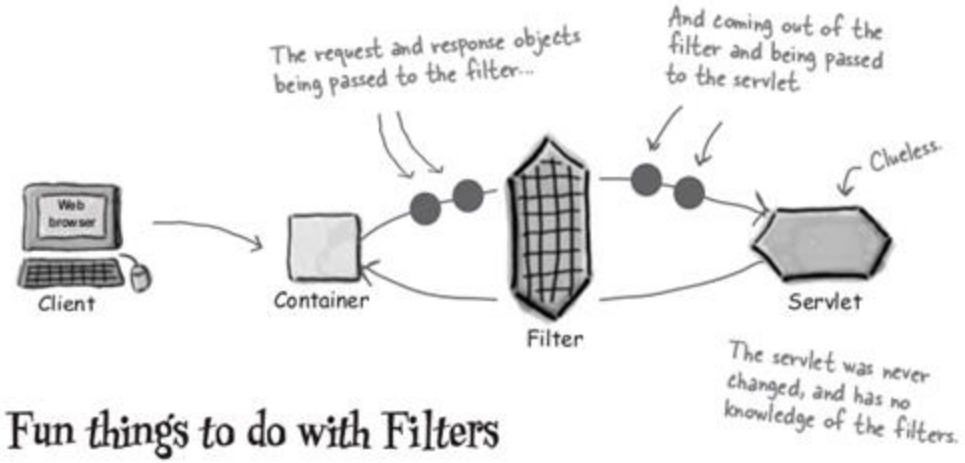
Филтри

Веб програмирање

Што се филтри

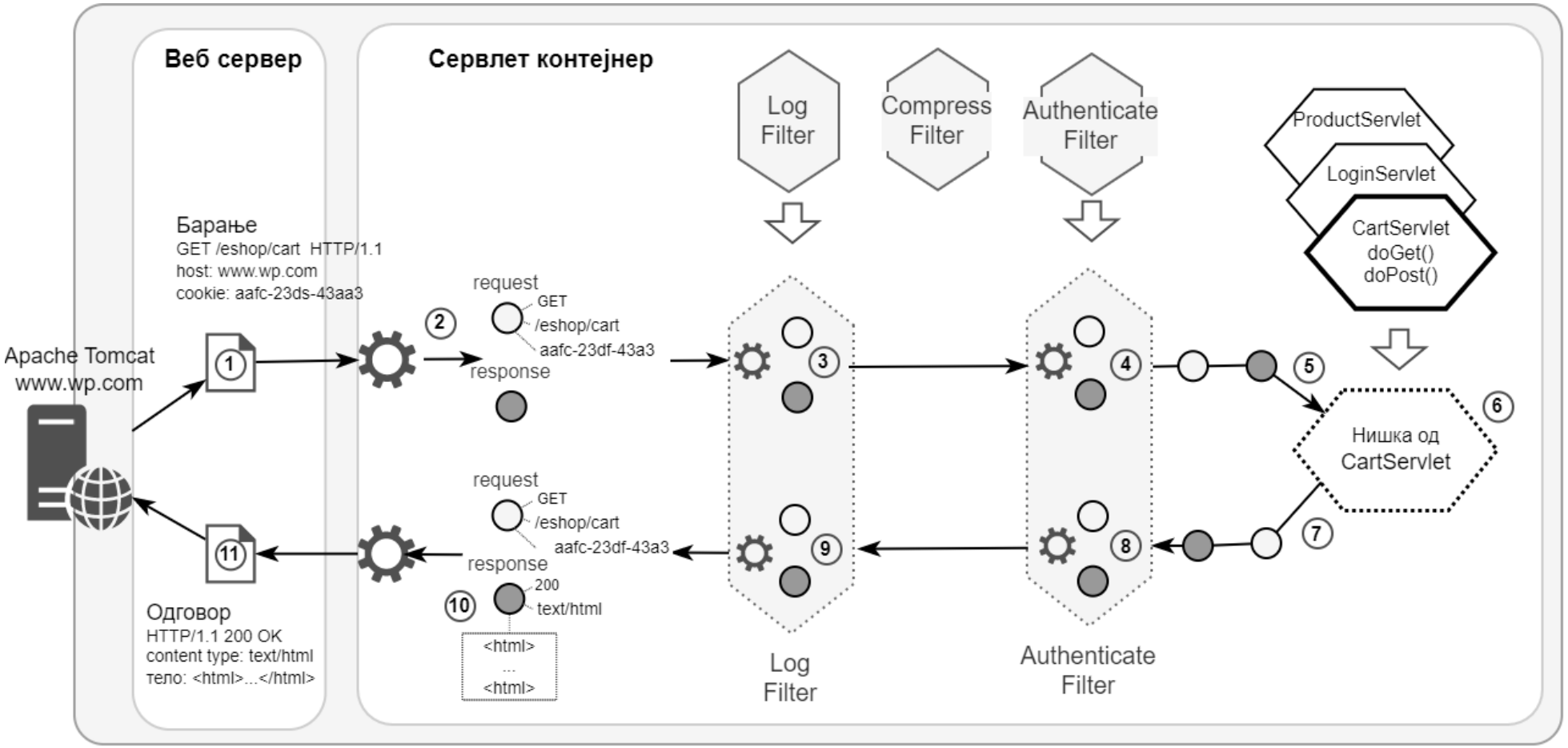
- Модуларни компоненти управувани од контејнерот
- Се вклучуваат со конфигурација
  - Слично како сервлетите
  - Независни од сервлетите



Зошто филтрите се од токлу голема важност во пракса?

- Проверка на привилегии
  - Во секој сервлет посебно!!!
- Логирање на параметри и заглавја
  - Во секој сервлет посебно!!!
- Компресирање на одговорот
  - Во секој сервлет посебно!!!
- Генерирање на страна за недозволен пристап
  - Во секој сервлет посебно!!!
- Или посебен **филтер** за секоја **намена** ☺

Претпроцесирање и постпроцесирање на барање и одговор со користење на филтри



Конфигурација на филтри и нивно пресликување во web.xml

- <filter-mapping>
  - <filter-name>
    - со името на претходно дефиниран филтер и
  - <url-pattern>
    - со URL адресата на барањето кое ќе го активира, или
  - <servlet-name>
    - со името на сервлетот за кој е наменето барањето. Овој елемент може да содржи листа од повеќе сервлети одвоени со запирка.
  - <dispatcher>
    - REQUEST
      - за барања директно од клиентот
    - FORWARD
      - за барања кои потекнуваат од методот forward()
    - INCLUDE
      - за барања кои потекнуваат од методот include() или
    - ERROR
      - за барања кои се проследуваат до конфигурирана страница за
      - Грешки (ако таква страна не е конфигурирана во web.xml, филтерот не се активира).

```
<web-app>
    <filter>
        <filter-name>timer</filter-name>
        <filter-class>mk.ukim.finki.TimerFilter</filter-class>
        <init-param>
            <param-name>timeout</param-name>
            <param-value>2000</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>timer</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
        <dispatcher>FORWARD</dispatcher>
    </filter-mapping>
<web-app>
```

Конфигурација на филтри со анотација

```
<web-app>
    <filter>
        <filter-name>timer</filter-name>
        <filter-class>mk.ukim.finki.TimerFilter</filter-
        class>
        <init-param>
            <param-name>timeout</param-name>
            <param-value>2000</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>timer</filter-name>
        <url-pattern>/*</url-pattern>
        <dispatcher>REQUEST</dispatcher>
        <dispatcher>FORWARD</dispatcher>
    </filter-mapping>
<web-app>
```

```
@WebFilter(
   filterName = "timer",
   urlPatterns = "/*",
   dispatcherTypes = {
      DispatcherType.REQUEST, DispatcherType.FORWARD
   },
    initParams = {
      @WebInitParam(name = "timeout", value = "2000“)
   }
)
public class TimerFilter extends Filter {
...
}
```

Подредување на филтри при извршување

- Прво се идентификуваат филтрите кои се совпаѓаат според **url-pattern** и се извршуваат според редоследот на дефинирање во web.xml
  - Може да бидат селектирани повеќе филтри кои ќе се извршат за едно барање
- Потоа се идентификуваат филтрите со **servlet-name** за селектираниот сервлет, според редоследот на дефинирање во web.xml
  - Овие филтри ќе се извршат по филтрите кои дефинираат url-pattern, дури и ако се дефинирани пред нив.
- Не можеме да го дефинираме редоследот на филтрите при конфигурација со анотации
- Кај Spring Boot апликации, можеме да ги поредиме филтрите доколку ги конфигурираме со **FilterRegistrationBean**

Подредување на филтри при извршување

```
<filter-mapping>
  <filter-name>Filter1</filter-name>
  <url-pattern>/Recipes/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>Filter2</filter-name>
  <servlet-name>/Recipes/HopsList.do</servlet-name>
</filter-mapping>

<filter-mapping>
  <filter-name>Filter3</filter-name>
  <url-pattern>/Recipes/Add/*</url-pattern>
</filter-mapping>

<filter-mapping>
  <filter-name>Filter4</filter-name>
  <servlet-name>/Recipes/Modify/ModRecipes.do</servlet-name>
</filter-mapping>

<filter-mapping>
  <filter-name>Filter5</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Request path	Filter Sequence
/Recipes/HopsReport.do	Filters: 1, 5
/Recipes/HopsList.do	Filters: 1, 5, 2
/Recipes/Modify/ModRecipes.do	Filters: 1, 5, 4
/HopsList.do	Filters: 5
/Recipes/Add/AddRecipes.do	Filters: 1, 3, 5

Имплементација на филтри

```
public class TimerFilter extends Filter {
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
                throws ServletException {
            //Preprocessing: the code above is executed before the objects are passed to the next filter in the chain in direction of activation towards the servlet
            chain.doFilter(request, response);
            //Postprocessing: the code below is executed when the objects are passed back in direction from the servlet towards the first filter
    }
}
```

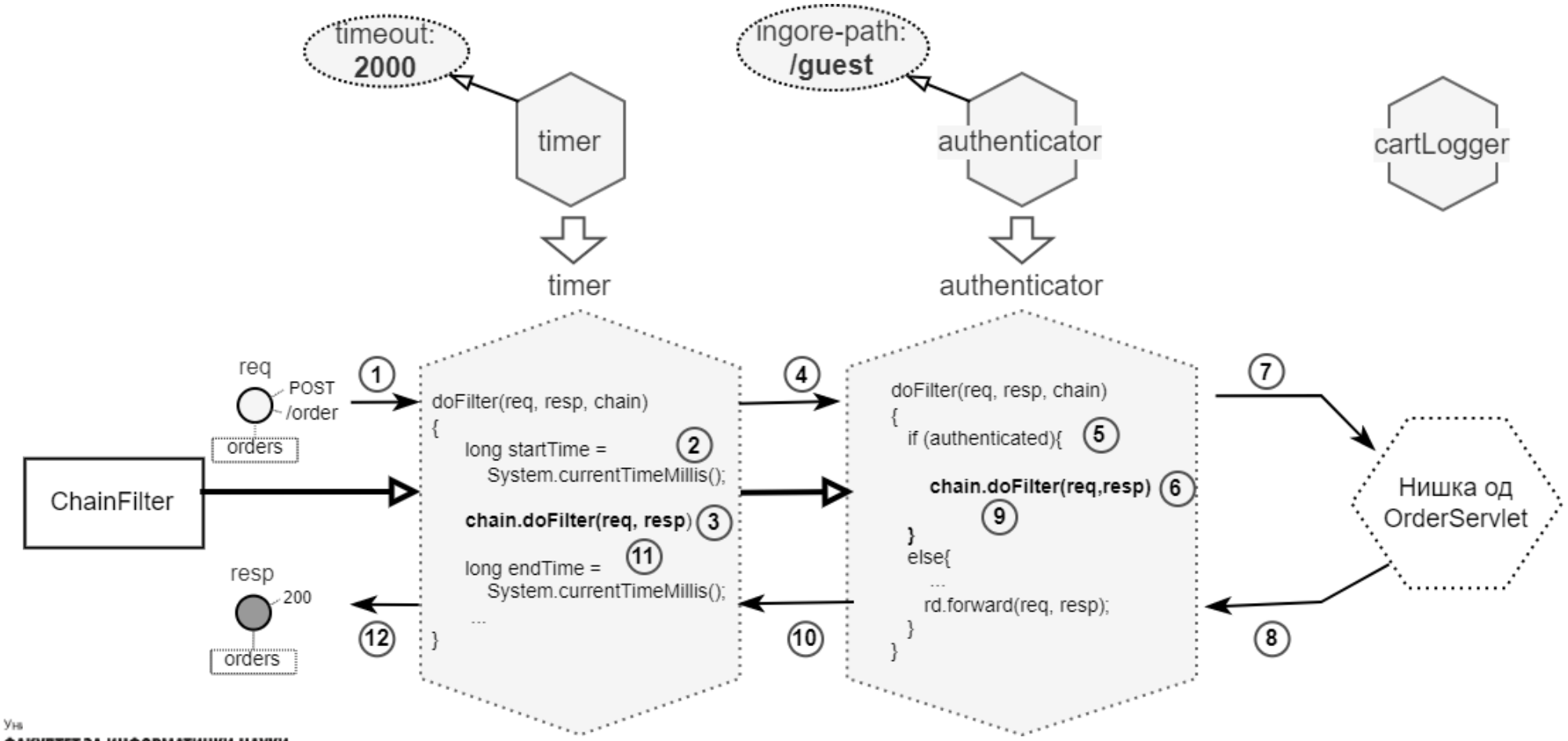
Пример 1. Имплементација на филтер за мерење на време на комплетна обработка на барање

```
1  public class TimerFilter implements Filter {
2    private ServletContext context;
3    private long timeout;
4    public void init(FilterConfig config) throws ServletException {
5      this.context = config.getServletContext();
6      this.timeout = (long)config.getParametar("timeout");
7    }
8    public void destroy() {}
9    public void doFilter(ServletRequest request, ServletResponse response,
         FilterChain chain) throws ServletException {
10     long startTime = System.currentTimeMillis();
11     chain.doFilter(request, response);
12     long endTime = System.currentTimeMillis();
13     long total endTime - startTime;
14     HttpServletRequest req = (HttpServletRequest) request;
15     String name = req.getRequestURI();
16     String comment = total > this.timeout ? " (too long!)": "";
17     this.context.log(name + " took " + total + " ms" + comment);
18   }
19 }
```

Пример 2. Имплементација на филтер за автентикација

```
1  public class AuthenticationFilter implements Filter {
2    privte String ignorePath;
3    public void init(FilterConfig config) throws ServletException {
4       this.ignorePath = config.getParametar("ignore-path");
5    }
6    public void destroy() {}
7    public void doFilter(ServletRequest request, ServletResponse response,
         FilterChain chain) throws ServletException {
8      HttpServletRequest req = (HttpServletRequest) request;
9      if (req.getRequestURI().startsWith(ingorePath) || req.getUserPrincipal()
           != null)
10         chain.doFilter(req,resp)
11     else{
12         RequestDispatcher rd=req.getRequestDispatcher("login.html");
13         rd.forward(req, resp);
14     }
15   }
16 }
```

Редослед на извршување на код од филтри при обработка на барање



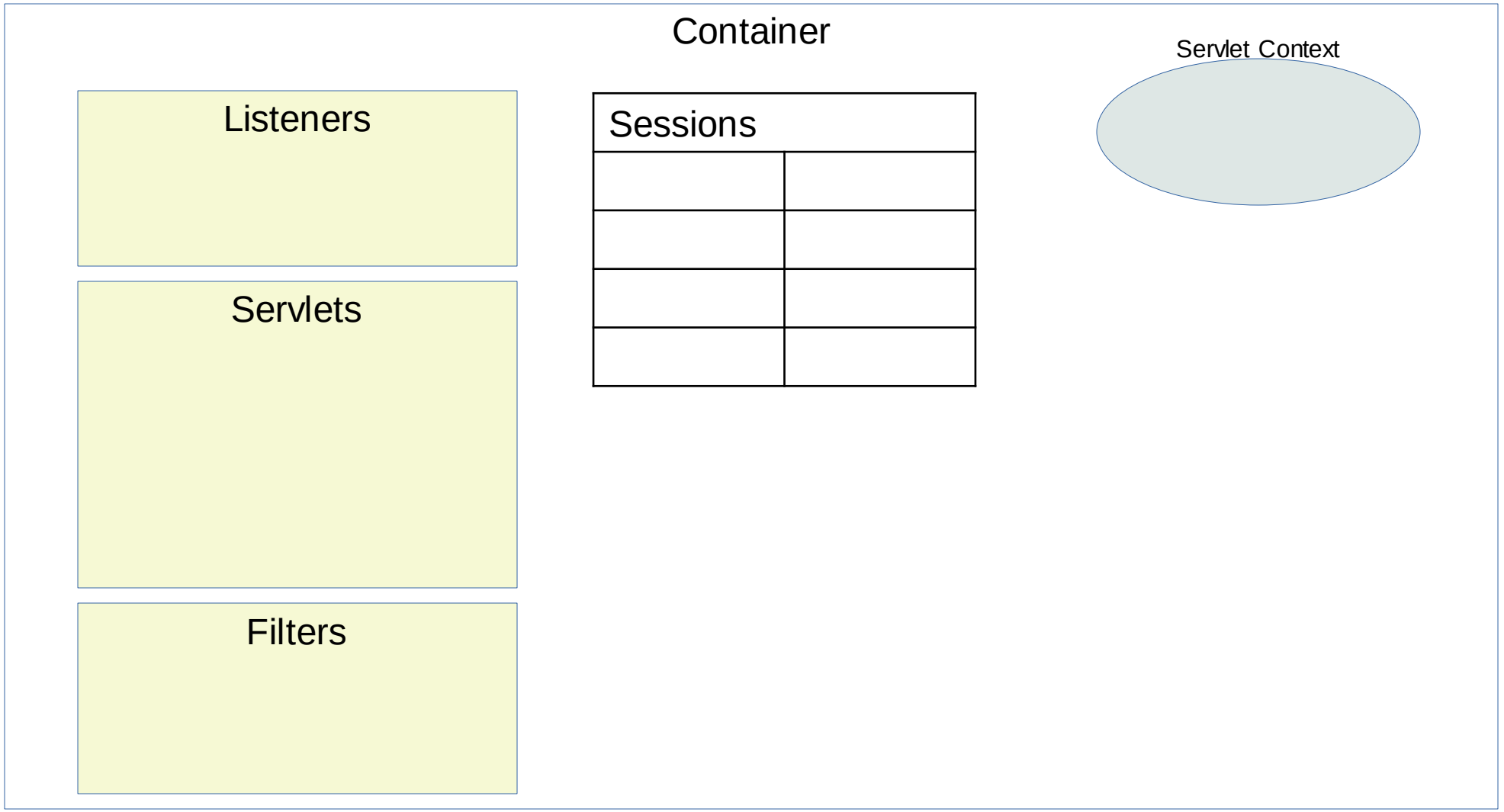
Преглед на JEE

Веб програмирање

Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Што чува JEE контејнерот во меморија

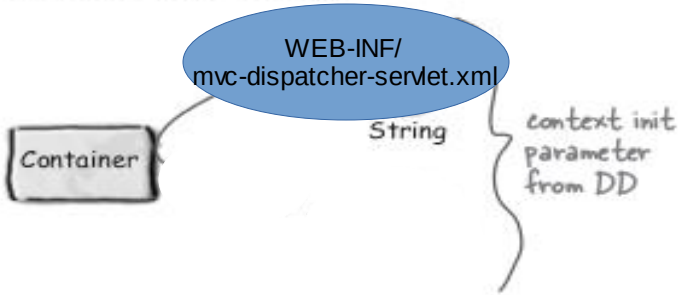
Иницијална состојба



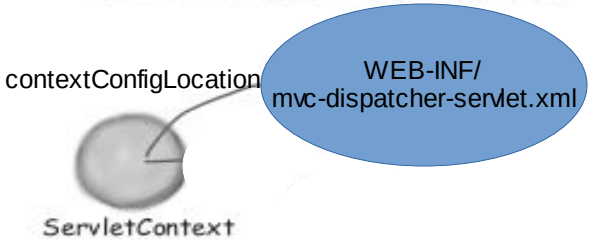
1. Container reads the Deployment Descriptor for this app, including the <listener> and <context-param> elements.

2. Container creates a new ServletContext for this application, that all parts of the app will share.

3. Container creates a name/value pair of Strings for each context init parameter. Assume we have only one.

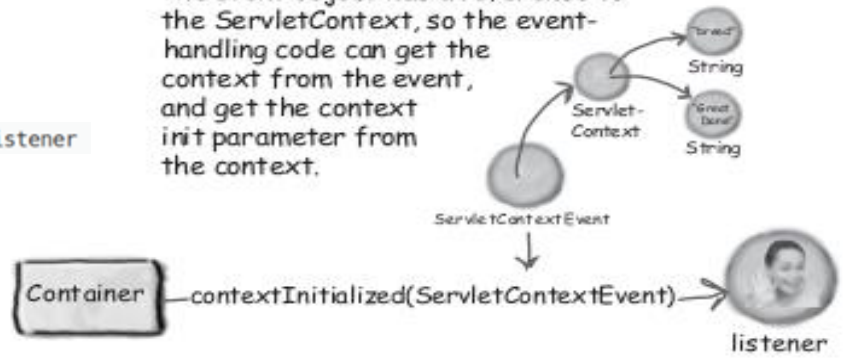


4. Container gives the ServletContext references to the name/value parameters.



5. Container creates a new instance of the `org.springframework.web.context.ContextLoaderListener`

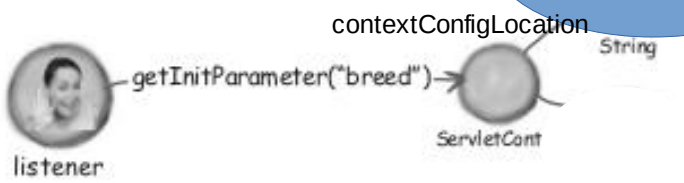
6. Container calls the listener's contextInitialized() method, passing in a new ServletContextEvent. The event object has a reference to the ServletContext, so the event-handling code can get the context from the event, and get the context init parameter from the context.



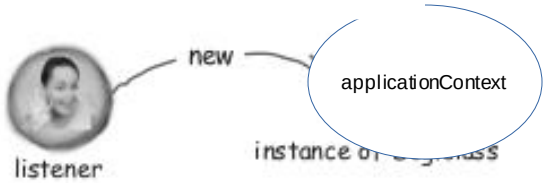

```xml
<web-app>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/mvc-dispatcher-servlet.xml</param-value>
    </context-param>

    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>

    <servlet>
        <servlet-name>comingsoon</servlet-name>
        <servlet-class>mysite.server.ComingSoonServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>comingsoon</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>redteam</servlet-name>
        <servlet-class>mysite.server.TeamServlet</servlet-class>
        <init-param>
            <param-name>teamColor</param-name>
            <param-value>red</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>redteam</servlet-name>
        <url-pattern>/red/*</url-pattern>
    </servlet-mapping>

    <filter>
        <filter-name>logSpecial</filter-name>
        <filter-class>mysite.server.LogFilterImpl</filter-class>
        <init-param>
            <param-name>logType</param-name>
            <param-value>special</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>logSpecial</filter-name>
        <servlet-name>comingsoon</servlet-name>
        <!-- <url-pattern>*.special</url-pattern> -->
    </filter-mapping>
</web-app>
```

7 Listener asks ServletContextEvent for a reference to the ServletContext.

8 Listener asks ServletContext for the context init parameter '

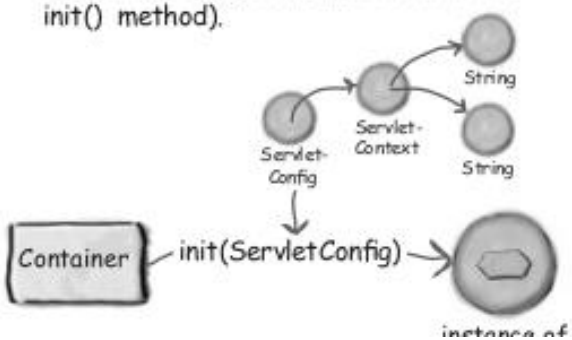


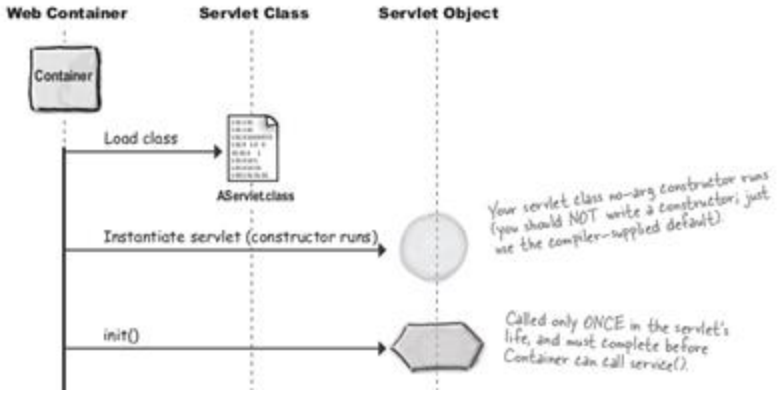
9 Listener uses the init parameter to construct a new



10 Listener sets attribute in the ServletContext.

11 Container makes a new Servlet (i.e., makes a new ServletConfig with init parameters, gives the ServletConfig a reference to the ServletContext, then calls the Servlet's init() method).

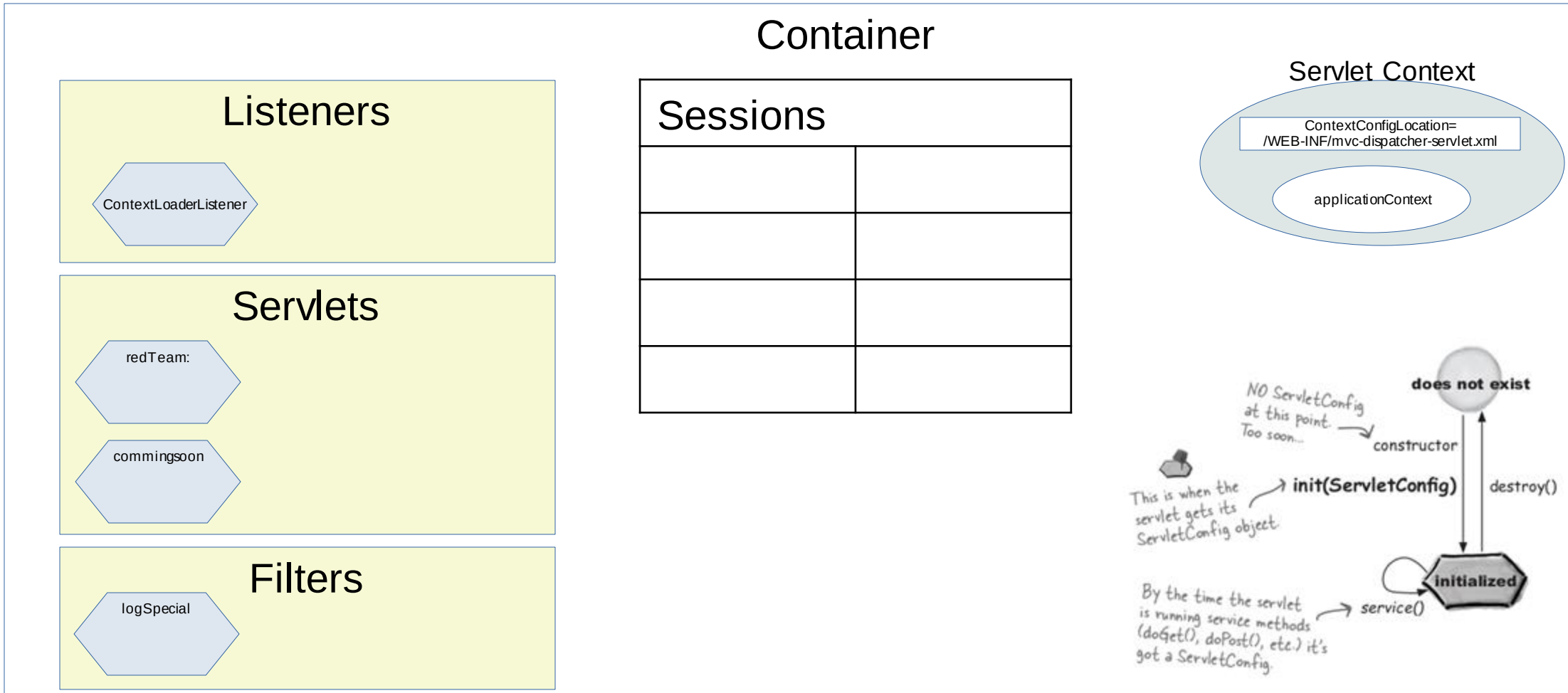




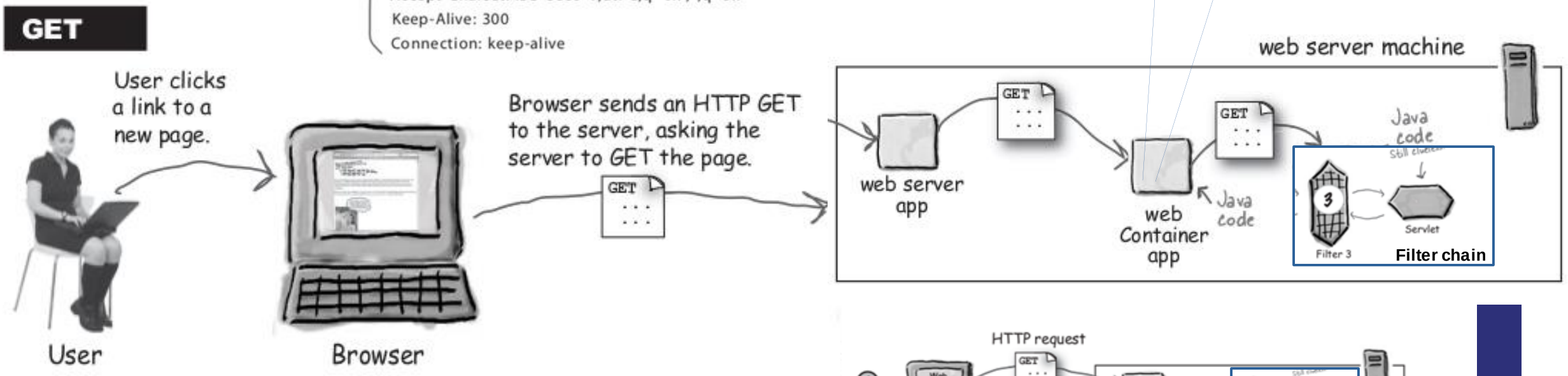
WEB-INF/
mvc-dispatcher-servlet.xml

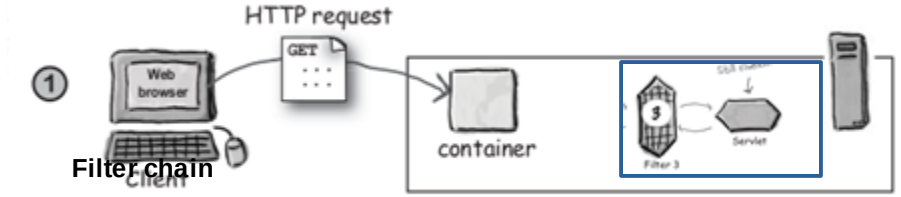
```xml
<web-app>
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/mvc-dispatcher-servlet.xml</param-value>
    </context-param>

    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>

    <servlet>
        <servlet-name>comingsoon</servlet-name>
        <servlet-class>mysite.server.ComingSoonServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>comingsoon</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>redteam</servlet-name>
        <servlet-class>mysite.server.TeamServlet</servlet-class>
        <init-param>
            <param-name>teamColor</param-name>
            <param-value>red</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>redteam</servlet-name>
        <url-pattern>/red/*</url-pattern>
    </servlet-mapping>

    <filter>
        <filter-name>logSpecial</filter-name>
        <filter-class>mysite.server.LogFilterImpl</filter-class>
        <init-param>
            <param-name>logType</param-name>
            <param-value>special</param-value>
        </init-param>
    </filter>

    <filter-mapping>
        <filter-name>logSpecial</filter-name>
        <servlet-name>comingsoon</servlet-name>
        <!-- <url-pattern>*.special</url-pattern> -->
    </filter-mapping>
</web-app>
```

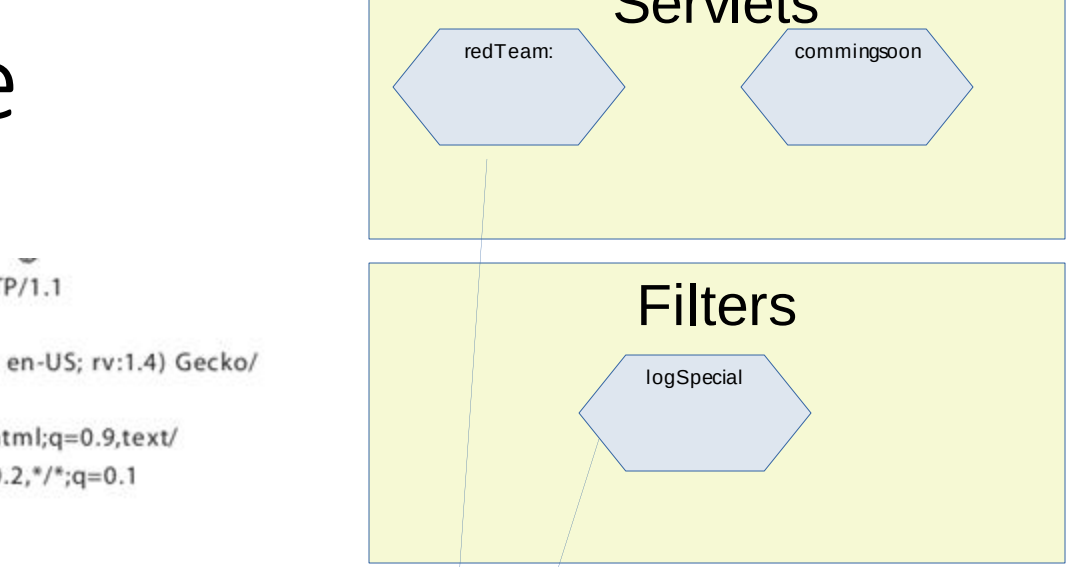
Состојба на JEE контејнерот по стартување



Процесирање на барање

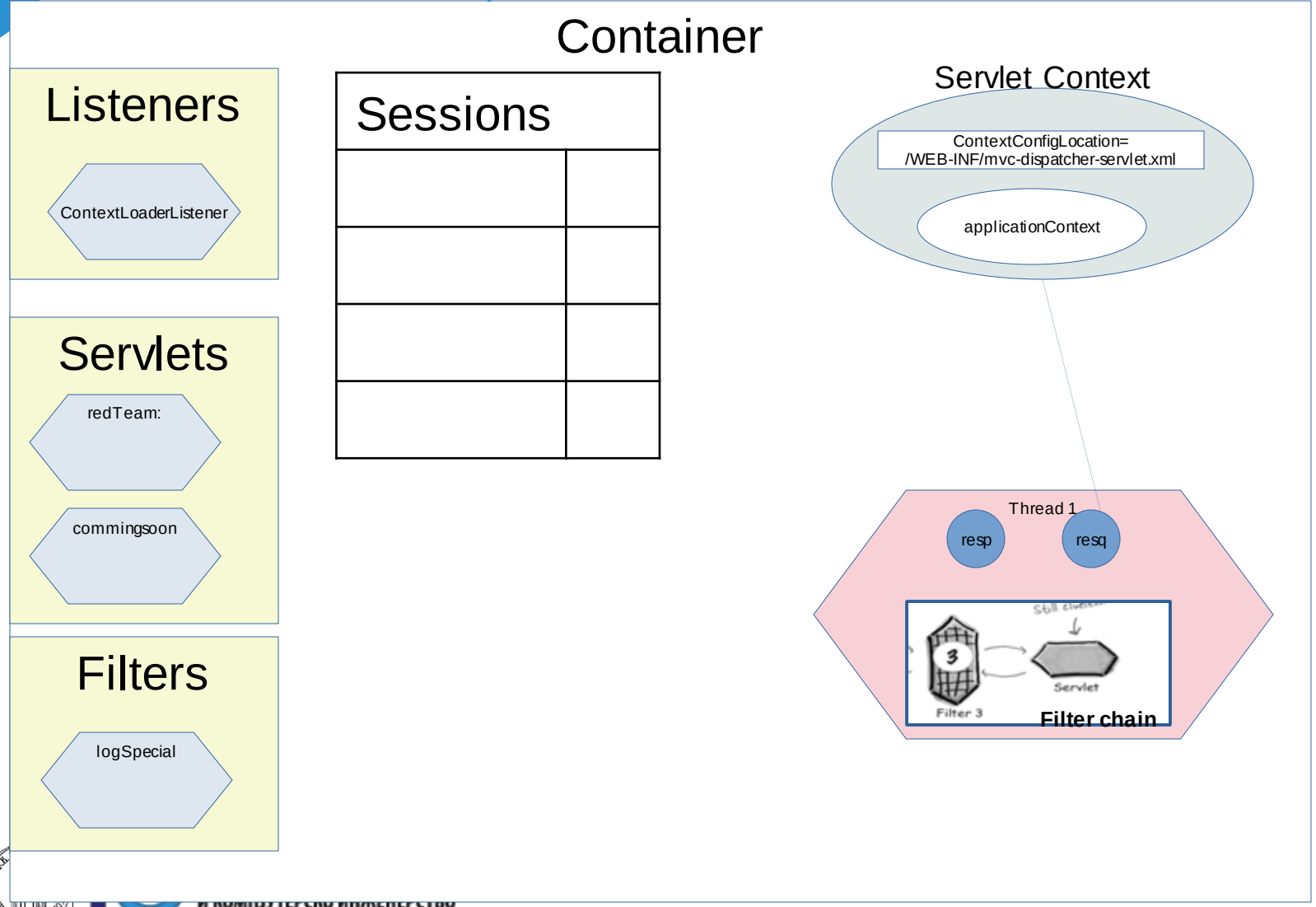


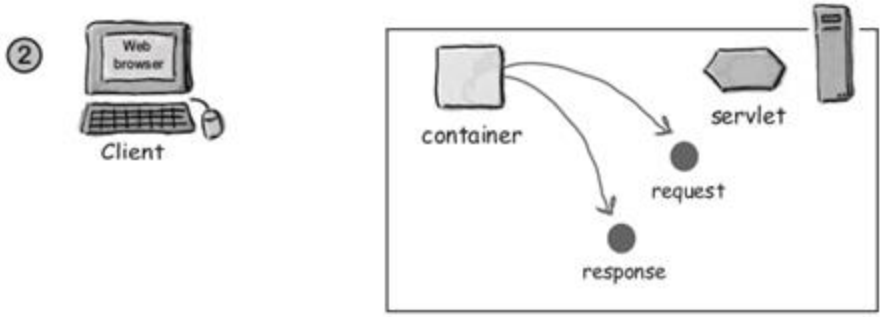


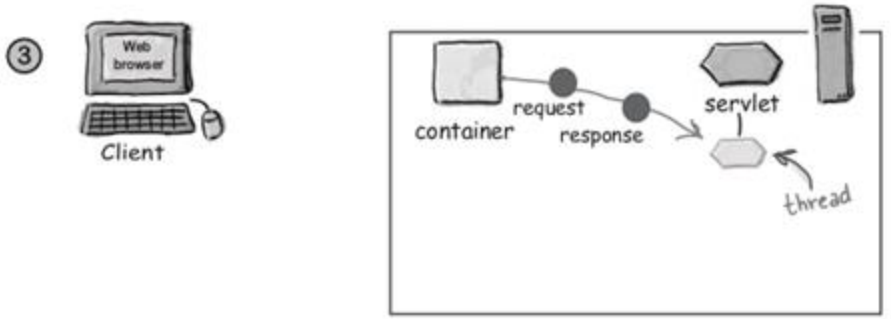


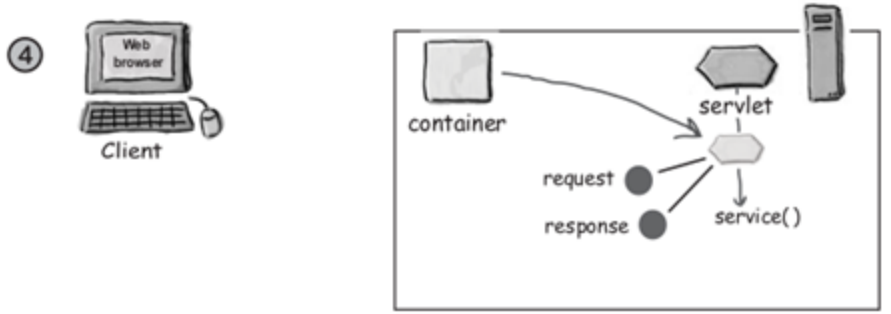
Извршување на барање

Повик на сервлет



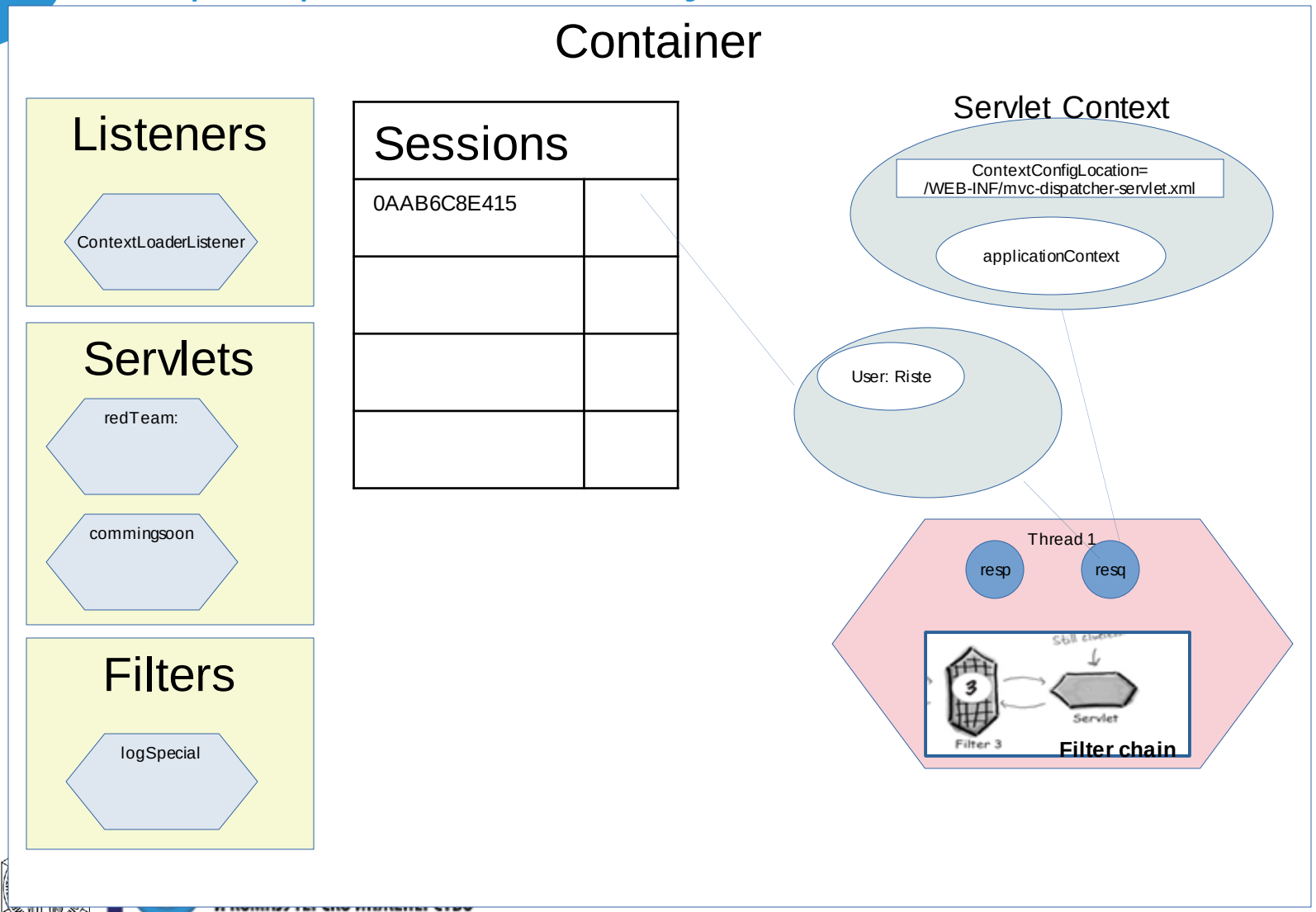


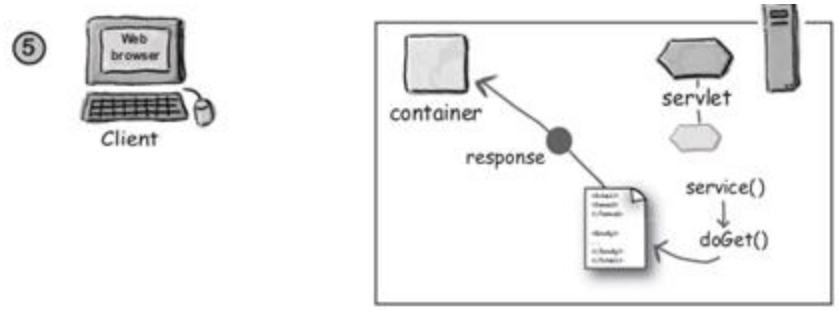


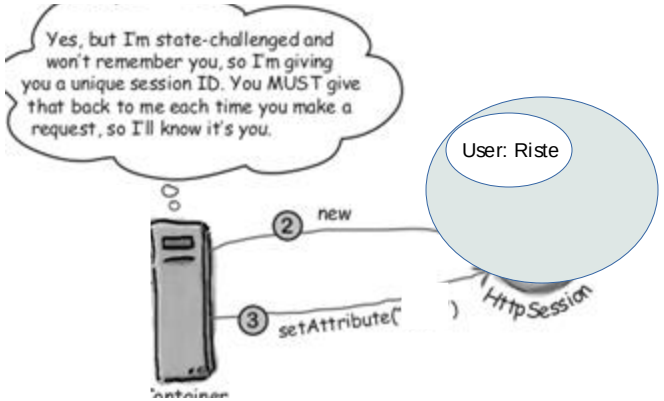


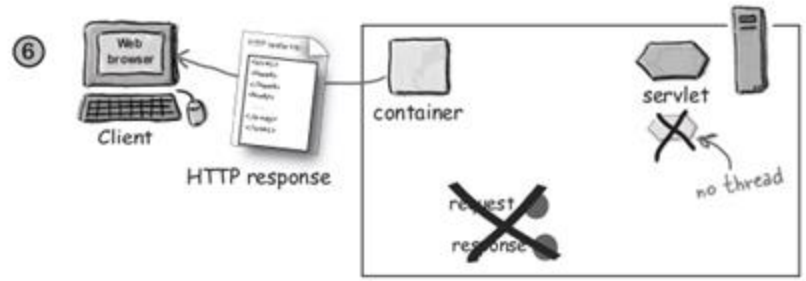
Извршување на барање

Креирање на сесија



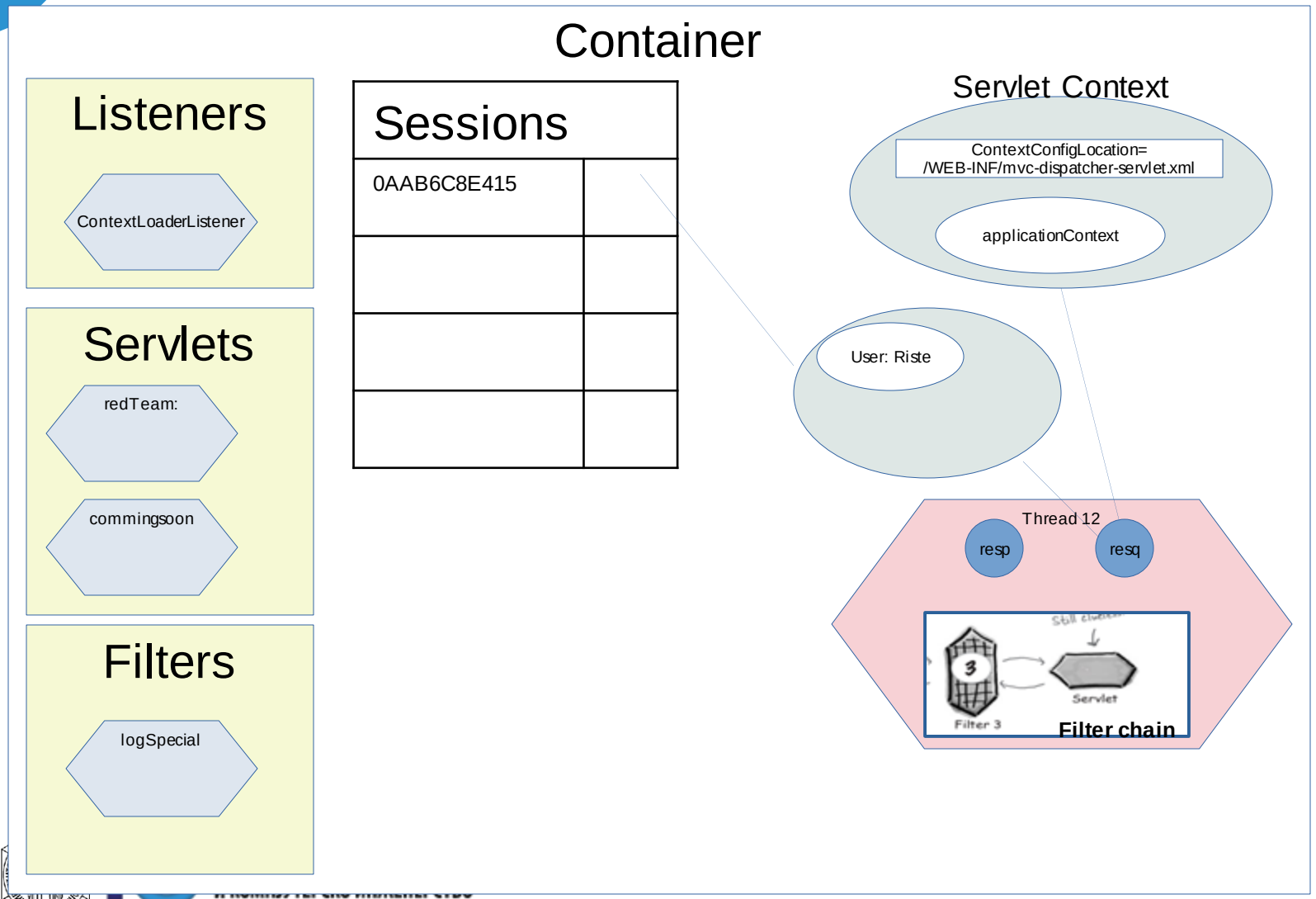


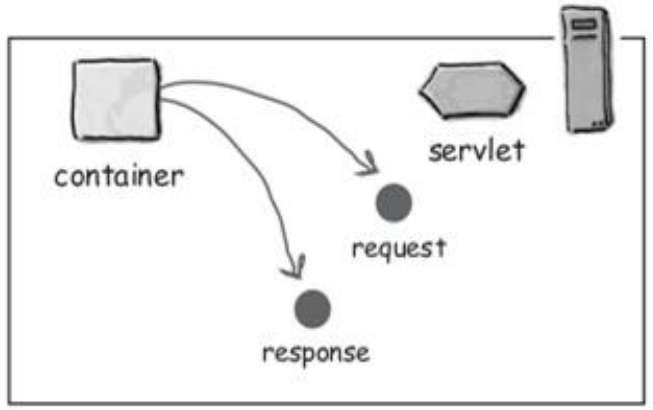


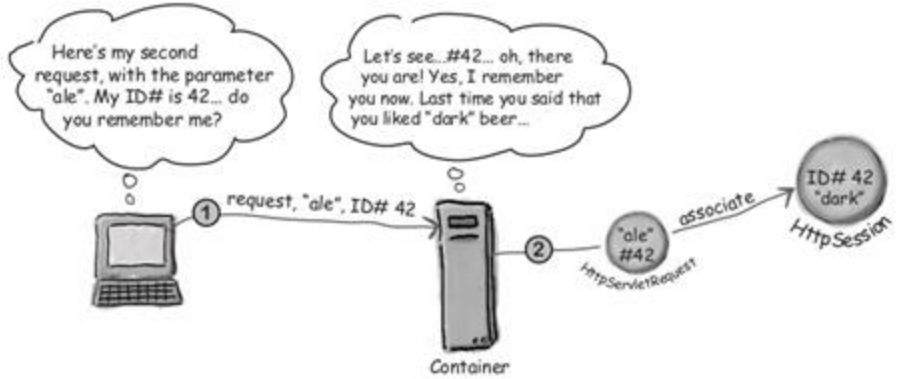


Извршување на барање

Повик на сервлет со детекција на сесија







Состојба на JEE контејнер по повеќе барања