



Java Enterprise Web Programming

Αθ. Ανδρούτσος



Java Enterprise Edition (EE)

Java EE

- Η δυνατότητα ανάπτυξης διαδικτυακών εφαρμογών Java (Web/HTTP-based εφαρμογές) παρέχεται από ένα σύνολο βιβλιοθηκών που ονομάζονταν **Java Enterprise Edition (Java EE)** και πλέον **Jakarta EE**
- Τεχνικά οι βιβλιοθήκες αυτές έρχονται σε μορφή jar και **παρέχουν APIs** ώστε να μπορούμε να αναπτύσσουμε Web εφαρμογές over HTTP



Java Enterprise εκδόσεις

Java EE

- Στην Java 1 (1996) οι EE βιβλιοθήκες ήταν μέρος του JDK.
- Στην Java 2 οι βιβλιοθήκες διαχωρίστηκαν σε ξεχωριστά binaries (jar files) και ονομάστηκαν **J2EE 1.2**
Το 2006 η έκδοση ήταν η **J2EE 1.4**
- Από το 2006 μετονομάστηκε σε **Java EE 5** και μετά μέχρι το 2017 οι εκδόσεις έφτασαν στην **Java EE 8**
- Στη συνέχεια η Oracle η οποία έχει τα δικαιώματα της Java, μεταβίβασε τα δικαιώματα της Java EE στο Eclipse Foundation (2018). Λόγω νομικών θεμάτων, η Java EE μετονομάστηκε το 2018 σε **Jakarta EE 8** (2019) και στη συνέχεια **Jakarta 9** (2020), **Jakarta 9.1** (2021) και **Jakarta 10** (2022)



Νέες Εκδόσεις Java ΕΕ

Java ΕΕ

- Τόσο στο παρελθόν η Oracle όσο και τώρα το Eclipse foundation ορίζει επιτροπές (Committees) που εισηγούνται προσθήκες και ανανεώσεις των εκδόσεων
- Αυτά τα σχέδια προτάσεων ονομάζονται JSR (Java Specification Request) και είναι σαν interfaces τα οποία στη συνέχεια **υιοθετούνται** από JCP Executive Committees (JCP = Java Community Process) και **υλοποιούνται** από διάφορους providers (όπως Apache Tomcat, Apache TomEE, Glassfish, Oracle WebLogic, κλπ.)



Βιβλιοθήκες Java EE 8

Java EE

- Η Java EE και πλέον Jakarta EE περιλαμβάνει διάφορες βιβλιοθήκες (που έρχονται ως διαφορετικά jar αρχεία)
- Οι πιο γνωστές είναι τα **Servlets** και τα **Java Server Pages (JSP)**. Άλλες βιβλιοθήκες είναι οι EL (Expression Language), JPA (Java Persistence API), EJB/CDI (Enterprise Java Beans / Contexts and Dependency Injection), JAX-RS (Java API for RESTful Web Services), JTA (Java Transaction API)
- Μπορείτε να δείτε το πλήρες σετ των βιβλιοθηκών στον σύνδεσμο <https://jakarta.ee/release/10/>



Servlets και JSPs

Java EE

- Τα **Servlets** είναι **Java Classes** που λειτουργούν ως **Controllers**. Δέχονται τα http requests που έρχονται από τους Web Browsers στον Web Server και ο Web Server στέλνει (δρομολογεί) τα requests στο κατάλληλο servlet (Java Controller) με βάση το URL που έχει ζητηθεί από τον χρήστη
- Τα **JSPs** είναι αρχεία που περιέχουν **HTML κώδικα** καθώς και **Java κώδικα**. Τελικά μετατρέπονται (transpile) σε HTML κώδικα και επιστρέφουν στον Web Browser του χρήστη
- Επομένως προγραμματιστικά το flow είναι ότι το Servlet δέχεται ένα request και επιστρέφει ένα response (που στην περίπτωση των JSP αρχείων, το response στο payload περιέχει HTML κώδικα)



Περιβάλλοντα Υλοποίησης

Java EE

- Όπως αναφέραμε τα Java EE / Jakarta EE ορίζονται από την Oracle παλαιότερα και πλέον από το Eclipse Foundation και υλοποιούνται από **compatible implementations**
- Μερικά implementations της Java EE / Jakarta EE παρέχουν ο **Apache Tomcat** (υλοποιεί ένα μέρος και όχι το σύνολο των βιβλιοθηκών), *Apache TomEE* (υλοποιεί ένα μεγάλο μέρος των EE βιβλιοθηκών), Oracle Glassfish (υλοποιεί όλες τις EE βιβλιοθήκες), Eclipse GlassGish (Eclipse Foundation) WildFly, JBoss EAP, Oracle WebLogic, Payara Server, κλπ..



Apache Tomcat

Java EE

- Ο Apache Tomcat ξεκίνησε το 1999 από το Apache Software Foundation (non-profit οργανισμός που συντηρείται από την κοινότητα και κάποιες εταιρείες όπως IBM, Microsoft, Google, κ.α.)
- Ο Apache Tomcat είναι ένας **open-source, πολύ δημοφιλής Web Server** που επιπλέον όμως παρέχει implementations για ένα μέρος των Java EE βιβλιοθηκών.
- Είναι Lightweight και παρέχει πολύ γρήγορο χρόνο εκκίνησης



Apache Tomcat & Java EE

Java EE

- Ο Apache Tomcat δεν υλοποιεί το σύνολο των προδιαγραφών της Java EE. Η τρέχουσα non-beta έκδοση του Apache Tomcat είναι **10.1** που υλοποιεί τις προδιαγραφές **Servlet 6.0, JSP 3.1, EL 5.0, WebSocket 2.1, Authentication Spec 3.0** της Jakarta EE 10
- Αν θέλουμε την υλοποίηση και των υπολοίπων προδιαγραφών μπορούμε να επιλέξουμε κάποιον άλλο Application Server (π.χ. WebLogic) που είναι μία μη-ευέλικτη λύση για μεγάλα projects ή μπορούμε επίσης να εισάγουμε με Maven στο project μας επιπλέον προδιαγραφές ως dependencies



Συμβατές εκδόσεις

Java EE

- Μπορείτε να αναφερθείτε στην επίσημη σελίδα του Apache Tomcat για τη συμβατότητα των εκδόσεων του Tomcat με Jakarta EE και Java SDK (βλ. Επόμενη διαφάνεια)
- Για παράδειγμα η Java EE 8 / Jakarta EE 8 υλοποιείται (τα specs που αναφέραμε) από τον Apache 9.x και θέλει Java 8 και πάνω
- Η Jakarta EE 9 υλοποιείται (τα specs που αναφέραμε) από τον Apache 10.0.x αλλά ο Apache 10.0 έχει φτάσει στο end of life. Πρέπει να γίνει upgrade σε Apache Tomcat 10.1
- Η Jakarta 10 υλοποιείται (τα specs που αναφέραμε) από τον Apache 10.1 και θέλει Java 11 και πάνω



Apache Versions & Java EE

Java EE

Apache Tomcat Versions

Apache Tomcat® is an open source software implementation of a subset of the Jakarta EE (formerly Java EE) technologies. Different versions of Apache Tomcat are available for different versions of the specifications. The mapping between [the specifications](#) and the respective Apache Tomcat versions is:

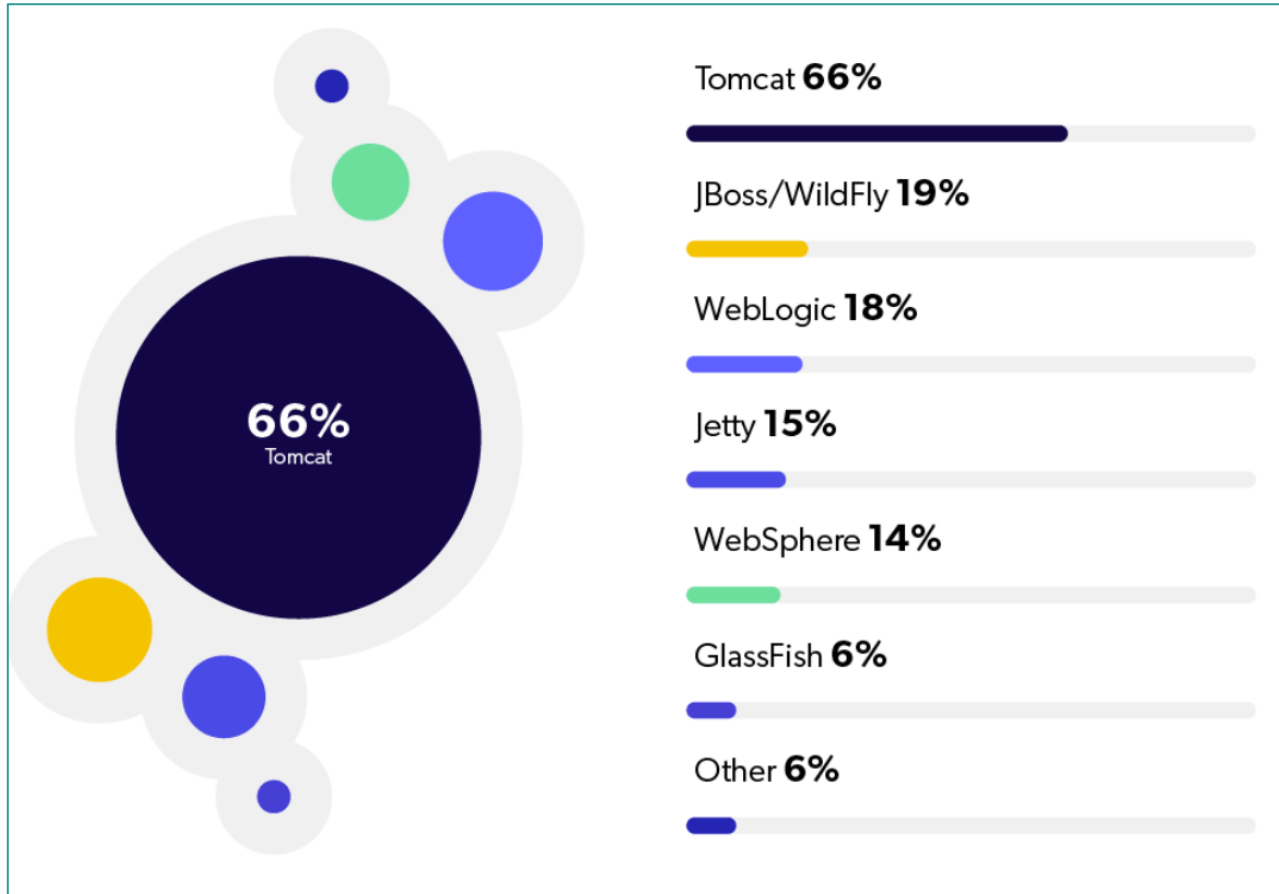
Servlet Spec	JSP Spec	EL Spec	WebSocket Spec	Authentication (JASPIC) Spec	Apache Tomcat Version	Latest Released Version	Supported Java Versions
6.1	4.0	6.0	TBD	TBD	11.0.x	11.0.0-M3 (alpha)	17 and later
6.0	3.1	5.0	2.1	3.0	10.1.x	10.1.6	11 and later
5.0	3.0	4.0	2.0	2.0	10.0.x (superseded)	10.0.27 (superseded)	8 and later
4.0	2.3	3.0	1.1	1.1	9.0.x	9.0.73	8 and later

- Στο παρόν θα χρησιμοποιήσουμε τον Apache 10.1 που είναι συμβατός και με JDK 11 (και μεγαλύτερα JDK) και υλοποιεί τα versions Servlet 6.0, JSP 3.1, EL 5.0, WebSocket 2.1 και Authentication Spec 3.0 (Jakarta EE 10)
- <https://tomcat.apache.org/whichversion.html>



Apache Tomcat vs Application Servers – 2020 stats

Java EE



- Ο Apache Tomcat είναι σταθερά, την τελευταία δεκαετία, πάνω από 65% των προτιμήσεων και θα συνεχίσει αυτή η τάση απομάκρυνσης από μεγάλους μονολιθικούς Application Servers προς πιο ευέλικτες λύσεις



Core Apache Tomcat parts

Java EE

- **Catalina.** Υλοποιεί τα Java Servlets και τα JSPs
- **Coyote.** Είναι connector και υλοποιεί το πρωτόκολλο HTTP 1.1 Δέχεται requests από τους clients (web browsers), δημιουργεί threads (ένα thread για κάθε request), δρομολογεί το request στο κατάλληλο servlet και αποστέλλει πίσω στον client το response
- **Jasper 2.** Μεταγλωττίζει τα JSP αρχεία (που περιέχουν HTML και Java) και επιστρέφει HTML κώδικα



Apache Tomcat / web.xml

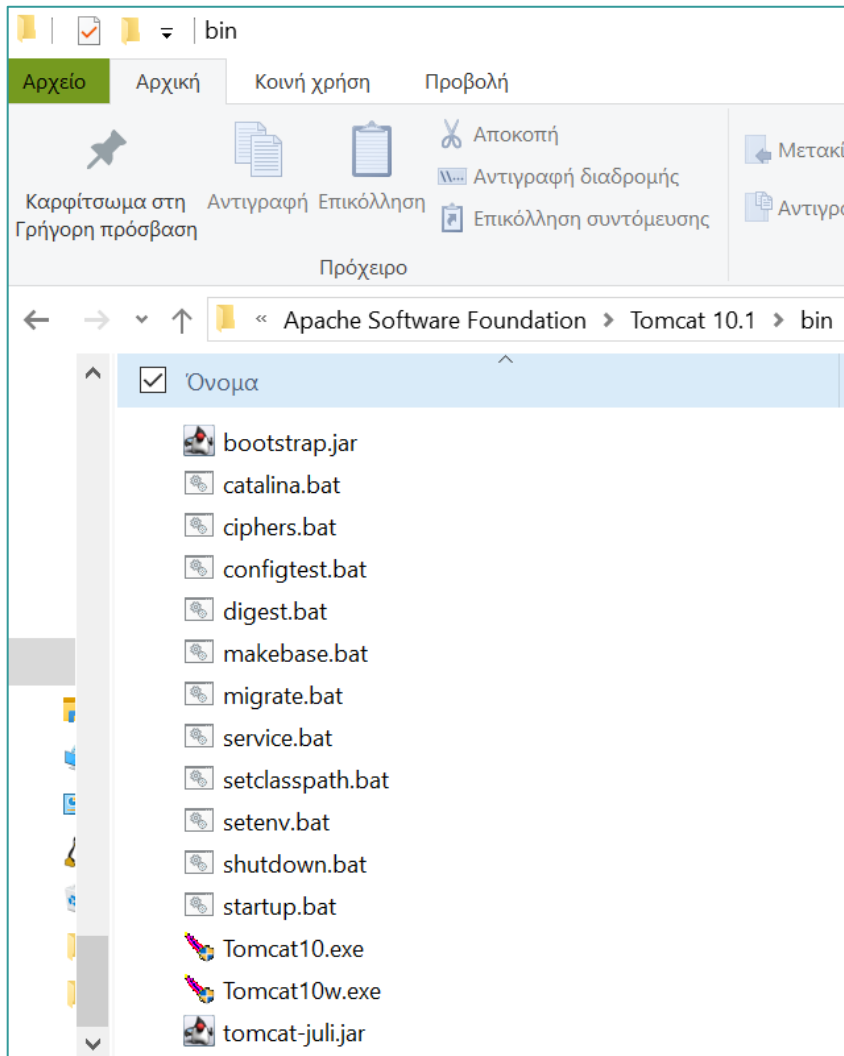
Java EE

- Ο Apache Tomcat είναι επομένως **όχι μόνο Web Server αλλά και ένας Java EE container περιβάλλον που υλοποιεί Servlets και JSP**
- Ένα από τα config αρχεία του Apache Tomcat που συνδέει τον Web Server και τα Servlets είναι το **web.xml** ή αλλιώς deployment descriptor (βρίσκεται στο WEB-INF/ directory)
- Πλέον σε νεότερες εκδόσεις (Java EE 6 / Servlet 3.0) η σύνδεση γίνεται με Annotations (@WebServlet) στον Controller



Apache Tomcat config

Java EE



- Ο Apache μπορεί να τρέξει σε δύο modes
 - Standalone
 - Windows (σε Windows OS)
- Σε standalone mode, μπορούμε να κάνουμε config ποιο JDK να χρησιμοποιήσει ο Apache. Στο αρχείο **setenv.bat** (setenv.sh σε Linux/macOS) ορίζουμε το **path του JDK**
- Σε Windows τρέχει με το Tomcat10w.exe και ως JDK χρησιμοποιεί ότι έχει οριστεί στο registry των Windows



Setenv.sh – Setenv.bat

Java EE


- Σε Linux, macOS συστήματα δίνουμε:

```
export JAVA_HOME= intended java home
```

- Σε Windows δίνουμε:

```
set JAVA_HOME= intended java home
```

- Για παράδειγμα:

 setenv.bat - Σημειωματάριο

Αρχείο Επεξεργασία Μορφή Προβολή Βοήθεια

```
set JAVA_HOME=C:\Program Files\Java\jdk17.0.6_10
```




- Με startup.bat (ή startup.sh) ξεκινάμε τον Tomcat και με shutdown.bat (ή shutdown.sh) σταματάμε
- Διαφορετικά, ο Tomcat μπορεί να ξεκινήσει σε Windows OS ως Windows service με διαφορετικό config ανάλογα τι έχει ως default JDK στο registry



Ορολογία

- ***Servlet*** – Η Java κλάση που λειτουργεί ως Controller και που δέχεται τις αιτήσεις του client και τις εξυπηρετεί
- ***HTML / JSP*** – Όταν αυτό που επιστρέφει ο Server είναι Web Σελίδα, τότε το περιεχόμενο του HTTP Response είναι html και το Content-Type στο HTTP header είναι text/html
- ***Web Container / Application Server*** – Περιβάλλον που μπορεί επιπλέον και εκτελεί προγράμματα Java – Όπως Apache Tomcat, TomEE, WebLogic, κλπ.
- ***Web.xml*** – config αρχείο του Apache Tomcat
- **Annotations** – Εναλλακτικά του config για το URL-Mapping του web.xml χρησιμοποιούνται **annotations** (π.χ. **@WebServlet**) στο servlet



Client - Server

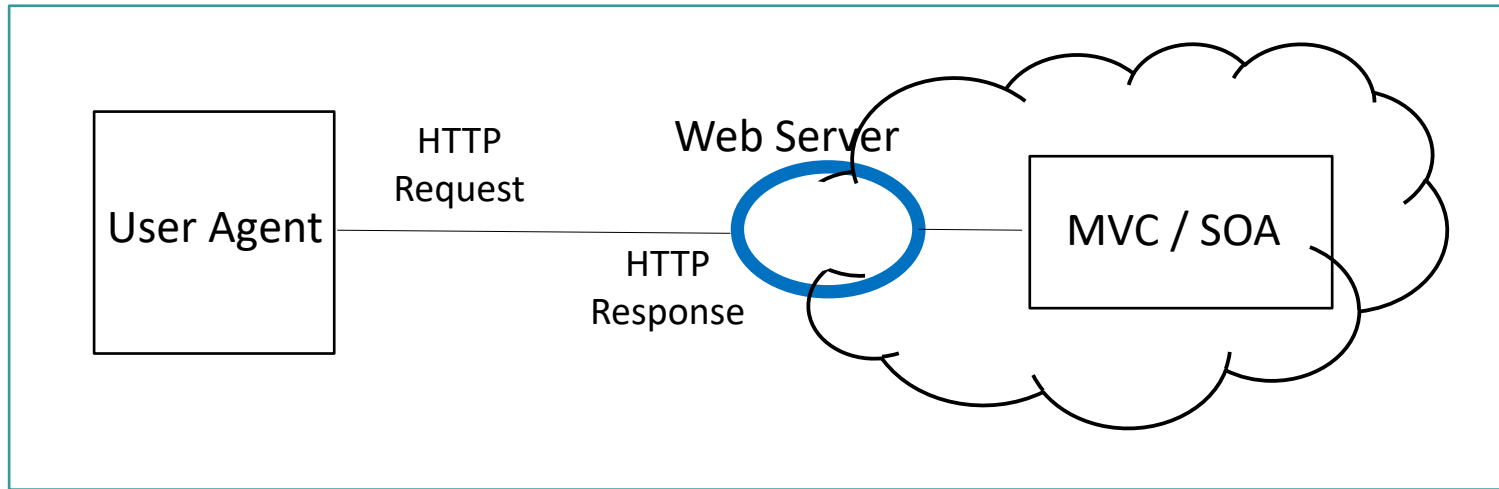
Java EE

- Το βασικό μοντέλο επικοινωνίας σε διαδικτυακές εφαρμογές είναι το μοντέλο **client-server** όπου στην πλευρά του client βρίσκεται ένας *user-agent* όπως για παράδειγμα ένας **web browser** (π.χ. Chrome, Mozilla, Safari, κλπ.) και στην πλευρά του Server βρίσκεται ένας **Web Server** (π.χ. Apache Tomcat)



Web App Architecture Big Picture (1)

Java EE

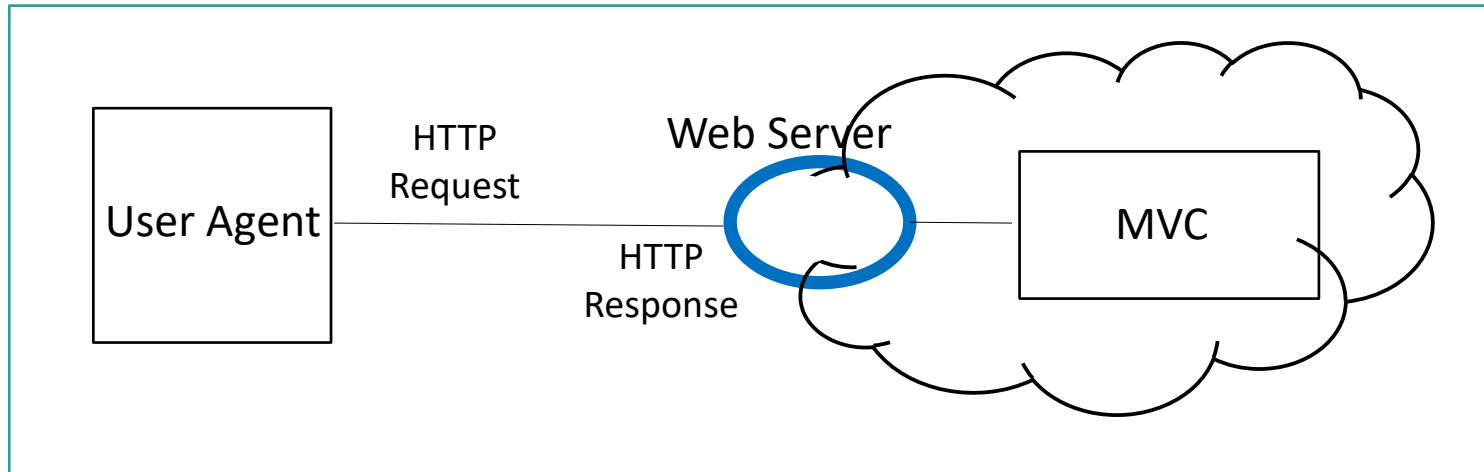


- Στα Web Apps υπάρχει ένας front web server που διενεργεί την επικοινωνία με το frontend
- Ο Web Server δρομολογεί τα requests προς τους controllers του MVC/SOA. Επομένως ο Apache Tomcat θα πρέπει κάπως να γνωρίζει το που θα δρομολογήσει κάθε request



Web App Architecture Big Picture (2)

Java EE

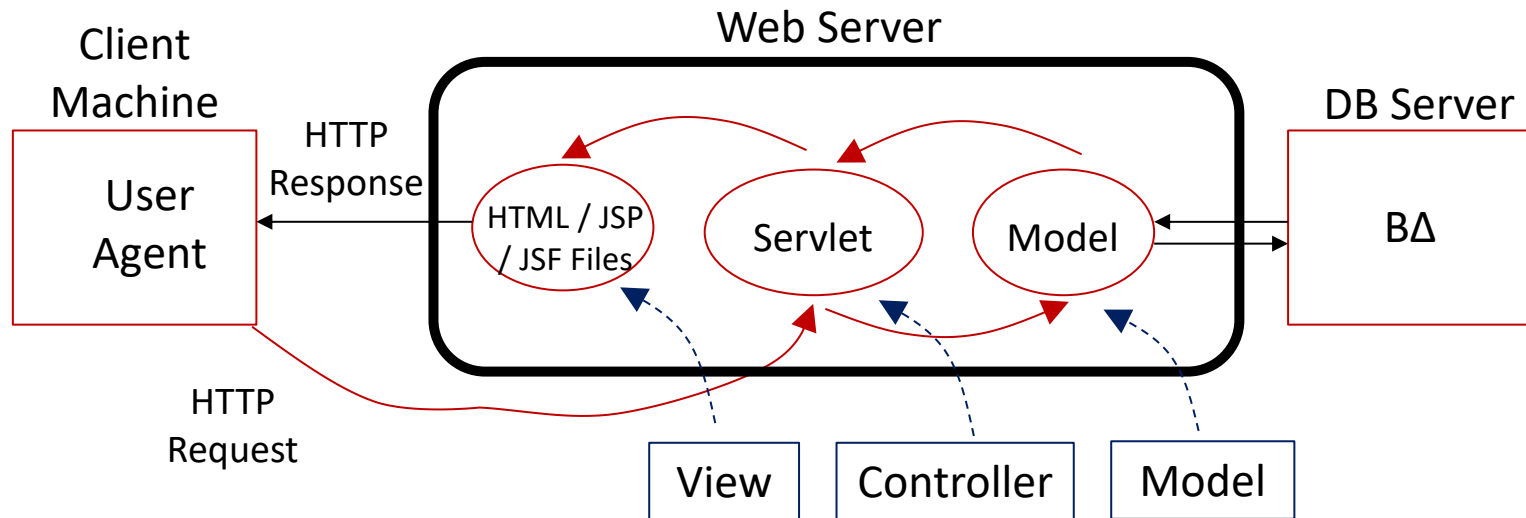


- Η δρομολόγηση των requests μπορεί να γίνει 1) προς ένα φυσικό αρχείο ή 2) προς ένα controller, **με βάση το URL** και την αντιστοίχιση του URL είτε στο όνομα του φυσικού αρχείου είτε στο όνομα του controller
- Η αντιστοίχιση αυτή όσο αφορά τους controllers μπορεί να γίνει είτε στο web.xml ή με annotations
- Για τα αρχεία γίνεται άμεσα με βάση το path του URL



Μοντέλο Διαδικτυακών Εφαρμογών

Java EE



- Ως **User agent** μπορεί να είναι ένας browser (που μπορεί να τρέχει και JavaScript)
- Το **View** είναι HTML / JSP / JSF αρχεία. Τα JSP αρχεία είναι HTML αρχεία που περιέχουν και κώδικα Java είτε ως native κώδικα ή σε μορφή JSTL tags (JSP Standard Tag Library). Τα JSF είναι XHTML αρχεία με JSF (JavaServer Faces) Tags και δίνουν έτοιμα components. Μπορεί επίσης να χρησιμοποιούνται frameworks όπως το Thymeleaf που δίνει native μορφή HTML αρχείων
- Το **Servlet** είναι μία κλάση Java που παίζει το ρόλο του Controller



URL Mapping & Rooting

Java EE

- Ο User Agent τυπικά επικοινωνεί με τον Web Server μέσω μίας URL διεύθυνσης.
- Η URL διεύθυνση αντιστοιχεί σε ένα resource (αρχείο ή controller).
- Ο Web Server με βάση την URL διεύθυνση δρομολογεί (routing) το αίτημα προς εξυπηρέτηση



Rooting

- Όσο αφορά στατικά αρχεία η δρομολόγηση είναι μία απλή αντιστοίχιση του URL σε ένα φυσικό αρχείο
- Όσο αφορά τα Servlets, πρέπει να ορισθεί το URL-Mapping, δηλαδή η αντιστοίχιση μίας URL διεύθυνσης σε ένα όνομα ενός Servlet
- Σε αυτή την περίπτωση το configuration του URL-Mapping με βάση το οποίο γίνεται το routing από τον Web Server, γίνεται είτε σε ένα **XML config αρχείο του Apache Tomcat (web.xml)** ή εναλλακτικά (νεότερος τρόπος) με Annotations (@WebServlet) στην κλάση του Servlet



Resources

- Υπάρχουν τρεις βασικοί τύποι resources που μπορούμε να αναφερθούμε μέσω ενός URI (Uniform Resource Identifier, τα URL είναι URIs)
 - **Στατικά Αρχεία**, όπως στατικά HTML αρχεία, CSS, JavaScript αρχεία, images, κλπ. όπου το URL αντιστοιχίζεται σε ένα φυσικό αρχείο στο δίσκο
 - **Web API**, όπου το URL αντιστοιχίζεται σε ένα Servlet (controller) ο οποίος με τη σειρά του επιστρέφει web σελίδες
 - **Web Services**, όπου το URL αντιστοιχίζεται σε ένα Servlet (Controller) το οποίο επιστρέφει δεδομένα σε μορφή JSON ή XML



Τύποι Web Apps

Java EE

- Αν θα θέλαμε να κατηγοριοποιήσουμε τις web εφαρμογές, θα είχαμε δύο κατηγορίες
 - Web Εφαρμογές όπου ο **Controller επιστρέφει web σελίδες** και επομένως η εφαρμογή απαρτίζεται από ένα σύνολο web σελίδων που φτιάχνονται στο backend και *σερβίρονται* προς το frontend (**Server –Side Rendering - SSR**)
 - RESTful Web Services, όπου ο **Controller επιστρέφει JSON ή XML** και οι σελίδες δημιουργούνται μπροστά στον client με JavaScript (**Client-Side Rendering - CSR**)



Servlets

Java EE

- Κλάσεις της Java που κάνουν **extend** το **HttpServlet abstract class** της Java EE και λειτουργούν ως Controllers
- Τρέχουν κάτω από ένα *Web Container* (Web Server + Java EE Runtime που υλοποιεί μέρος των Java EE Specs) ή κάποιο *Application Server* (Web Server και Java EE Runtime που υλοποιεί όλα τα Java EE Specs)
- Τα Servlets είναι controllers (με αναφορά στο μοντέλο MVC) που λαμβάνουν πρώτοι το αίτημα του πελάτη και το εξυπηρετούν



HTTP / HTTP Server

Java EE

- Το βασικό πρωτόκολλο επικοινωνίας ανάμεσα στον User Agent και τον Web Server είναι το HTTP (Hypertext Transfer Protocol)
- Όπως αναφέραμε όταν επικοινωνεί ο client με ένα Web Server (**HTTP Server**), του ζητάει είτε σελίδα (.jsp) ή και άλλες μορφές αρχείων) ή δεδομένα που τυπικά αποστέλλονται σε μορφή XML ή JSON



Επιστρεφόμενα Δεδομένα (1)

Java EE

- Όταν επιστρέφονται HTML/JSP σελίδες από ένα Web Server, απευθύνονται σε ανθρώπους (Web Browsers)
- Όταν επιστρέφονται πληροφορίες XML ή JSON, απευθύνονται σε εφαρμογές (JavaScript με AJAX), που επεξεργάζονται τις πληροφορίες και μετά τις εμφανίζουν προς ανάγνωση σε Human-readable μορφή σε Web Browsers



Επιστρεφόμενα Δεδομένα (2)

Java EE

- Αν δώσουμε για παράδειγμα www.aueb.gr/courses-servlet τότε αυτό που ζητάμε είναι από ένα Servlet να επιστρέψει είτε μία σελίδα ή δεδομένα (XML, JSON)
- Αν από το Servlet επιστρέφονται δεδομένα, τότε ο όρος που χρησιμοποιούμε για το service που παρέχει το Servlet είναι **Web Service**



Επιστρεφόμενα Δεδομένα (3)

Java EE

- Όταν το web service παρέχει δεδομένα σε μία Representational State Transfer (REST) μορφή όπως για παράδειγμα JSON μέσω HTTP τότε αναφερόμαστε σε **Restful Web Services**
- Όταν η υπηρεσία παρέχεται μέσω του πρωτοκόλλου SOAP (Simple Object Access Protocol) αναφερόμαστε σε **SOAP Web Services που χρησιμοποιούν XML**



XML / JSON Μορφή

Java EE

- Αν για παράδειγμα ζητήσουμε από ένα Web Service τα στοιχεία ενός Καθηγητή με id=1, τότε αυτά επιστρέφονται ως εξής

XML

```
<teacher>
  <id>1</id>
  <name>Th. Andr.</name>
</teacher>
```

JSON

```
{
  "id": 1,
  "name": "Th. Andr."
}
```




XML vs JSON

- Σχετικά με τις δύο μορφές αναπαράστασης δεδομένων XML και JSON για τη μεταφορά τους μέσω δικτύου είναι προτιμότερη η μορφή JSON γιατί:
 - είναι πιο σύντομη
 - μπορεί να αναπαραστήσει πίνακες
 - Το συντακτικό είναι παρόμοιο με της JavaScript
 - Μπορεί να επεξεργαστεί ευκολότερα με JavaScript



Αποστολή Δεδομένων

Java EE

- Όταν πατάμε το button μίας φόρμας (με `type="submit"`) στον client μπορούμε να αποστείλουμε τα δεδομένα της φόρμας στον server
- Τα δεδομένα μπορούν να αποσταλούν με HTTP methods όπως: *get*, *post*, *put*, *delete*
- Το default HTTP method είναι *get*



HTTP Methods & REST API

Java EE

- Σε όρους REST API, τα semantics της GET είναι να φέρει πίσω ένα resource, της POST είναι insert, της PUT είναι update και της DELETE είναι delete
- Επομένως όταν κάνουμε GET ζητάμε ένα resource. Αν το resource είναι στοιχεία μίας ΒΔ, τότε μπορεί να θέλουμε να εκφράσουμε και ένα query μέσω του URL της GET



POST

- Παράδειγμα αποστολής φόρμας με τη μέθοδο *get*
- `<form action="{pageContext.request.contextPath}/get-student" method="POST">`
First name: `<input type="text" name="firstname">`
Last name: `<input type="text" name="lastname">`
`<input type="submit" value="Get Student">`
`</form>`
- Η διεύθυνση **`/insert-student`** είναι **relative σε σχέση με το context path**. Επειδή το context root μπορεί να αλλάζει δεν μπορεί να είναι absolute path. Η μεταβλητή `{pageContext.request.contextPath}` δίνει το context path ή `""` αν το context path είναι το default (root) context.
- Το τελικό URL που σχηματίζεται είναι (αν το context root είναι serverapp):
- <http://localhost:8080/serverapp/get-student>



GET και Query Strings

Java EE

- Το μέρος της διεύθυνσης μετά το ? είναι το query string το οποίο αποτελείται από ζεύγη key=value που συνδέονται με το & (AND)
- Επειδή τα query strings είναι εμφανή και επομένως δεν μπορούμε να στείλουμε ευαίσθητα δεδομένα ενώ επίσης το URL έχει περιορισμένο length (2048 characters) καλό θα ήταν να χρησιμοποιούνται στο πλαίσιο αυτό μόνο για non-critical functionality όπως sorting, filtering, pagination των search results



Query Strings (1)

Java EE

- Σύμφωνα με τα HTTP Specs (RFC 3986) το μέρος του URI που αφορά το scheme και το host είναι case-insensitive αλλά θα πρέπει να γίνεται normalize σε lowercase χαρακτήρες ενώ το υπόλοιπο μέρος (path και query string) είναι case-sensitive
- Επομένως, το HTTP://AUEB.gr είναι ίδιο με <http://aueb.gr> ενώ το ?key=value είναι διαφορετικό από το ?KEY = Value



Query Strings (2)

- Τα query strings:
 - Θα πρέπει να είναι μικρά (short) και readable
 - Τα keys να είναι descriptive
 - Με lowercase ως convention μιας και είναι case-sensitive
 - Με URL-Encoding στους ειδικούς χαρακτήρες (space, &, /, =, :, @, ?, \$, #, κλπ.)
 - Να μην περιέχουν ευαίσθητες πληροφορίες (π.χ. password)
 - Να χρησιμοποιούνται για non-critical functionality
 - Να ελέγχονται (validation) ώστε να αποφεύγεται XSS (Cross site scripting)



URL Encoding

Java EE

- Ονομάζεται και **percent-encoding**. Πρόκειται για μηχανισμό encoding ειδικών χαρακτήρων ενός URL
- Το σχήμα του encoding είναι ότι ο ειδικός χαρακτήρας αντιστοιχίζεται σε ένα % που ακολουθείται από την δεκαεξαδική ASCII τιμή του ειδικού χαρακτήρα
- Για παράδειγμα το # αντιστοιχίζεται στο %23 μιας και στο ASCII το # είναι το Hex 23



Post / put / delete

Java EE

- Για όλους τους προηγούμενους λόγους οι πληροφορίες από φόρμες για εισαγωγή / update / delete θα πρέπει στο πλαίσιο αυτό να περνάνε μέσα στο HTTP Packet και όχι στο URL
- Για εισαγωγή μπορούμε να χρησιμοποιούμε POST, για update την PUT και για διαγραφή την DELETE
- Τα παραπάνω είναι τα semantics των HTTP methods σύμφωνα με το REST, ωστόσο τεχνικά μπορούμε με POST να κάνουμε update και delete



POST (1)

- Παράδειγμα φόρμας με τη μέθοδο post
- `<form action=" ${pageContext.request.contextPath}/add-student" method="POST">`
 Firstname: `<input type="text" name="firstname">
`
 Last name: `<input type="text" name="lastname">
`
 `<input type="submit" value="Submit">`
 `</form>`
- `/add-student` είναι το URL-Mapping για το Servlet που θα χειριστεί τα δεδομένα της φόρμας



POST (2)

- Αν υποθέσουμε ότι η URL διεύθυνση στην οποία έχει γίνει deploy η εφαρμογή είναι η <http://localhost:8080/serverapp/> τότε η URL διεύθυνση που θα σχηματιστεί με την αποστολή της παραπάνω φόρμας θα είναι:
- <http://localhost:8080/serverapp/add-student>
- Τα δεδομένα με την μέθοδο post δεν αποστέλλονται στο URL αλλά μέσα στο HTTP Request, στα data ως payload



DELETE

Java EE

```
<form action="{pageContext.request.contextPath}/  
del-student" method="DELETE">  
    Ssn: <input type="text" name="ssn"><br>  
    <button type="submit" value="Submit">  
</form>
```

- Αν υποθέσουμε ότι η URL διεύθυνση στην οποία έχει γίνει deploy η εφαρμογή είναι η <http://localhost:8080/serverapp/> τότε η URL διεύθυνση που θα σχηματιστεί με την αποστολή της παραπάνω φόρμας θα είναι:
- <http://localhost:8080/serverapp/del-student>



```
<form action="$ {pageContext.request.contextPath}/update-
student" method="PUT">
    Ssn: <input type="text" name="ssn"><br>
    <button type="submit" value="Submit">
</form>
```

- Αν υποθέσουμε ότι η URL διεύθυνση στην οποία έχει γίνει deploy η εφαρμογή είναι η <http://localhost:8080/serverapp/> τότε η URL διεύθυνση που θα σχηματιστεί με την αποστολή της παραπάνω φόρμας θα είναι:
- <http://localhost:8080/serverapp/update-student>