Основи на Java EE втор дел

Веб програмирање

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

Управување со состојба

- Потреба од зачувување на привремени и постојани податоци
- Параметри
  - Read only
  - Текстуален тип
- Атрибути
  - Read/write
  - Object (потребен е cast)
- Области на важење
  - Сервлет
  - Контекст
  - Барање
  - Сесија

Управување со состојба

- Преглед на карактеристики на параметри и атрибути

	Атрибут	Параметар
Опсег на важност	ServletContext Request Session	ServletContext ServletConfig Request
Тип на податок	Object	String
Метод за читање	getAttribute(String ime)	getInitParameter(String ime)
Метод за поставување	setAttribute(String ime, Object vrednost)	Не може да се постави во код, туку во web.xml или преку анотации

Работа со податоци во апликациски опсег

- При иницијализација на апликацијата, контејнерот креира инстанца од `ServletContext`
  - Објект кој содржи референци кон апликациски параметри и атрибути
  - Централен објект од големо значење преку кој може да комуницираат компонентите на апликацијата
- Пристап до апликациски контекст

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    public void init()  {
        ServletContext context1 = getServletContext();
        ServletConfig config = getServletConfig();
        ServletContext context2 = cofnig.getServletContext();
    }
}
```

Дефиниција и пристап до контекстни параметри

- Дефиниција во web.xml

```
<web-app>
    ...
    <context-param>
        <param-name>admin-user</param-name>
        <param-value>administrator</param-value>
    </context-param>
    <context-param>
        <param-name>admin-password</param-name>
        <param-value>5pr1ng</param-value>
    </context-param>
    ...
</web-app>
```

- Пристап преку методи на ServletContext

```
public String getInitParameter(String name)
public Enumeration getInitParameterNames()
```

Управување со контекстни атрибути

- Методи од ServletContext

```
public void setAttribute(String name, Object object)
public Object getAttribute(String name)
public void removeAttribute(String name):
```

- Пример

```
@WebServlet("/cart")
public class ShoppingCartServlet extends HttpServlet {
    public void init() throws ServletException {
        super.init();
        Integer visits = 0;
        getServletContext().setAttribute("cartVisits", visits);
    }
    public void doGet(HttpServletRequest request, HttpServletResponse
     ↪  response) throws IOException {
        ServletContext context = getServletContext();
        Integer visits = (Integer) context.getAttribute("cartVisits");
        getServletContext().setAttribute("cartVisits", ++visits);
    }
    public void destroy() {
        super.destroy();
        getServletContext().removeAttribute("cartVisits");
    }
}
```

Контекстни настани и обсервери

- При промена на контекст (креирање и уништување), контејнерот исфрла настан од типот **`ServletContextEvent`**
  - Настанот пренесува аргумент од типот `ServletContext`

```
ServletContextEvent(ServletContext e)
```

- Настаните поврзани со промена на контекстот се обработуваат во класа - обсервер која го имплементира интерфејсот **`ServletContextListener`**
  - Методи на интерфејс

```
public void contextInitialized(ServletContextEvent e)
public void contextDestroyed(ServletContextEvent e)
```

Контекстни настани и обсервери

- Пример

```
public class EshopListener implements ServletContextLlistener {
   public void contextInitialized(ServletContextEvent e)
        // initialize connection to database
        // get number of saved visits from database and store it in visits
        ServletContext context = e.getServletContext();
        context.setAttribute("cartVisits", visits);
    }
    public void contextDestroyed(ServletContextEvent e) {
        ServletContext context = getServletContext();
        Integer visits = (Integer) context.getAttribute("cartVisits");
        //store visits in database
        //close connection to database
    }
}
```

Контекстни настани и обсервери

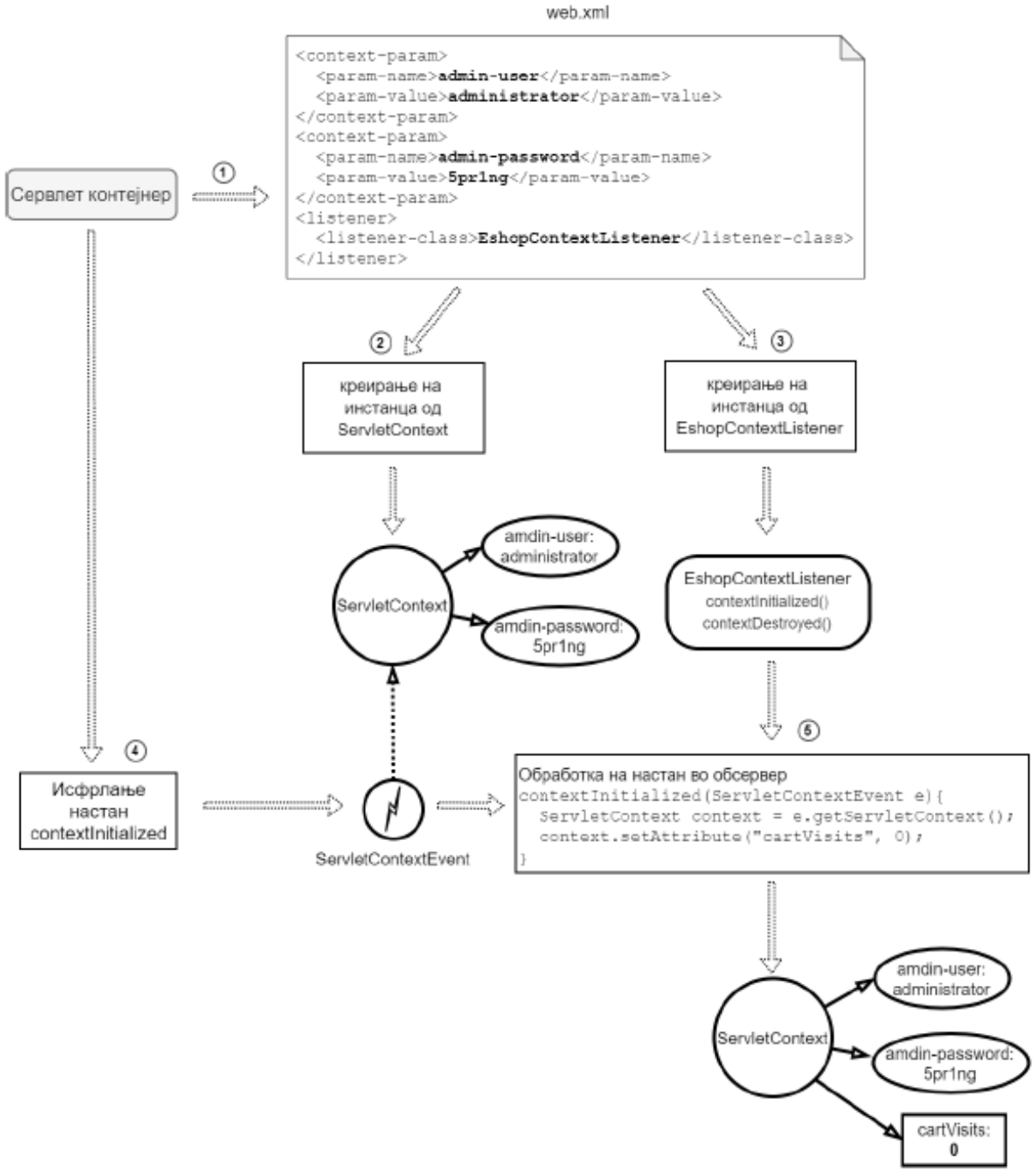
- Регистрација на обсервер
  - web.xml

```
<web-app>
    ...
    <listener>
        <listener-class>EshopListener</listener-class>
    </listener>
    ...
</web-app>
```

  - Анотација

```
@WebListener
```

Контекстни настани и обсервери - преглед



Настани и обсервери на контекстни атрибути

- При промена на контекстни атрибути (креирање, промена и бришење) се исфрла настан `ServletContextAttributeEvent`
  - Настанот пренесува контекст, име и вредност на атрибутот
  - Пристап до име и вредност преку методите

```
public String getName()
public Object getValue()
```

- Обсервер за обработка на настани поврзани со контекстен атрибут имплементира интерфејс `SerlvetContextAttributeListener` со методи

```
public void attributeAdded(ServletContextAttributeEvent e)
public void attributeRemoved(ServletContextAttributeEvent e)
public void attributeReplaced(ServletContextAttributeEvent e)
```

Работа со податоци во сервлетски опсег

- При инстанцирање на сервлет, контејнерот креира инстанца од `ServletConfig`
  - Објект кој содржи референци кон параметри дефинирани за сервлетот (web.config или анотации)
  - Се пренесува како аргумент на init() или се добива со повик на

```
public ServletConfig  getServletConfig();
```

- Методи на `ServletConfig`

```
public String getInitParameter(String name)
```

```
public Enumeration getInitParameterNames()
```

Параметри на сервлети

- Преку web.xml

```
<servlet>
    <servlet-name>login</servlet-name>
    <servlet-class>mk.ukim.finki.Login</servlet-class>
    <init-param>
        <param-name>login-attempts</param-name>
        <param-value>3</param-value>
    </init-param>
    <init-param>
        <param-name>lock-account</param-name>
        <param-value>true</param-value>
    </init-param>
</servlet>
```

- Преку анотација

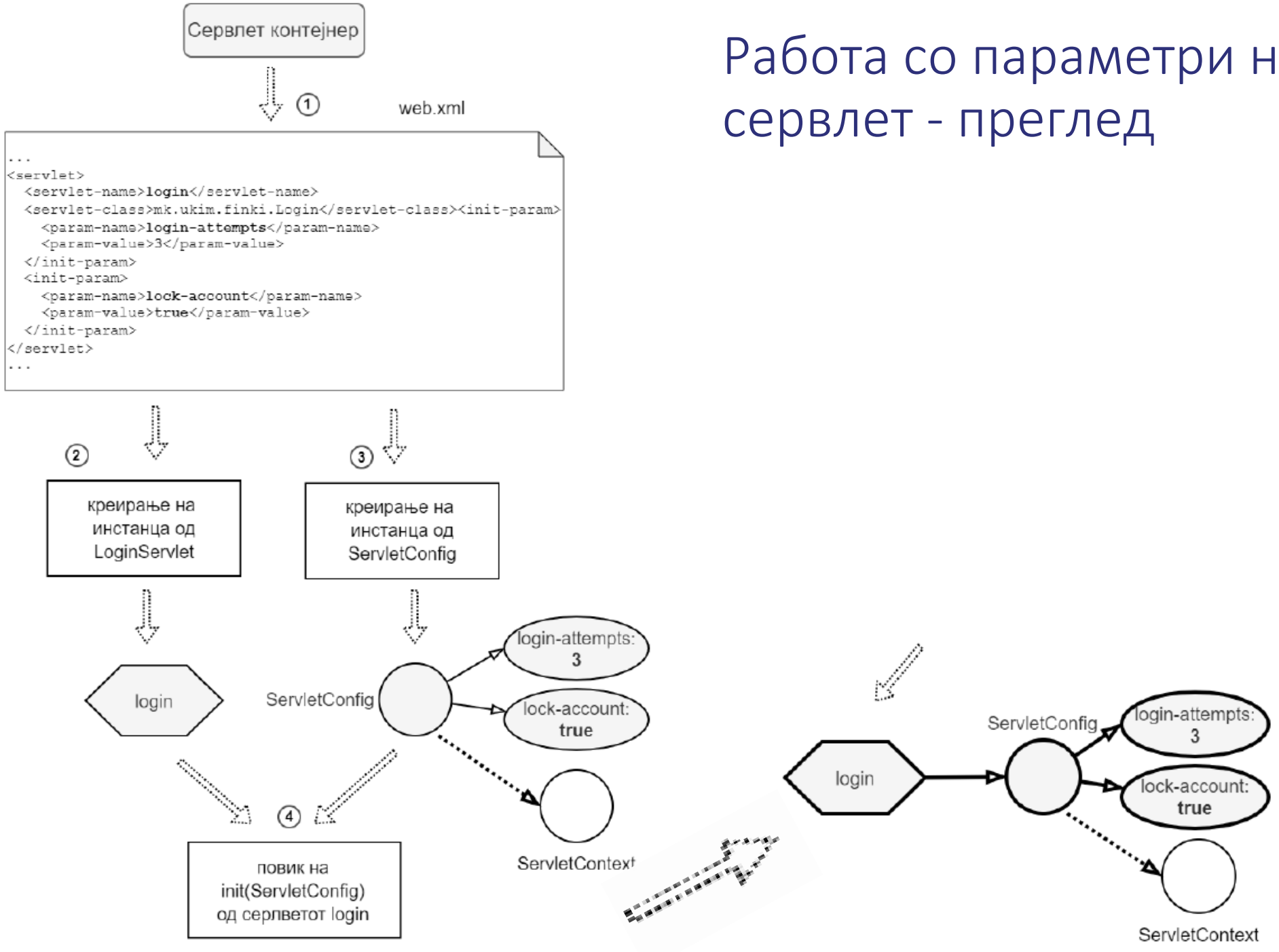
```
@WebServlet(name = "login", value = "/login", initParams = {
                @WebInitParam(name = "login-attempts", value = "3"),
                @WebInitParam(name = "lock-account", value = true)}
        )
public class Login extends HttpServlet {
```

Работа со параметри на сервлет

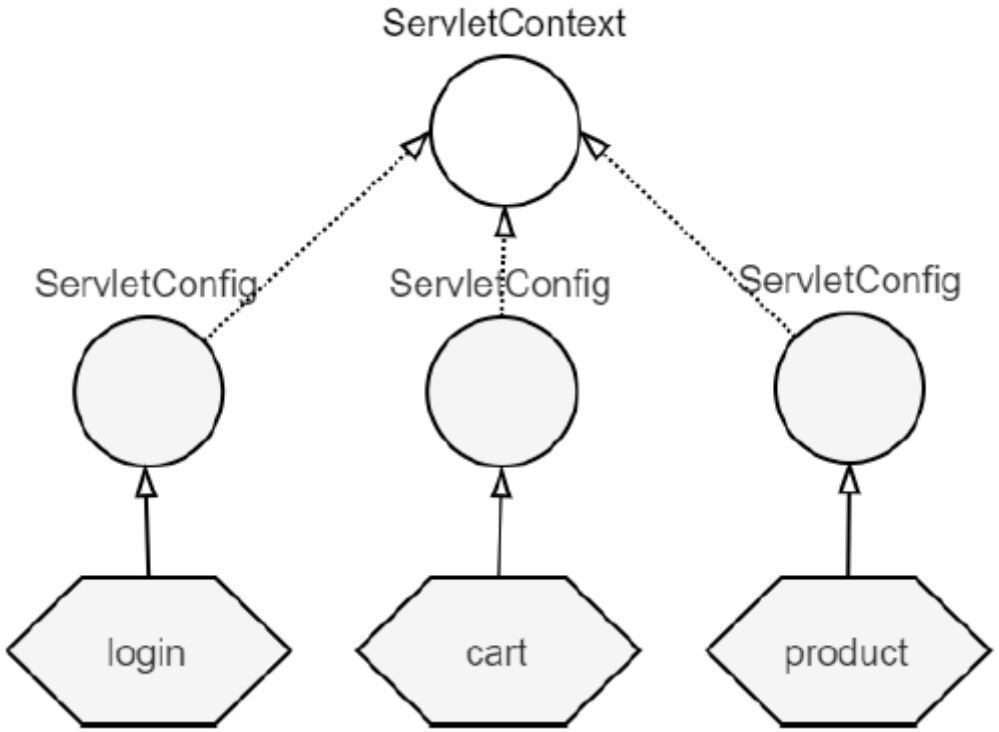
- Пример

```
@WebServlet(name = "login", value = "/login")
public class LoginServlet extends HttpServlet {
    public void init()  {
        ServletConfig config = getServletConfig();
        String loginAttempts = config.getInitParameter("login-attempts");
        System.out.println("The current value of login-attempts is " +
         ↪  loginAttempts);
        Enumeration<String> configParameters =
         ↪  config.getInitParameterNames();
        while (configParameters.hasMoreElements()){
            String paramName = configParameters.nextElement();
            System.out.println("param-name: " + paramName + "  param-value:
             ↪  " + config.getInitParameter(paramName) );
        }
    }
}
```

Работа со параметри на сервлет - преглед



Врска помеѓу ServletContext и ServletConfig



Работа со податоци во опсег на барање

- За секое барање, контејнерот креира инстанца од `HttpServletRequest`
  - Содржи референци кон параметри и атрибути на барање
  - Методи за читање на параметри

```
public String getParameter(String name)
public Enumeration getParameterNames()
```

  - Методи за читање и уредување на атрибути
    - Атрибутите имаат животен век во рамките на истото барање
    - Постојат додека постои и барањето

```
public void setAttribute(String name, Object object)
public Object getAttribute(String name)
public void removeAttribute(String name):
```

Работа со податоци во опсег на барање

- Пример

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
     ↪  response) throws IOException, ServletException {
        String password = request.getParameter("password");
        if (password.equals("spring")){
            Random random = new Random();
            request.setAttribute("luckyUser", random.nextBoolean())
            RequestDispatcher dispatcher =
             ↪  request.getRequestDispatcher("home");
            dispatcher.forward(request,response);
        }
    }
}
```

Работа со податоци во опсег на барање

- Пример

```
@WebServlet("/home")
public class HomeServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
     ↪  response) throws IOException, ServletException {
       response.setContentType("text/html");
       PrintWriter out = response.getWriter();
       out.println("Welcome " + request.getParameter("username");)
       Boolean luckyUser = (Boolean) request.getAttribute("luckyUser")
       if (luckyUser)
         out.println("You are winner of a free coupon! ");
    }
}
```

Настани и обсервери на барање

- При промена на барање (креирање и бришење), контејнерот исфрла настан од типот `ServletRequestEvent`
  - Содржи референца до барањето
  - Барањето содржи референца до контекстот
  - Пристап преку методите

```
public ServletContext getServletContext()
public ServletRequest getServletRequest()
```

- Настаните се обработуваат преку обсервер кој имплементира интерфејс `ServletRequestListener` со методи

```
public void requestInitialized(SerlvetRequestEvent e)
public void requestDestroyed(SerlvetRequestEvent e)
```

Настани и обсервери на атрибути на барање

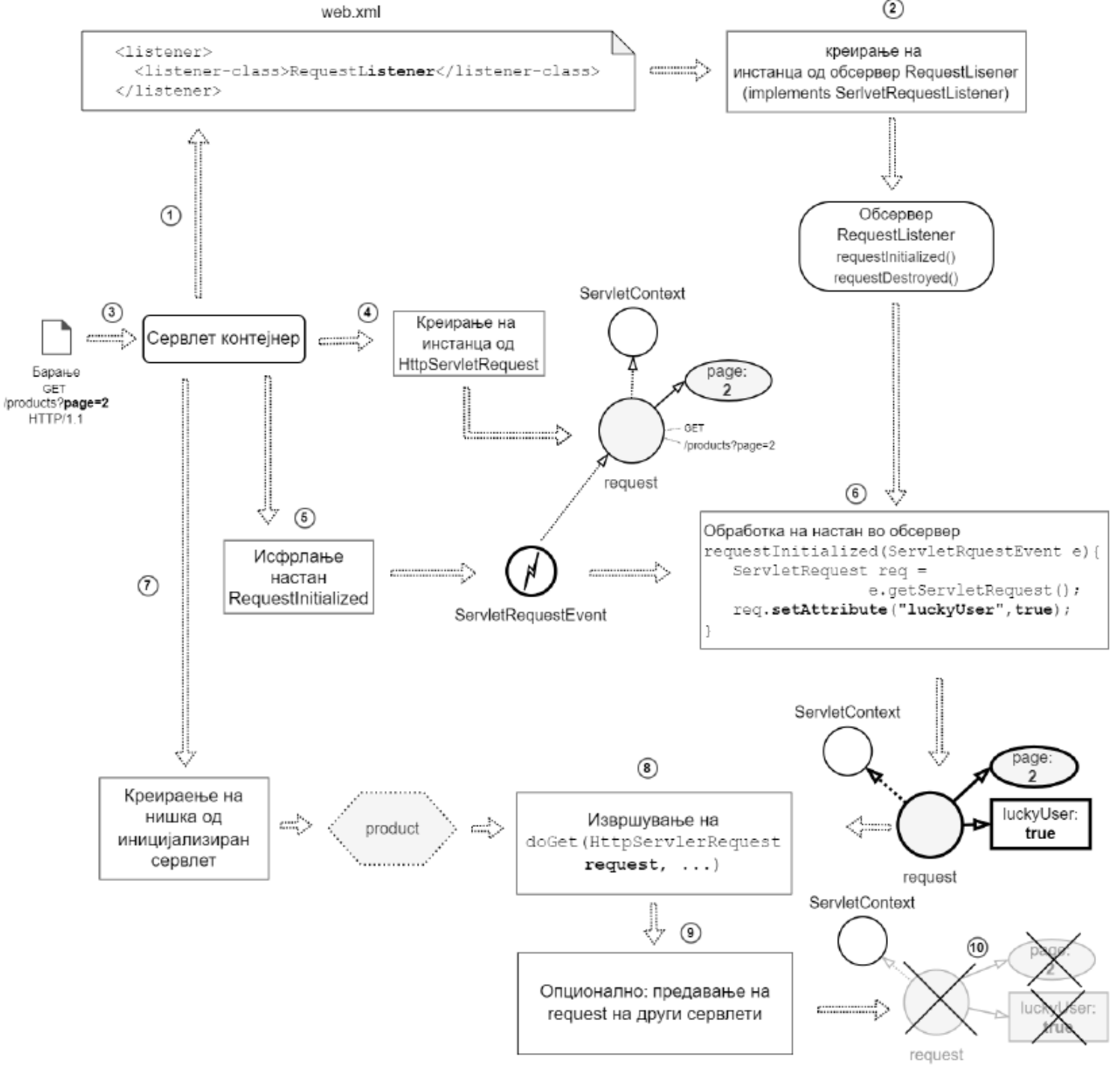
- При промена на атрибут на барање (креирање, промена и бришење), контејнерот исфрла настан од типот `ServletRequestAttributeEvent`
  - Пренесува контекст, атрибут и влредност
  - Пристап преку методите

```
public String getName()
public Object getValue()
```

- Настаните се обработуваат преку обсервер кој имплементира интерфејс `ServletRequestAttributeListener` со методи:

```
public void attributeAdded(ServletRequestAttributeEvent e)
public void attributeRemoved(ServletRequestAttributeEvent e)
public void attributeReplaced(ServletRequestAttributeEvent e)
```

Настани и обсервери на барање - преглед



Работа со податоци во сесиски опсег

- Имплементација на сесии преку колачиња
- За секој корисник, контејнерот креира сесиски објект `HttpSession`
  - Чува кориснички атрибути во текот на целата сесија на корисникот
- Креирање и пристап до објектот

```
public HttpSession getSession();
public HttpSession getSession(boolean create);
```

- Животен век на сесија
  - Предефинирана вредност: 30 мин
  - Поставка во web.config (во минути)

```
void setMaxInactiveInterval(int intervalInSeconds)
```

Работа со податоци во сесиски опсег

- Поважни методи на сесиски објект

```
public void invalidate()
public String getId()
public long getCreationTime()
public long getLastAccessedTime()
void setMaxInactiveInterval()
public boolean isNew()
```

- Кај сесиите не постојат параметри, туку само атрибути
  - Методи за читање и уредување на атрибути

```
public void setAttribute(String name, Object object)
public Object getAttribute(String name)
public void removeAttribute(String name):
```

Работа со податоци во сесиски опсег

- Пример

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
     ↪  response) throws IOException, ServletException {
        String username = request.getParameter("username");
        String password = request.getParameter("password");
        if (password.equals("spring")){
            HttpSession session = request.getSession()
            session.setAttribute("user", username)
            response.sendRedirect("home");
        }
    }
}
```

Работа со податоци во сесиски опсег

- Пример (продолжение)

```
@WebServlet("/home")
public class HomeServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
     ↪  response) throws IOException, ServletException {
        HttpSession session = request.getSession()
        if (session.isNew())
            response.sendRedirect("/login.html");
        }
        else {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession();
        String username = (String) session.getAttribute("user");
        out.println("Welcome " + username)
        }
    }
}
```

Настани и обсервери на сесија

- При промена на сесија (креирање и бришење) контејнерот фрла настан од типот `HttpSessionEvent`
  - Носи референца од сесијата

- Настаните се обработуваат преку обсервер кој имплементира интерфејс `HttpSessionListener` со методи:

```
public void sessionCreated(HttpSessionEvent e)
public void sessionDestroyed(HttpSessionEvent e)
```

Настани и обсервери на атрибути на сесија

- При промена на сесиски атрибути (креирање, промена и бришење), контејнерот исфрла настан од типот `HttpSessionBindEvent`
  - Пренесува референца до сесија, име и вредност на атрибут
  - Пристап преку методите

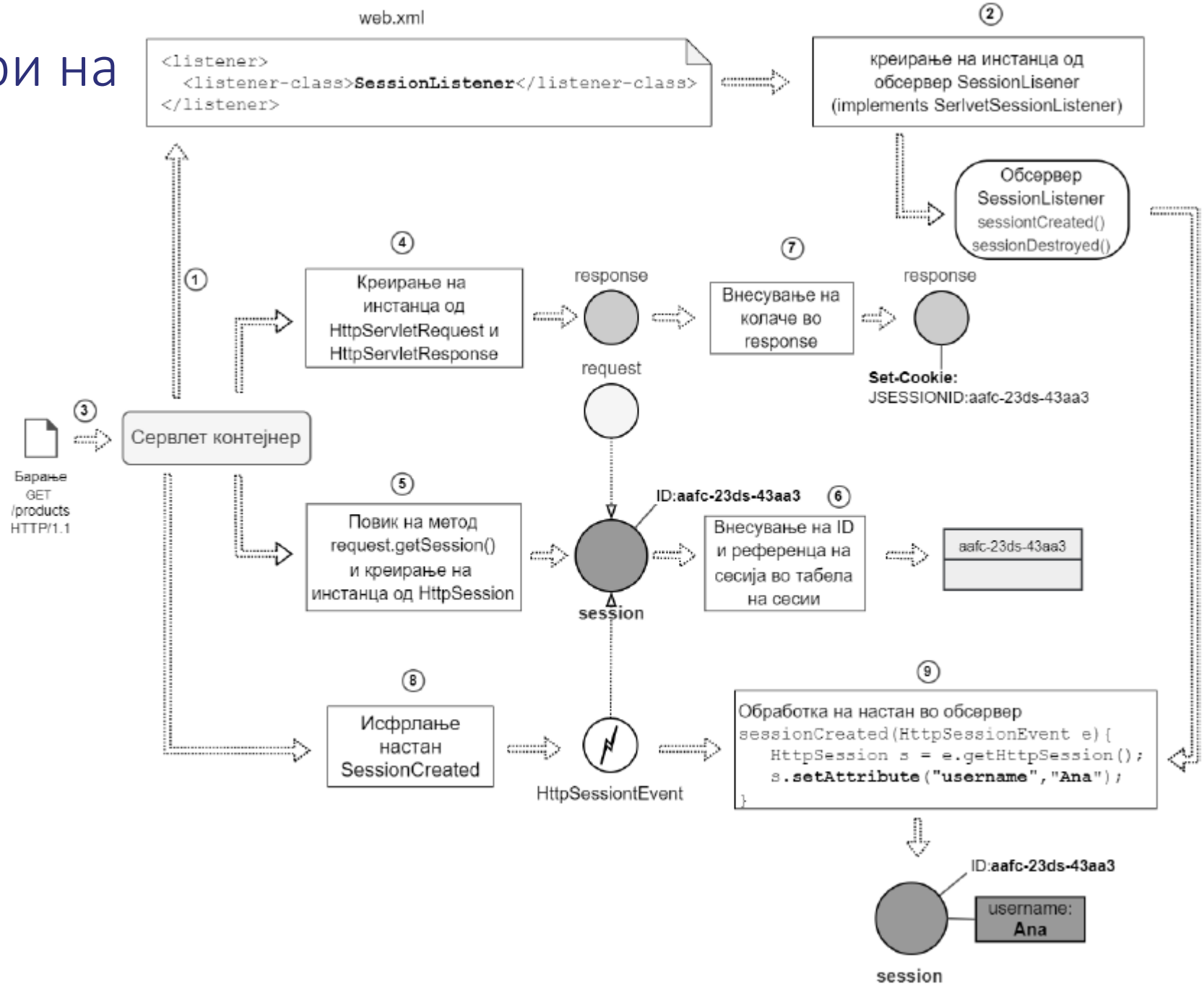
```
public String getName()
public Object getValue()
public HttpSession getSession()
```

- Настаните се обработуваат преку обсервер кој имплементира интерфејс `HttpSessionAtributeListener` со методи

```
public void attributeAdded(HttpSessionBindingEvent e)
public void attributeRemoved(HttpSessionBindingEvent e)
public void attributeReplaced(HttpSessionBindingEvent e)
```

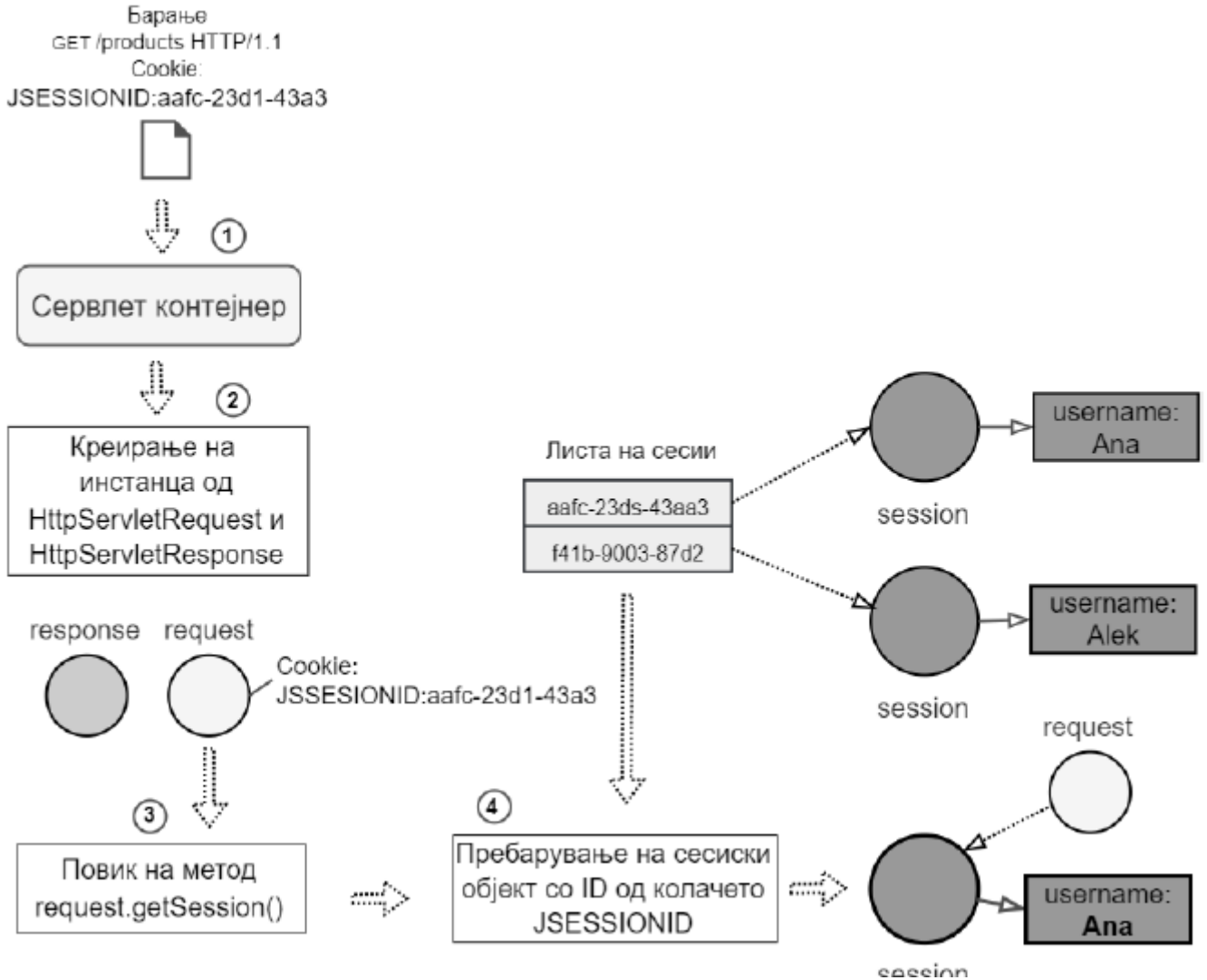
Настани и обсервери на сесија - преглед

Креирање нова сесија



Настани и обсервери на сесија - преглед

Пристап до постоечка сесија



Опсези на важење

- Барање
  - Параметри и атрибути
  - Активни во рамки на барање
- Сервлет
  - Параметри
  - Пристап од сите клиенти во рамките на сервлет
- Сесија
  - Атрибути
  - Пристап од било кој сервлет само од еден клиент
- Контекст
  - Параметри и атрибути
  - Пристап од било кој сервлет од било кој клиент

Hands-on

Работа со параметри и атрибути во различни опсези

- Index.html

```
<html>
    <body>
        <a href="/s1?page=1">Servlet 1</a><br>
        <a href="/s2?page=2">Servlet 2</a><br>
        <a href="/s3">Servlet 3</a><br>
    </body>
</html>
```

- Web.xml

```
<context-param>
    <param-name>size</param-name>
    <param-value>100</param-value>
</context-param>
```

- Слушачи на настани

```
@WebListener
public class MyContextListener implements ServletContextListener {
    public void contextInitialized(ServletContextEvent event) {
        ServletContext context = event.getServletContext();
        context.setAttribute("app-counter",0);
    }
    public void contextDestroyed(ServletContextEvent event) {
    }
}
@WebListener
public class MySessionListener implements HttpSessionListener {
    public class MySessionListener implements HttpSessionListener {
    public void sessionCreated(HttpSessionEvent event) {
        HttpSession session = event.getSession();
        session.setAttribute("user-counter",0);
    }
    public void sessionDestroyed(HttpSessionEvent event) {
    }
}
```

```java
@WebListener
public class MyRequestListener implements ServletRequestListener {
    public void requestInitialized(ServletRequestEvent event) {

        ServletRequest request = event.getServletRequest();
        request.setAttribute("request-counter",0);
    }
    public void requestDestroyed(ServletRequestEvent event) {
    }
}
```

- Servlet1

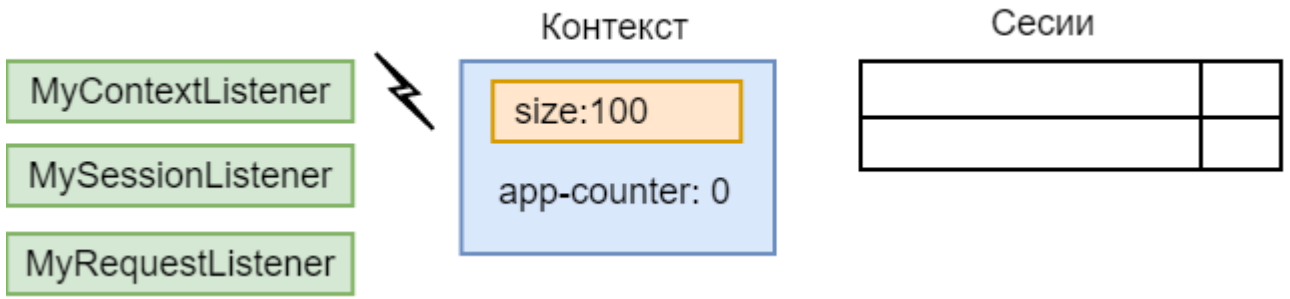
```
@WebServlet(name = "Servlet1", value = "/s1",
        initParams = {@WebInitParam(name = "pages", value = "10")})
public class ScopeServlet1 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
     ↪  response) throws IOException {
        ServletConfig config = getServletConfig();
        ServletContext context = getServletContext();
        HttpSession session = request.getSession();

        int appSize = Integer.parseInt(context.getInitParameter("size"));
        int configPages =
         ↪  Integer.parseInt(config.getInitParameter("pages"));
        int requestPage = Integer.parseInt(request.getParameter("page"));

        int appCounter = (int)context.getAttribute("app-counter");
        context.setAttribute("app-counter", ++appCounter);
        int userCounter =  (int)session.getAttribute("user-counter");
        session.setAttribute("user-counter", ++userCounter);
        int requestCounter = (int)request.getAttribute("request-counter");
        request.setAttribute("request-counter",++requestCounter);
```

```
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body><h4>"+config.getServletName()+"</h4>");
        out.println("Parameters: appSize = " + appSize + "  configPages =
        ↪  "+configPages+ "  requestPage = "+requestPage+"</br>");
        out.println("Attributes: appCounter = " + appCounter + "
        ↪  userCounter = "+userCounter+ "  requestCounter =
        ↪  "+requestCounter);
        out.println("</body></html>");
    }
}
```

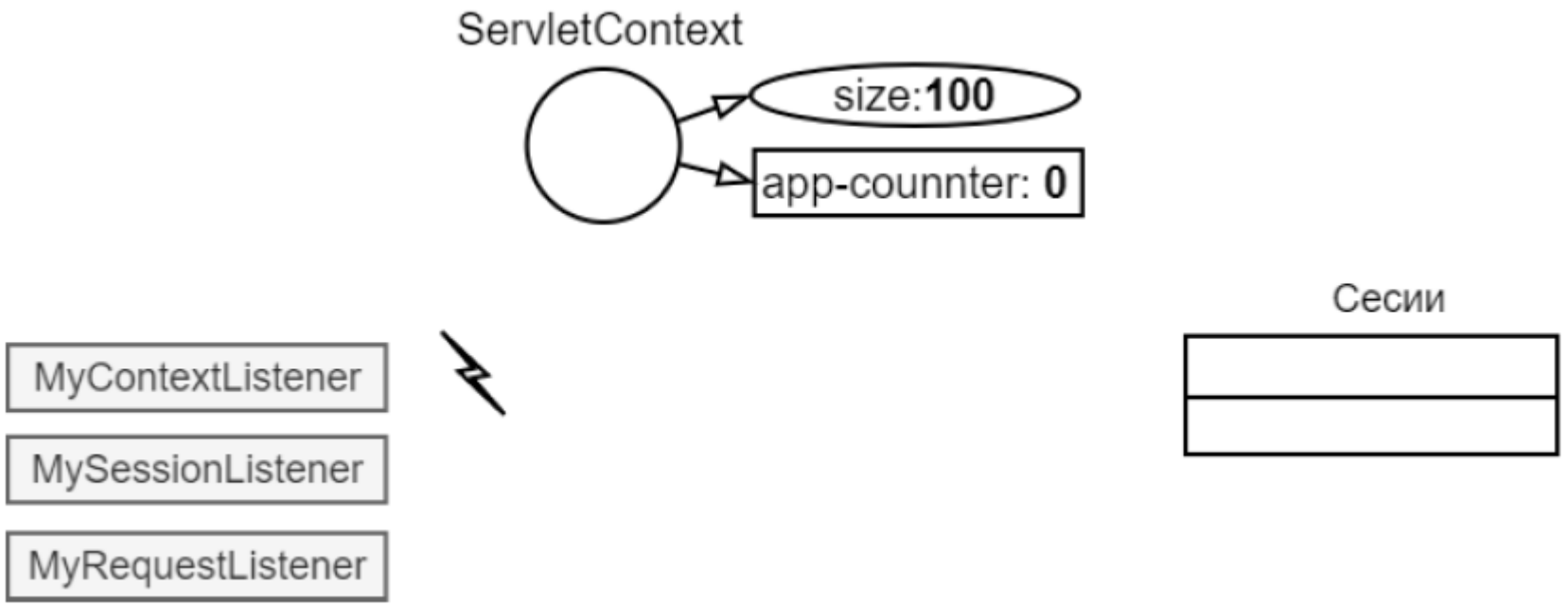
- Servlet2 и Servlet3

```
@WebServlet(name = "Servlet2", value = "/s2",
        initParams = {@WebInitParam(name = "pages", value = "20")})
public class ScopeServlet1 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
     ↪  response) throws IOException {
       //same code as doGet in Servlet1
    }
}
@WebServlet(name = "Servlet3", value = "/s3")
public class ScopeServlet3 extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
     ↪  response) throws IOException, ServletException {
        Integer requestCounter =
         ↪  (Integer)request.getAttribute("request-counter");
        request.setAttribute("request-counter",++requestCounter);
        RequestDispatcher rd = request.getRequestDispatcher("/s1?page=3");
        rd.forward(request,response);
    }
}
```

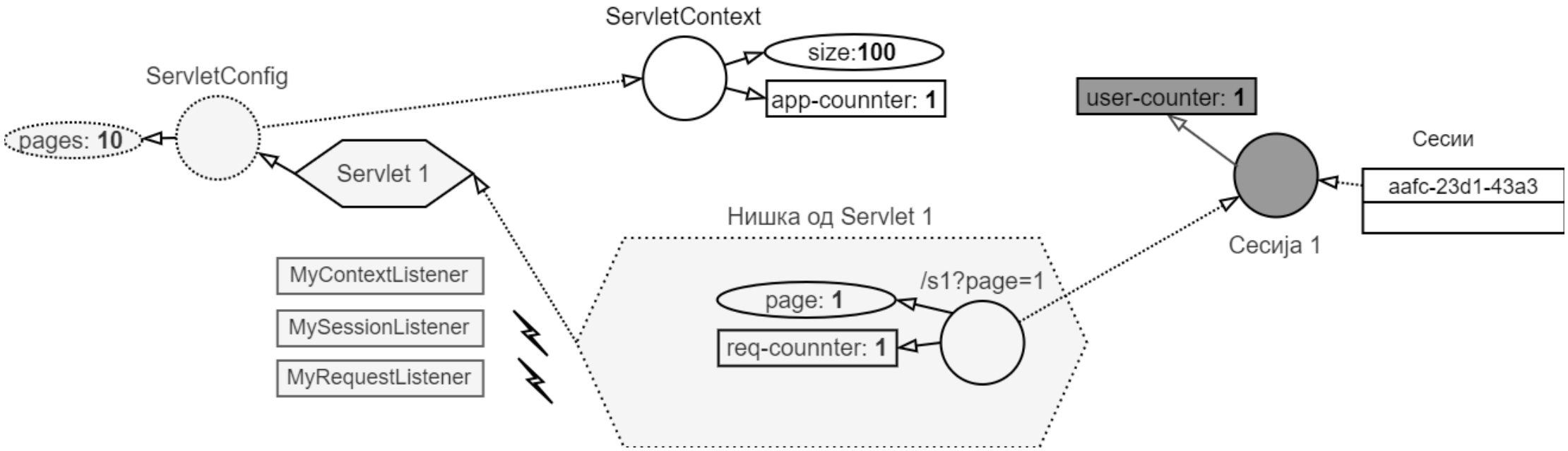
Почетна состојба



Почетна состојба

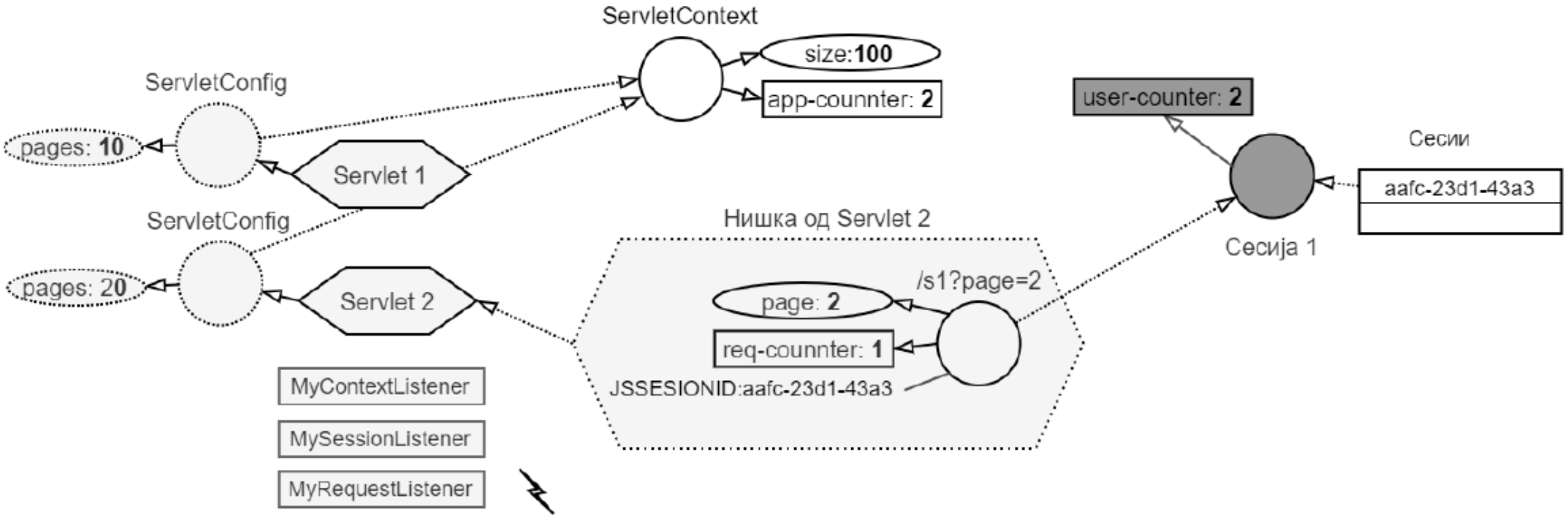


Повик на /s1?page=1



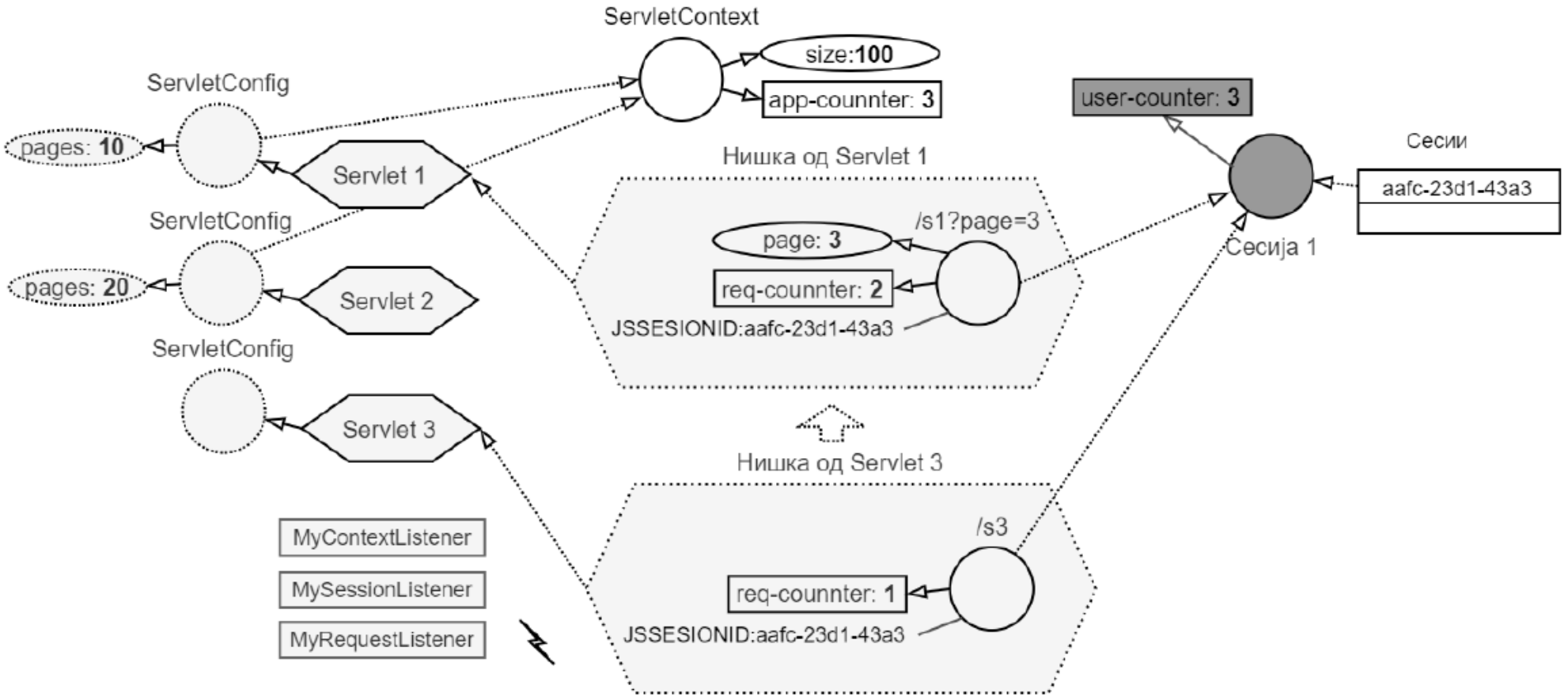

```
Servlet1
Parameters: appSize = 100 configPages = 10 requestPage = 1
Attributes: appCounter = 1 userCounter = 1 requestCounter = 1
```

Повик на /s2?page=2



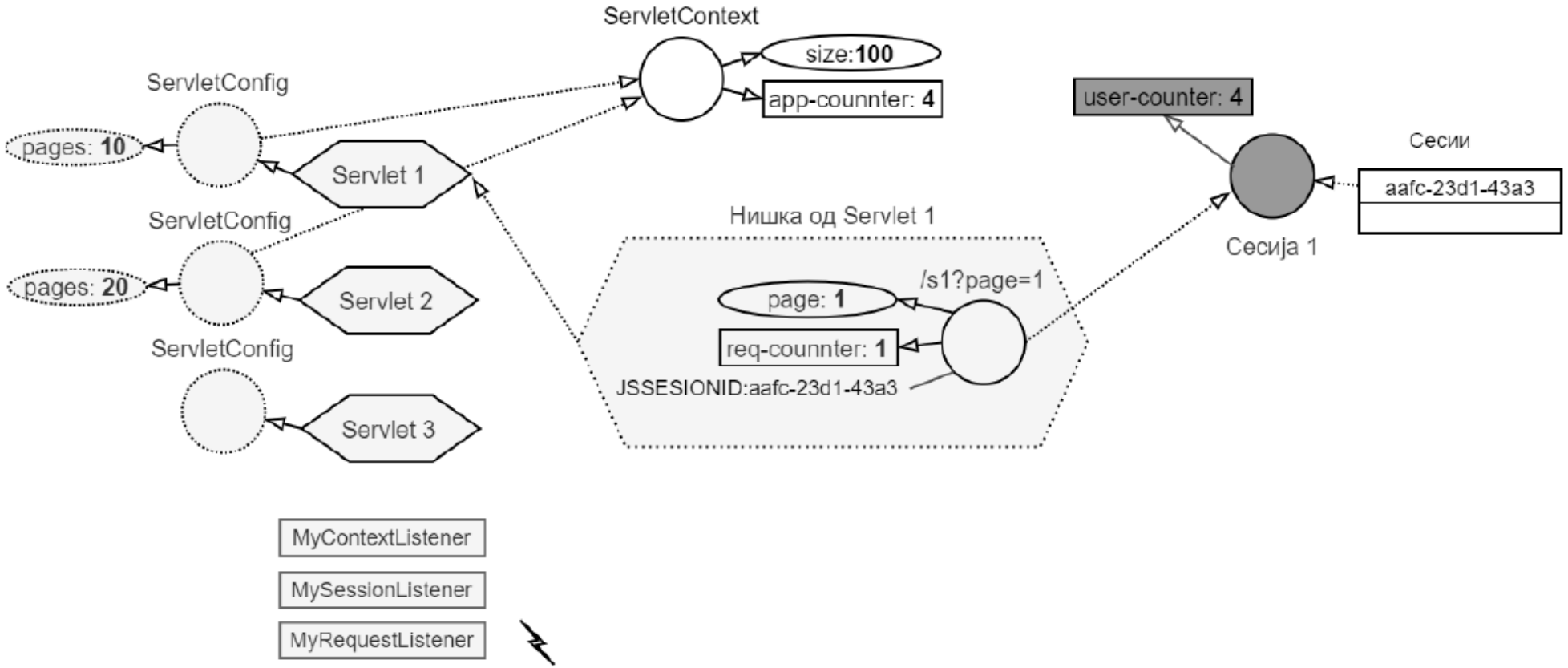

```
Servlet2
Parameters: appSize = 100 configPages = 20 requestPage = 2
Attributes: appCounter = 2 userCounter = 2 requestCounter = 1
```

Повик на /s3



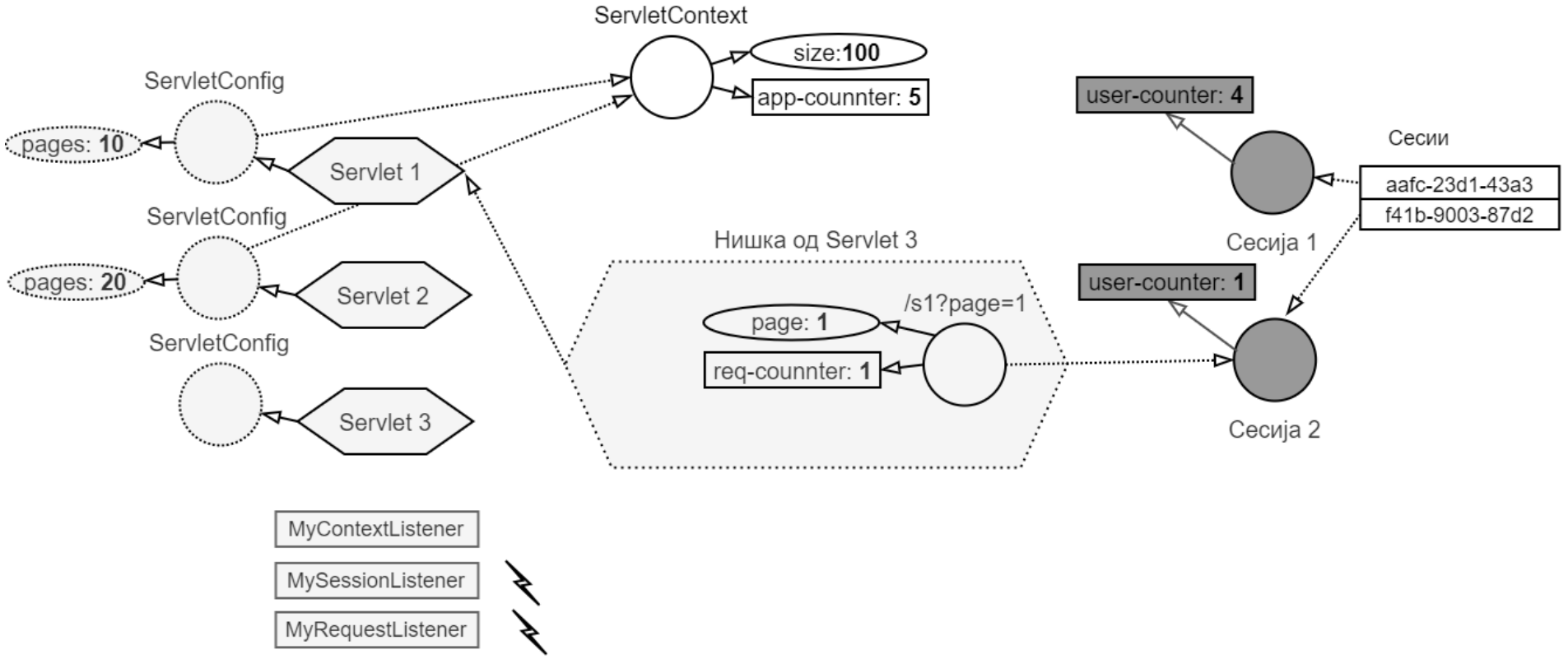

```
Servlet1
Parameters: appSize = 100 configPages = 10 requestPage = 3
Attributes: appCounter = 3 userCounter = 3 requestCounter = 2
```

Повик на /s1?page=1




```
Servlet1
Parameters: appSize = 100 configPages = 10 requestPage = 1
Attributes: appCounter = 4 userCounter = 4 requestCounter = 1
```

Повик на /s1?page=1 (incognito)




```
Servlet1
Parameters: appSize = 100 configPages = 10 requestPage = 1
Attributes: appCounter = 5 userCounter = 1 requestCounter = 1
```