσιο εργασιών.

Αρχιτεκτονική ή προσέγγιση

Ο όρος "αρχιτεκτονική" συνήθως αναφέρεται στο "συνολικό σχεδιασμό ενός συστήματος". Αν και πολλά συστήματα έχουν σχεδιαστεί με την MVC στον πυρήνα τους, δεν είναι απαραίτητο να ξανασχεδιάσετε το σύστημά σας για να κάνετε χρήση της προσέγγισης MVC. Καθόλου. Είναι αρκετά συνηθισμένο στις εφαρμογές να γίνεται ο χειρισμός ορισμένων αιτήσεων από μικροϋπηρεσίες, άλλων αιτήσεων από σελίδες JSP, και άλλων αιτήσεων από μικροϋπηρεσίες και σελίδες JSP που δουλεύουν από κοινού, όπως περιγράφεται σε αυτό το κεφάλαιο. Μη θεωρήσετε ότι θα πρέπει να αλλάξετε ολόκληρη την αρχιτεκτονική του συστήματός σας μόνο και μόνο για να χρησιμοποιήσετε την προσέγγιση MVC: αρχίστε να την εφαρμόζετε σε εκείνα τα τμήμα της εφαρμογής σας όπου ταιριάζει καλύτερα.

15.2 Υλοποίηση της αρχιτεκτονικής MVC με την κλάση RequestDispatcher

Το πιο σημαντικό σημείο σχετικά με την αρχιτεκτονική MVC είναι η ιδέα του διαχωρισμού του επιπέδου της επιχειρηματικής λογικής και της προσπέλασης δεδομένων από το επίπεδο της παρουσίασης. Η σύνταξη είναι αρκετά απλή, και στην πραγματικότητα θα πρέπει να είστε ήδη εξοικειωμένοι με το μεγαλύτερο τμήμα της. Ακολουθεί μία σύντομη σύνοψη των απαραίτητων βημάτων, ενώ οι υποενότητες που ακολουθούν παρουσιάζουν τις λεπτομέρειες.

1. **Ορισμός κόκκων για την αναπαράσταση των δεδομένων.** Όπως γνωρίζετε από την Ενότητα 14.2, οι κόκκοι είναι απλώς αντικείμενα Java που ακολουθούν μερικές απλές

συμβάσεις. Το πρώτο σας βήμα είναι να ορίσετε τους κόκκους που θα αντιπροσωπεύουν τα αποτελέσματα τα οποία θα παρουσιαστούν στο χρήστη.

2. **Χρήση μιας μικροϋπηρεσίας για το χειρισμό των αιτήσεων.** Στις περισσότερες περιπτώσεις η μικροϋπηρεσία θα διαβάζει τις παραμέτρους αίτησης, όπως περιγράφεται στο Κεφάλαιο 4.
3. **Συμπλήρωση των κόκκων.** Η μικροϋπηρεσία καλεί τον ειδικό κώδικα της εφαρμογής (τον κώδικα επιχειρηματικής λογικής) ή τον κώδικα προσπέλασης δεδομένων (δείτε το Κεφάλαιο 17) για να λάβει τα αποτελέσματα. Τα αποτελέσματα τοποθετούνται στους κόκκους που ορίστηκαν στο βήμα 1.
4. **Αποθήκευση του κόκκου στο αντικείμενο αίτησης, συνεδρίας, ή πλαισίου μικροϋπηρεσίας.** Η μικροϋπηρεσία καλεί τη μέθοδο `setAttribute` για το αντικείμενο αίτησης, συνεδρίας, ή πλαισίου μικροϋπηρεσίας, με στόχο την αποθήκευση μιας αναφοράς στους κόκκους που αντιπροσωπεύουν τα αποτελέσματα της αίτησης.
5. **Προώθηση της αίτησης σε μια σελίδα JSP.** Η μικροϋπηρεσία προσδιορίζει ποια σελίδα JSP είναι κατάλληλη για την περίσταση και χρησιμοποιεί τη μέθοδο `forward` της κλάσης `RequestDispatcher` για να μεταφέρει τον έλεγχο σε αυτή τη σελίδα.
6. **Εξαγωγή των δεδομένων από τους κόκκους.** Η σελίδα JSP προσπελάζει τους κόκκους με το στοιχείο `jsp:useBean` και την ιδιότητα `scope` που ταιριάζει με την περίπτωση του βήματος 4. Στη συνέχεια η σελίδα χρησιμοποιεί το στοιχείο `jsp:getProperty` για να εμφανίσει τις ιδιότητες των κόκκων. Η σελίδα JSP ούτε δημιουργεί ούτε τροποποιεί τον κόκκο· απλώς εξάγει και εμφανίζει τα δεδομένα τα οποία δημιούργησε η μικροϋπηρεσία.

Ορισμός κόκκων για την αναπαράσταση των δεδομένων

Οι κόκκοι είναι αντικείμενα Java που ακολουθούν ορισμένες απλές συμβάσεις. Σε αυτή την περίπτωση, επειδή τους κόκκους θα τους δημιουργήσει μια μικροϋπηρεσία ή κάποια άλλη ρουτίνα Java (και ποτέ μια σελίδα JSP), δεν λαμβάνεται υπόψη η απαίτηση για συνάρτηση δόμησης χωρίς ορίσματα. Έτσι τα αντικείμενά σας θα πρέπει απλώς να ακολουθήσουν τις κανονικές, συνιστώμενες πρακτικές ώστε να είναι ιδιωτικές οι μεταβλητές στιγμιοτύπου και να χρησιμοποιούνται μέθοδοι προσπέλασης (accessor methods) που να ακολουθούν τη σύμβαση ονομασίας get/set.

Επειδή η σελίδα JSP θα προσπελάσει μόνο τους κόκκους — ούτε θα τους δημιουργήσει, ούτε θα τους τροποποιήσει — μια συνηθισμένη πρακτική είναι ο ορισμός *αντικειμένων τιμών* (value objects): αντικείμενα που αντιπροσωπεύουν αποτελέσματα αλλά έχουν πολύ λίγες ή και καθόλου πρόσθετες λειτουργίες.

Συγγραφή μικροϋπηρεσιών για το χειρισμό αιτήσεων

Αφού οριστούν οι κλάσεις των κόκκων, η επόμενη εργασία είναι να γραφτεί μια μικροϋπηρεσία για την ανάγνωση των πληροφοριών αίτησης. Επειδή με την αρχιτεκτονική MVC η μικροϋπηρεσία αποκρίνεται μόνο στην αρχική αίτηση, χρησιμοποιούνται οι κανονικές προσεγγίσεις για την ανάγνωση των παραμέτρων αίτησης (Κεφάλαιο 4) και των κεφαλίδων αίτησης (Κεφάλαιο 5). Μπορεί να χρησιμοποιηθεί και η μέθοδος συντόμευσης `populateBean` του Κεφαλαίου 4, αλλά θα πρέπει να προσέξετε ότι αυτή η τεχνική συμπληρώνει έναν κόκκο *φόρμας* (ένα αντικείμενο Java που αντιπροσωπεύει τις παραμέτρους της φόρμας), και όχι έναν κόκκο *αποτελέσματος* (ένα αντικείμενο Java που αντιπροσωπεύει τα αποτελέσματα της αίτησης).

Αν και οι μικροϋπηρεσίες χρησιμοποιούν τις κανονικές τεχνικές για την ανάγνωση των πληροφοριών της αίτησης και την παραγωγή των δεδομένων, δεν χρησιμοποιούν τις κανονικές τεχνικές για την παρουσίαση των αποτελεσμάτων. Στην πραγματικότητα, με την προσέγγιση MVC οι μικροϋπηρεσίες δεν δημιουργούν *καμία* έξοδο· ο χειρισμός της εξόδου γίνεται εξολοκλήρου από τις σελίδες JSP. Έτσι οι μικροϋπηρεσίες δεν καλούν τις μεθόδους `response.setContentType`, `response.getWriter`, ή `out.println`.

Συμπλήρωση των κόκκων

Αφού διαβάσετε τις παραμέτρους της φόρμας, τις χρησιμοποιείτε για να προσδιορίσετε τα αποτελέσματα της αίτησης. Αυτά τα αποτελέσματα προσδιορίζονται με έναν τρόπο που αφορά αποκλειστικά τη συγκεκριμένη εφαρμογή. Μπορεί να καλέσετε κάποιον ειδικό κώδικα ανάλογα με την εφαρμογή, να καλέσετε στοιχεία Enterprise JavaBean, ή να εκτελέσετε ένα ερώτημα σε μια βάση δεδομένων. Ανεξάρτητα από τον τρόπο με τον οποίο θα αποκτήσετε τα δεδομένα, θα πρέπει να χρησιμοποιήσετε αυτά τα δεδομένα για να συμπληρώσετε τα αντικείμενα τιμών για τους κόκκους που ορίσατε στο πρώτο βήμα.

Αποθήκευση των αποτελεσμάτων

Έχετε διαβάσει τις πληροφορίες της φόρμας, έχετε δημιουργήσει δεδομένα ειδικά για την αίτηση. Έχετε τοποθετήσει τα δεδομένα σε κόκκους. Τώρα θα πρέπει να αποθηκεύσετε αυτούς τους κόκκους σε μια θέση όπου θα μπορούν να τους προσπελάσουν οι σελίδες JSP.

Οι μικροϋπηρεσίες μπορούν να αποθηκεύσουν δεδομένα για σελίδες JSP σε τρεις κύριες θέσεις: στο `HttpServletRequest`, στο `HttpSession`, και στο `ServletContext`. Αυτές οι θέσεις αποθήκευσης αντιστοιχούν στις τρεις μη προεπιλεγμένες τιμές της ιδιότητας `scope` του στοιχείου `jsp:useBean`. Δηλαδή, στις τιμές `request`, `session`, και `application`.

- **Αποθήκευση δεδομένων τα οποία θα χρησιμοποιήσει η σελίδα JSP μόνο σε αυτή την αίτηση.** Αρχικά η μικροϋπηρεσία θα δημιουργήσει και θα αποθηκεύσει τα δεδομένα με τον εξής τρόπο:

```
ValueObject value = new ValueObject(...);
request.setAttribute("key", value);
```

Στη συνέχεια η μικροϋπηρεσία θα προωθήσει την αίτηση σε μια σελίδα JSP που χρησιμοποιεί τον ακόλουθο κώδικα για να ανακτήσει τα δεδομένα:

```
<jsp:useBean id="key" type"=somePackage.ValueObject"
             scope="request" />
```

Προσέξτε ότι οι *ιδιότητες* αίτησης δεν έχουν καμία σχέση με τις *παραμέτρους* αίτησης ή τις *κεφαλίδες* αίτησης. Οι ιδιότητες αίτησης είναι ανεξάρτητες από τις πληροφορίες που προέρχονται από τον πελάτη· είναι απλώς ειδικές καταχωρίσεις της εφαρμογής σε έναν πίνακα κατακερματισμού (hash table) ο οποίος είναι προσαρτημένος στο αντικείμενο αίτησης. Αυτός ο πίνακας απλώς αποθηκεύει δεδομένα σε μια θέση η οποία μπορεί να προσπελαστεί και από την τρέχουσα μικροϋπηρεσία και από τη σελίδα JSP, αλλά όχι από κάποιον άλλον πόρο ή αίτηση.

- **Αποθήκευση δεδομένων τα οποία θα χρησιμοποιήσει η σελίδα JSP σε αυτή την αίτηση και σε επόμενες αιτήσεις του ίδιου πελάτη.** Αρχικά η μικροϋπηρεσία θα δημιουργήσει και θα αποθηκεύσει τα δεδομένα με τον εξής τρόπο:

```
ValueObject value = new ValueObject(...);
HttpSession session = request.getSession();
session.setAttribute("key", value);
```

Στη συνέχεια η μικροϋπηρεσία θα προωθήσει την αίτηση σε μια σελίδα JSP που χρησιμοποιεί τον ακόλουθο κώδικα για την ανάκτηση των δεδομένων:

```
<jsp:useBean id="key" type="somePackage.ValueObject"
             scope="session" />
```

- **Αποθήκευση δεδομένων τα οποία θα χρησιμοποιήσει η σελίδα JSP σε αυτή την αίτηση και σε επόμενες αιτήσεις οποιουδήποτε πελάτη.** Αρχικά η μικροϋπηρεσία θα δημιουργήσει και θα αποθηκεύσει τα δεδομένα με τον εξής τρόπο:

```
ValueObject value = new ValueObject(...);
getServletContext().setAttribute("key", value);
```

Στη συνέχεια η μικροϋπηρεσία θα προωθήσει την αίτηση σε μια σελίδα JSP που χρησιμοποιεί τον ακόλουθο κώδικα για την ανάκτηση των δεδομένων:

```
<jsp:useBean id="key" type="somePackage.ValueObject"
             scope="application" />
```

Όπως περιγράφεται στην Ενότητα 15.3, ο κώδικας είναι συνήθως συγχρονισμένος έτσι ώστε να εμποδίζεται η τροποποίηση των δεδομένων μεταξύ της μικροϋπηρεσίας και της σελίδας JSP.

Προώθηση αιτήσεων σε σελίδες JSP

Η προώθηση των αιτήσεων γίνεται με τη μέθοδο `forward` της κλάσης `RequestDispatcher`. Για να λάβετε ένα αντικείμενο `RequestDispatcher`, καλείτε τη μέθοδο `getRequestDispatcher` της κλάσης `ServletRequest` δίνοντας μια σχετική διεύθυνση. Επιτρέπεται να καθορίσετε διευθύνσεις στον κατάλογο WEB-INF· αν και οι πελάτες δεν επιτρέπεται να προσπελάσουν άμεσα τα αρχεία του καταλόγου WEB-INF, ο διακομιστής επιτρέπεται να μεταφέρει τον έλεγχο εκεί. Η χρήση μιας θέσης στον κατάλογο WEB-INF εμποδίζει τους πελάτες να προσπελάσουν ακούσια με άμεσο τρόπο τις σελίδες JSP, χωρίς να περάσουν πρώτα από τις μικροϋπηρεσίες που δημιουργούν τα δεδομένα JSP.

Η προσέγγιση του βιβλίου

Αν οι σελίδες JSP έχουν νόημα μόνο μέσα στο πλαίσιο των δεδομένων που παράγονται από μικροϋπηρεσίες, τοποθετήστε τις σελίδες στον κατάλογο WEB-INF. Με αυτόν τον τρόπο οι μικροϋπηρεσίες θα μπορούν να προωθήσουν αιτήσεις στις σελίδες, αλλά οι πελάτες δεν θα μπορούν να τις προσπελάσουν άμεσα.

Αφού αποκτήσετε το αντικείμενο `RequestDispatcher`, χρησιμοποιείτε τη μέθοδο `forward` για να μεταφέρετε τον έλεγχο στην αντίστοιχη διεύθυνση. Ως ορίσματα δίνετε αντικείμενα των κλάσεων `HttpServletRequest` και `HttpServletResponse`. Προσέξτε ότι η μέθοδος `forward` της κλάσης `RequestDispatcher` διαφέρει αρκετά από τη μέθοδο `sendRedirect` της κλάσης `HttpServletRequest` (Ενότητα 7.1). Με τη μέθοδο `forward` δεν υπάρχει επιπλέον ζευγάρι απάντησης/αίτησης, όπως συμβαίνει με τη μέθοδο `sendRedirect`. Έτσι η διεύθυνση URL που εμφανίζεται στο χρήστη δεν αλλάζει όταν χρησιμοποιείτε τη μέθοδο `forward`.

Σημείωση

Όταν χρησιμοποιείτε τη μέθοδο `forward` της κλάσης `RequestDispatcher`, ο πελάτης βλέπει τη διεύθυνση URL της αρχικής μικροϋπηρεσίας, και όχι τη διεύθυνση URL της τελικής σελίδας JSP.

Για παράδειγμα, η Λίστα 15.1 παρουσιάζει ένα τμήμα μικροϋπηρεσίας που προωθεί την αίτηση σε μια από τρεις διαφορετικές σελίδες JSP, ανάλογα με την τιμή της παραμέτρου αίτησης `operation`.

Λίστα 15.1 Παράδειγμα προώθησης αίτησης

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
  throws ServletException, IOException {
  String operation = request.getParameter("operation");
```

Λίστα 15.5 Παράδειγμα προώθησης αίτησης *(συνέχεια)*

```
  if (operation == null) {
    operation = "unknown";
  }
  String address;
  if (operation.equals("order")) {
    address = "/WEB-INF/Order/jsp";
  } else if (operation.equals("cancel")) {
    address = "/WEB-INF/Cancel.jsp";
  } else {
    address = "/WEB-INF/UnkownOperation.jsp";
  }
  RequestDispatcher dispatcher =
    request.getRequestDispatcher(address);
  dispatcher.forward(request, response);
}
```

Προώθηση σε στατικούς πόρους

Στις περισσότερες περιπτώσεις θα προωθείτε τις αιτήσεις σε σελίδες JSP ή σε άλλες μικροϋπηρεσίες. Σε ορισμένες περιπτώσεις, όμως, μπορεί να θέλετε να στείλετε τις αιτήσεις σε στατικές σελίδες HTML. Για παράδειγμα, σε μια τοποθεσία ηλεκτρονικού εμπορίου οι αιτήσεις που υποδεικνύουν ότι ο χρήστης δεν έχει κάποιον έγκυρο λογαριασμό μπορούν να προωθηθούν σε μια σελίδα λογαριασμών της εφαρμογής, η οποία χρησιμοποιεί φόρμες HTML για τη συγκέντρωση των απαραίτητων πληροφοριών. Με τις αιτήσεις GET, η προώθηση αιτήσεων σε μια στατική σελίδα HTML είναι απολύτως έγκυρη και δεν χρειάζεται καμία ειδική σύνταξη· δίνετε απλώς τη διεύθυνση της σελίδας HTML ως όρισμα στη μέθοδο `getRequestDispatcher`. Επειδή όμως οι αιτήσεις που προωθούνται χρησιμοποιούν την ίδια μέθοδο αίτησης με την αρχική αίτηση, οι αιτήσεις POST δεν μπορούν να προωθηθούν σε κανονικές σελίδες HTML. Η λύση σε αυτό το πρόβλημα είναι η απλή αλλαγή του ονόματος της σελίδας HTML έτσι ώστε να έχει προέκταση .jsp. Η αλλαγή ονόματος του αρχείου somefile.html σε somefile.jsp δεν αλλάζει την έξοδό του για τις αιτήσεις GET· όμως, ενώ το αρχείο somefile.html δεν μπορεί να χειριστεί αιτήσεις POST, το αρχείο somefile.jsp δίνει παρόμοια απάντηση και σε αιτήσεις GET, και σε αιτήσεις POST.

Ανακατεύθυνση αντί για προώθηση

Η καθιερωμένη προσέγγιση MVC είναι η χρήση της μεθόδου `forward` της κλάσης `RequestDispatcher` για τη μεταφορά του ελέγχου από τη μικροϋπηρεσία στη σελίδα JSP. Όταν όμως χρησιμοποιείτε κοινοχρησία δεδομένων με βάση τη συνεδρία, μερικές φορές είναι προτιμότερη η χρήση της μεθόδου `response.sendRedirect`.

Ακολουθεί μια περίληψη της συμπεριφοράς της μεθόδου `forward`.

- Ο έλεγχος μεταφέρεται εξολοκλήρου στο διακομιστή. Δεν προκαλείται κυκλοφορία στο δίκτυο.
- Ο χρήστης δεν βλέπει τη διεύθυνση της σελίδας JSP προορισμού, και οι σελίδες μπορούν να τοποθετηθούν στον κατάλογο WEB-INF για να εμποδίσουν το χρήστη να τις προσπελάσει χωρίς να περάσει από τη μικροϋπηρεσία που διαμορφώνει τα δεδομένα. Αυτό είναι πολύ χρήσιμο όταν η σελίδα JSP έχει νόημα μόνο στο πλαίσιο των δεδομένων που παράγονται από τη μικροϋπηρεσία.

Ακολουθεί μια περίληψη της μεθόδου `sendRedirect`.

- Η μεταφορά του ελέγχου γίνεται με αποστολή στον πελάτη του κωδικού κατάστασης 302 και μιας κεφαλίδας απάντησης `Location`. Η μεταφορά απαιτεί πρόσθετη κυκλοφορία στο δίκτυο.
- Ο χρήστης βλέπει τη διεύθυνση της σελίδας προορισμού και μπορεί να δημιουργήσει κάποιο σελιδοδείκτη και να την προσπελάσει ανεξάρτητα. Αυτό είναι ωφέλιμο αν η σελίδα JSP έχει σχεδιαστεί ώστε να χρησιμοποιεί προεπιλεγμένες τιμές όταν λείπουν τα δεδομένα. Για παράδειγμα, αυτή η προσέγγιση θα μπορούσε να χρησιμοποιηθεί κατά την επανεμφάνιση μιας ημιτελούς φόρμας HTML, ή όταν συνοψίζονται τα περιεχόμενα ενός καλαθιού αγορών. Και στις δύο περιπτώσεις τα δεδομένα που δημιουργήθηκαν προηγουμένως θα μπορούσαν να εξάγονται από τη συνεδρία του χρήστη, οπότε η σελίδα JSP έχει νόημα ακόμα και για αιτήσεις που δεν περιλαμβάνουν τη μικροϋπηρεσία.

Εξαγωγή δεδομένων από κόκκους

Αφού φθάσει η αίτηση στη σελίδα JSP, η σελίδα JSP χρησιμοποιεί τα στοιχεία `jsp:useBean` και `jsp:getProperty` για να εξαγάγει τα δεδομένα. Στο μεγαλύτερο τμήμα της, αυτή η προσέγγιση είναι ακριβώς όπως περιγράφεται στο Κεφάλαιο 14. Υπάρχουν όμως δύο διαφορές:

- **Η σελίδα JSP δεν δημιουργεί ποτέ τα αντικείμενα.** Η μικροϋπηρεσία, και όχι η σελίδα JSP, θα πρέπει να δημιουργεί όλα τα αντικείμενα δεδομένων. Έτσι, για να υπάρχει εγγύηση ότι η σελίδα JSP δεν θα δημιουργήσει τα αντικείμενα, θα πρέπει να χρησιμοποιήσετε την εντολή

  ```
  <jsp:useBean ... type="package.Class" />
  ```

  αντί του

  ```
  <jsp:useBean ... class="package.Class" />
  ```

- **Η σελίδα JSP δεν θα πρέπει να τροποποιεί τα αντικείμενα.** Κατά συνέπεια θα πρέπει να χρησιμοποιήσετε το στοιχείο `jsp:getProperty`, και όχι το στοιχείο `jsp:setProperty`.

Η εμβέλεια (scope) που θα καθορίσετε θα πρέπει να ταιριάζει με τη θέση αποθήκευσης που χρησιμοποιείται από τη μικροϋπηρεσία. Για παράδειγμα, οι τρεις εντολές που ακολουθούν θα μπορούσαν να χρησιμοποιηθούν για κοινοχρησία με βάση την αίτηση, τη συνεδρία, και την εφαρμογή, αντίστοιχα.

```
<jsp:useBean id="key" type="somePackage.SomeBeanClass"
             scope="request" />
<jsp:useBean id="key" type="somePackage.SomeBeanClass"
             scope="session" />
<jsp:useBean id="key" type="somePackage.SomeBeanClass"
             scope="application" />
```

15.3 Σύνοψη του κώδικα MVC

Σε αυτή την ενότητα συνοψίζουμε τον κώδικα που θα χρησιμοποιούσατε για τις προσεγγίσεις MVC που βασίζονται στην αίτηση, τη συνεδρία, και την εφαρμογή.

Κοινοχρησία δεδομένων με βάση την αίτηση

Στην κοινοχρησία δεδομένων με βάση την αίτηση η μικροϋπηρεσία αποθηκεύει τους κόκκους στο αντικείμενο `HttpServletRequest`, όπου είναι προσπελάσιμοι μόνο από τη σελίδα προορισμού JSP.

Μικροϋπηρεσία

```
ValueObject value = new ValueObject(...);
request.setAttribute("key", value);
RequestDispatcher dispatcher =
  request.getRequestDispatcher("/WEB-INF/SomePage.jsp");
dispatcher.forward(request, response);
```

Σελίδα JSP

```
<jsp:useBean id="key" type"=somePackage.ValueObject"
             scope="request" />
<jsp:getProperty name="key" property="somePackage" />
```

Κοινοχρησία δεδομένων με βάση τη συνεδρία

Στην κοινοχρησία δεδομένων με βάση τη συνεδρία η μικροϋπηρεσία αποθηκεύει τους κόκκους στο αντικείμενο `HttpSession`, όπου είναι προσπελάσιμοι από τον ίδιο πελάτη στη σελίδα προορισμού JSP ή σε άλλες σελίδες.

Μικροϋπηρεσία

```
ValueObject value = new ValueObject(...);
HttpSession session = request.getSession();
session.setAttribute("key", value);
RequestDispatcher dispatcher =
  request.getRequestDispatcher("/WEB-INF/SomePage.jsp");
dispatcher.forward(request, response);
```

Σελίδα JSP

```
<jsp:useBean id="key" type"=somePackage.ValueObject"
             scope="session" />
<jsp:getProperty name="key" property="somePackage" />
```

Κοινοχρησία δεδομένων με βάση την εφαρμογή

Στην κοινοχρησία δεδομένων με βάση την εφαρμογή η μικροϋπηρεσία αποθηκεύει τους κόκκους στο αντικείμενο `ServletContext`, όπου είναι προσπελάσιμοι από οποιαδήποτε μικροϋπηρεσία ή σελίδα JSP της εφαρμογής Ιστού. Για να εξασφαλίσετε ότι η σελίδα JSP θα εξαγάγει τα ίδια δεδομένα που εισήγαγε η μικροϋπηρεσία, θα πρέπει να συγχρονίζετε τον κώδικά σας με τον εξής τρόπο:

Μικροϋπηρεσία

```
synchronize(this) {
  ValueObject value = new ValueObject(...);
  getServletContext().setAttribute("key", value);
  RequestDispatcher dispatcher =
    request.getRequestDispatcher("/WEB-INF/SomePage.jsp");
  dispatcher.forward(request, response);
}
```

Σελίδα JSP

```
<jsp:useBean id="key" type"=somePackage.ValueObject"
             scope="application" />
<jsp:getProperty name="key" property="somePackage" />
```

15.4 Ερμηνεία των σχετικών URL στη σελίδα προορισμού

Αν και η μικροϋπηρεσία μπορεί να προωθήσει την αίτηση σε μια τυχαία θέση στον ίδιο διακομιστή, η διαδικασία είναι αρκετά διαφορετική από εκείνη που χρησιμοποιεί η μέθοδος `sendRedirect` της κλάσης `HttpServletResponse`. Καταρχήν, η μέθοδος `sendRedirect` απαιτεί από τον πελάτη να επανασυνδεθεί με το νέο πόρο, ενώ ο χειρισμός της μεθόδου `forward` της κλάσης `RequestDispatcher` γίνεται εξολοκλήρου στο διακομιστή. Δεύτερον, η μέθοδος `sendRedi-`